Creating External Facing Web Sites with WebSphere Portal

Adam de Leeuw
Ali Beklen
David Axinn
Fanie Korff
Graham Wallace
Janice Taylor
JP Thambuganipalle
Projwal Saha
Rajaguru Balagurusamy
Raul Munoz Serrano
Bart Jacob
**Important Note about referencing the latest information**

This PDF Document represents a snapshot of the Web-based wiki content. It was created on January 11, 2010.

For the latest information contained within this Lotus Redbooks Wiki, please refer to

Table of Contents

Creating External Facing Web Sites with WebSphere Portal ......................................................... 1
  Important Note about referencing the latest information ......................................................... 2
Table of Contents ......................................................................................................................... 3
Preface .......................................................................................................................................... 15
  About the authors....................................................................................................................... 15
Differences between internal and external facing sites ........................................................... 19
  Killer applications .................................................................................................................... 19
  Internal sites ............................................................................................................................. 19
  External sites ............................................................................................................................ 20
About successful Web sites....................................................................................................... 21
  Information architecture and site design ................................................................................ 21
    Information architecture ......................................................................................................... 22
    Defining the information architecture .................................................................................. 22
    Web Content Management deliverables .............................................................................. 24
    WCM Web site decomposition .............................................................................................. 24
  User interface and user experience design ............................................................................ 25
  Usability ................................................................................................................................... 25
    Identifying the different user types ....................................................................................... 25
    80 20 rule .............................................................................................................................. 25
    Function and appearance ...................................................................................................... 25
    User centric design ............................................................................................................... 25
    Content .................................................................................................................................. 25
  Look and feel ............................................................................................................................ 26
    Clean design .......................................................................................................................... 26
    The KISS principle ................................................................................................................. 28
    Fixed versus fluid .................................................................................................................. 28
    Navigation ............................................................................................................................. 29
    Three clicks rule .................................................................................................................... 29
  Web 2 0 ..................................................................................................................................... 29
  Accessibility ............................................................................................................................... 30
    Accessibility in WebSphere Portal ......................................................................................... 30
  Browser compatibility ............................................................................................................... 31
    Define your support list .......................................................................................................... 31
  Content centric design ............................................................................................................ 32
    Understanding Web content management ............................................................................ 32
    Content and its management ................................................................................................. 32
    Web content types .................................................................................................................. 32
    Web content management ..................................................................................................... 33
      Web content management fundamentals ............................................................................ 33
      The need for Web content management ............................................................................. 34
    Defining a content centric site ............................................................................................... 38
  Introduction to IBM Lotus Web Content Management ........................................................... 38
    Lotus Web Content Management new features and improvements .................................... 38
  IBM Lotus Web Content Management Architecture ............................................................. 40
    Architectural frameworks for content in WCM ................................................................. 40
    Information architecture and site design ............................................................................. 41
      Defining the information architecture ................................................................................ 41
      Physical architecture best practices .................................................................................. 42
  IBM Lotus Web Content Management system components .................................................. 48
    IBM Lotus Web Content Management Server ...................................................................... 48
    IBM Lotus Web Content Management Content Repository .............................................. 48
    IBM Lotus Web Content Management Authoring Portlet ................................................... 48
    IBM Lotus Web Content Management Connect Servlet ..................................................... 48
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>JSF application architecture</td>
<td>112</td>
</tr>
<tr>
<td>JSF advantages and disadvantages</td>
<td>115</td>
</tr>
<tr>
<td>Spring 2.0</td>
<td>116</td>
</tr>
<tr>
<td>Model</td>
<td>116</td>
</tr>
<tr>
<td>View</td>
<td>116</td>
</tr>
<tr>
<td>Controller</td>
<td>116</td>
</tr>
<tr>
<td>Adobe Flex</td>
<td>125</td>
</tr>
<tr>
<td>Deciding on the right framework</td>
<td>126</td>
</tr>
<tr>
<td>Personalization components</td>
<td>127</td>
</tr>
<tr>
<td>Personalization browser</td>
<td>127</td>
</tr>
<tr>
<td>Rules engine</td>
<td>128</td>
</tr>
<tr>
<td>LikeMinds recommendation engine</td>
<td>128</td>
</tr>
<tr>
<td>Resource engine</td>
<td>128</td>
</tr>
<tr>
<td>Logging framework</td>
<td>128</td>
</tr>
<tr>
<td>Four approaches to personalization</td>
<td>128</td>
</tr>
<tr>
<td>Security</td>
<td>128</td>
</tr>
<tr>
<td>User preferences</td>
<td>128</td>
</tr>
<tr>
<td>Rules engine</td>
<td>129</td>
</tr>
<tr>
<td>Collaborative filtering</td>
<td>129</td>
</tr>
<tr>
<td>Rules driven personalization</td>
<td>129</td>
</tr>
<tr>
<td>Profiling</td>
<td>129</td>
</tr>
<tr>
<td>Selection</td>
<td>130</td>
</tr>
<tr>
<td>Binding</td>
<td>131</td>
</tr>
<tr>
<td>Email</td>
<td>133</td>
</tr>
<tr>
<td>Content spot</td>
<td>134</td>
</tr>
<tr>
<td>Campaigns</td>
<td>135</td>
</tr>
<tr>
<td>Recommendation</td>
<td>135</td>
</tr>
<tr>
<td>Visibility</td>
<td>136</td>
</tr>
<tr>
<td>Extending personalization</td>
<td>137</td>
</tr>
<tr>
<td>Sample scenarios</td>
<td>137</td>
</tr>
<tr>
<td>Personalization scenario 1</td>
<td>138</td>
</tr>
<tr>
<td>Personalization scenario 2</td>
<td>144</td>
</tr>
<tr>
<td>Personalization demonstration sample</td>
<td>153</td>
</tr>
<tr>
<td>Summary</td>
<td>154</td>
</tr>
<tr>
<td>Search</td>
<td>155</td>
</tr>
<tr>
<td>Search engine optimization</td>
<td>155</td>
</tr>
<tr>
<td>Overview of Search Engine Optimization</td>
<td>155</td>
</tr>
<tr>
<td>Definition of terms</td>
<td>155</td>
</tr>
<tr>
<td>How search engines work</td>
<td>156</td>
</tr>
<tr>
<td>More on Google</td>
<td>156</td>
</tr>
<tr>
<td>How Internet search engines work</td>
<td>159</td>
</tr>
<tr>
<td>Challenges with portal crawlability</td>
<td>159</td>
</tr>
<tr>
<td>Portal crawler awareness and normalized URLs</td>
<td>160</td>
</tr>
<tr>
<td>Sitemap portlet</td>
<td>160</td>
</tr>
<tr>
<td>External crawlability of Portal with WCM content</td>
<td>160</td>
</tr>
<tr>
<td>Web 2.0 and search</td>
<td>161</td>
</tr>
<tr>
<td>Case study using the sitemap portlet</td>
<td>161</td>
</tr>
<tr>
<td>Searching and crawling portal and other sites with portal search</td>
<td>171</td>
</tr>
<tr>
<td>Understanding portal search</td>
<td>171</td>
</tr>
<tr>
<td>Portal search architecture</td>
<td>171</td>
</tr>
<tr>
<td>Search services collections and scopes</td>
<td>172</td>
</tr>
<tr>
<td>Search and administration user interfaces</td>
<td>173</td>
</tr>
<tr>
<td>Portal search APIs</td>
<td>178</td>
</tr>
<tr>
<td>Seedlist framework</td>
<td>178</td>
</tr>
<tr>
<td>Crawlability of portal with WCM content</td>
<td>179</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>Categorization and taxonomy</td>
<td>179</td>
</tr>
<tr>
<td>Case study crawling the portal and displaying search results</td>
<td>180</td>
</tr>
<tr>
<td>Site analytics and optimization</td>
<td>189</td>
</tr>
<tr>
<td>Server side site analytics</td>
<td>190</td>
</tr>
<tr>
<td>Server side site analytics architecture</td>
<td>190</td>
</tr>
<tr>
<td>Server side loggers</td>
<td>190</td>
</tr>
<tr>
<td>Analytics and reporting with WCM</td>
<td>191</td>
</tr>
<tr>
<td>Active site analytics</td>
<td>191</td>
</tr>
<tr>
<td>Instrumenting a theme for active site analytics</td>
<td>193</td>
</tr>
<tr>
<td>Differences between server side and active site analytics</td>
<td>193</td>
</tr>
<tr>
<td>Server side</td>
<td>193</td>
</tr>
<tr>
<td>Active site</td>
<td>194</td>
</tr>
<tr>
<td>User experience optimization initiative</td>
<td>194</td>
</tr>
<tr>
<td>Future outlook and summary</td>
<td>194</td>
</tr>
<tr>
<td>Additional information and resources</td>
<td>194</td>
</tr>
<tr>
<td>Mobile device support</td>
<td>196</td>
</tr>
<tr>
<td>Overview</td>
<td>196</td>
</tr>
<tr>
<td>Mobile Portal Accelerator</td>
<td>196</td>
</tr>
<tr>
<td>New features in Mobile Portal Accelerator</td>
<td>198</td>
</tr>
<tr>
<td>Customer experiences</td>
<td>199</td>
</tr>
<tr>
<td>High level architecture</td>
<td>199</td>
</tr>
<tr>
<td>Multi Channel Server</td>
<td>200</td>
</tr>
<tr>
<td>Mobile Portal Accelerator extensions</td>
<td>200</td>
</tr>
<tr>
<td>Manage Mobile Pages portlet</td>
<td>201</td>
</tr>
<tr>
<td>Extended Properties portlet</td>
<td>201</td>
</tr>
<tr>
<td>XDIME Aggregator</td>
<td>201</td>
</tr>
<tr>
<td>Mobile Portal Toolkit</td>
<td>201</td>
</tr>
<tr>
<td>Runtime component interaction</td>
<td>202</td>
</tr>
<tr>
<td>Installation hints and tips</td>
<td>203</td>
</tr>
<tr>
<td>Create a DB2 MCS database</td>
<td>203</td>
</tr>
<tr>
<td>Installing MPA extensions</td>
<td>206</td>
</tr>
<tr>
<td>Installing Mobile Portal Toolkit</td>
<td>207</td>
</tr>
<tr>
<td>Validating installation of mobile portlets</td>
<td>207</td>
</tr>
<tr>
<td>Tools for Mobile Portal Accelerator</td>
<td>208</td>
</tr>
<tr>
<td>Tools for authoring content in LWCM</td>
<td>208</td>
</tr>
<tr>
<td>Design considerations for your mobile portal</td>
<td>209</td>
</tr>
<tr>
<td>MPA user interface</td>
<td>211</td>
</tr>
<tr>
<td>Preload notice portlet</td>
<td>213</td>
</tr>
<tr>
<td>Extended Properties portlet</td>
<td>214</td>
</tr>
<tr>
<td>Developing mobile portlets</td>
<td>216</td>
</tr>
<tr>
<td>Mobile Portal Toolkit</td>
<td>216</td>
</tr>
<tr>
<td>MPA client framework</td>
<td>223</td>
</tr>
<tr>
<td>MPA widgets</td>
<td>223</td>
</tr>
<tr>
<td>Sample scenario</td>
<td>226</td>
</tr>
<tr>
<td>Delivering mobile content</td>
<td>227</td>
</tr>
<tr>
<td>Integrating MPA with Lotus Web Content Management</td>
<td>228</td>
</tr>
<tr>
<td>Create a new WCM library</td>
<td>228</td>
</tr>
<tr>
<td>Assign role to library</td>
<td>228</td>
</tr>
</tbody>
</table>
Assign role to item types

Configuring Web Content Libraries

Library Resources RiverBend

Set the permissions on the item types for this Web Content library.

<table>
<thead>
<tr>
<th>Library Name</th>
<th>229</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authoring Templates</td>
<td></td>
</tr>
<tr>
<td>Components</td>
<td></td>
</tr>
<tr>
<td>Content</td>
<td></td>
</tr>
<tr>
<td>Presentation Templates</td>
<td></td>
</tr>
<tr>
<td>Site and Site Areas</td>
<td></td>
</tr>
<tr>
<td>Taxonomy</td>
<td></td>
</tr>
<tr>
<td>Workflow and workflow elements</td>
<td></td>
</tr>
</tbody>
</table>

Configure WCM authoring portlet to view library ................................................................. 230
Create workflow .................................................................................................................... 231
Create site ............................................................................................................................ 236
Create site area ..................................................................................................................... 237
Create an authoring template ................................................................................................. 240
Create HTML presentation template for home page ............................................................... 242
Create content ......................................................................................................................... 243
Create template mappings for default content ...................................................................... 245
Configure portlets to display LWCM content on PC based browser .................................... 246
Display LWCM content on mobile devices .................................................................................. 249
Update existing authoring template to handle non HTML elements ..................................... 249
Create new XDIME presentation template .................................................................................. 250
Update content .......................................................................................................................... 250
Update template mappings for default content ........................................................................ 251
Configure portlets to display LWCM content on mobile devices ............................................ 252

1. Log in to WebSphere Portal as administrator.  
2. Click the Administration tab to open WebSphere Portal Administration.  
3. Click Portlet Management.  
4. Click Portlets.  
5. In the Search window, type Web and click Search.  
6. Locate the Web Content Viewer portlet and click Copy Portlets.  
7. Enter Name for portlet (for example, Mobile Web Content Viewer)  
8. Click OK.  
9. From the WebSphere Portal page, select Manage Mobile Pages or Manage Pages.  
10. Expand Content Root > Home  
11. In the left panel under Manage Mobile Pages or Manage Pages, click the New Page icon.  
12. Under Title, type Mobile Test Page  
13. For the Theme, select Mobile Portal 14. In the Type of Page section, select the XDIME option  
Note: If the xdime markup language is checked without support for html, the Mobile Test page will not display in the PC-web based browser. The HTML setting should remain until testing of web site is not necessary.
15. Click OK. Note: By default, only one root node (page) is handled by a mobile device, so make any previous pages that support XDIME inactive or your page will not display on mobile device.

16. In the left panel under Manage Mobile Pages or Manage Pages, select Mobile Test Page.

17. In the Portlets section of the right panel, click Edit.

18. In the left panel, click Add portlets.

19. Set the search by Title starts with and type Mobile RiverBend Web Content Viewer portlet name (Note: add Mobile Login portlet to layout as well).

20. Select portlet and click OK.

21. Click Done.

22. Click the down arrow next to Mobile Test Page and select Edit Shared Setting.

23. In the Content section, click Edit.

24. In the Sites section, select the content.

25. In the Alternate Presentation Template section, click Edit.

26. Select Home Page PT XDIME.

27. Click OK.

28. In the Broadcast Links To section, select None.

29. In the Select a target portal page section, select Mobile Test page.

30. In the Receive Links From section, select the Other portlets and this portlet option.
31. Click OK. 32. Login via Mobile Login portlet to access website
Welcome to the River Bend Tea and Coffee company

Our company River Bend has been established since 1973, for 30+ years we've been in the business of making and serving the best coffee and tea. Today River Bend is a thriving business and community and we welcome you to our site, and hope you enjoy your experience when you next visit one of our stores. Take a look at our specials menu - we've added up some tasty snacks and lunch time specials.
Welcome to the River Bend Tea and Coffee company

Our company River Bend has been established since 1971, for 30+ years we’ve been in the business of making our own special blends of tea and coffee. Today River Bend is a thriving business and community and we welcome you to our site, and hope you enjoy your experience when you next visit one of our stores. Take a look at our specials menu - we’ve cooked up some tasty deals and lunch time specials.
Virtual portal support................................................................. 260
MPA device subscription service ............................................... 260
Preface

This wiki addresses considerations for the creation of external-facing Web sites utilizing IBM WebSphere Portal Version 6.1.5. We focus on areas related to external facing Web sites versus topics that might apply to only internal-facing sites.

In the first section, we provide a brief overview of the differences between internal and external facing Web sites. We then provide an overview of what factors contribute to successful Web sites. Next we provide more technical content related to Web Content Management and Web 2.0 considerations. We then discuss the various UI frameworks supported by WebSphere Portal and personalization for the end user.

We include an extensive discussion related to search and the integration of search engines. Section 8 covers site analytics and optimization and we conclude this wiki with information on mobile device support.

Throughout this wiki we have included examples, screen captures and code samples based on various scenarios.

About the authors

The initial structure and content of this wiki was built by a team of experts from around the world.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bart Jacob</td>
<td>Bart Jacob is a Senior Consulting IT Specialist at the IBM International Technical Support Organization, Austin Center. He has over 25 years of experience providing technical support across a variety of IBM products and technologies, including communications, object oriented software development, and systems management. He joined the ITSO in 1989, where he has been writing IBM Redbooks publications and creating and teaching workshops around the world on a variety of topics. He holds a Masters degree in Numerical Analysis from Syracuse University.</td>
</tr>
<tr>
<td>Adam de Leeuw</td>
<td>Adam de Leeuw is an accredited senior IT specialist with 12 years experience. He works in IBM Software Services for WebSphere at IBM Hursley in the UK and provides consultancy on client technologies including WebSphere Portal, Web 2.0 and Lotus Expeditor.</td>
</tr>
<tr>
<td><strong>Ali Beklen</strong></td>
<td>Ali Beklen is a Software IT Architect working for IBM Turkey. He is experienced on designing integration solutions and he is Open Group Certified IT Specialist on collaboration solutions. Ali holds a Master of Computer Engineering degree from Maltepe University in Istanbul (Turkey).</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td><strong>David Axinn</strong></td>
<td>David Axinn is a Certified IT Specialist with IBM Lotus and WebSphere Portal Technical Sales in Washington, DC. He has deep technical expertise and a strong business understanding of Lotus strategy, portfolio and related solutions. His product focus includes WebSphere Portal, Lotus Connections, Lotus Quickr, Lotus Sametime and Lotus Notes / Domino, and he has lead many projects to success in this area.</td>
</tr>
</tbody>
</table>
| **Fanie Korff** | Fanie Korff has been involved in the IT industry as a Web development specialist and Senior IT consultant for the past 12 years. His tasks and responsibilities have covered WebSphere Portal and Web Content Management (WCM).

Fanie is currently a Senior WCM Consultant at UK Business Partner Open Logic, UK leaders in WebSphere Portal. Over the past six years Fanie has worked as a specialist across a number of industries, implementing global solutions on the IBM WebSphere Portal platform. These have included installation, configuration, information architecture, development, training, usability and accessibility features as well as functionality of Web Content Management.

At Open Logic, Fanie has worked with a variety of customers including Cardiff University, Compass Group and The University of London. |
| **Graham Wallace** | Graham has a very strong background in designing usable user interfaces particularly in websites. His speciality is web usability and accessibility, where he has worked on various sites over the past 5 years. He has recently started working for UK Business Partner Open Logic. |
Janice Taylor

Janice Taylor is a Certified Senior IT Specialist within the Global Solution Center (GSC) in Coppell, TX. She provides technical sales support (TSS) for WebSphere Portal, Rational Team Concert, and Rational Rhapsody. She has been focused on designing and developing innovative, cross-brand, cross-platform pre-sales portal solutions across all sectors for over six years.

JP Thambukanipalle

JayaPrasad (JP) Thambukanipalle is a Premium Support Manager (PSM) for Websphere Portal and prior to becoming a PSM, JP was a lead portal architect in IBM Software Services for Lotus (ISSL). Prior to joining IBM, JP worked with leading global investment and/or retails banks as an infrastructure specialist/architect.

JP has presented at various IBM technical conferences.

Projjwal Saha

Projjwal Saha is a Software Engineer as a part of Software Group, based in India Software Labs, Gurgaon. He has worked in IBM for over 4 years. He has experience in development and testing of various Portal and Collaboration suite of products. He has also worked as a support engineer for WebSphere Portal 5.x - 6.0.x to customers worldwide and develop prototypes as per customers requirements.

Rajaguru Balagurusamy

Rajaguru Balagurusamy is an IT Specialist with IBM India Software Labs. He specializes in IBM WebSphere Portal, IBM Lotus Forms, IBM WebSphere Portlet Factory, and J2EE development. He began his career as a WebSphere Portal and J2EE developer. In his current role, he has been helping customer in developing Portal using IBM WebSphere Portal Server and development of e-Forms using IBM Lotus Forms. Rajaguru has presented sessions at various IBM Lotus Technical Conferences. He also published article for DeveloperWorks that explains the various ways that you can integrate Lotus Forms with Lotus Domino and details of the benefits that integration brings. He also got certified in IBM WebSphere Portal Application Development.

Raul Munoz Serrano

Raul Munoz Serrano is an Advisory IT Specialist in IBM Software Group. He lives in the beautiful city of Madrid, Spain. He has been working with Lotus technologies for ten years, starting with the Domino family of products and moving to Portal in the early Portal 4.0 days. He likes playing with his children and running long distances from time to time.

Special thanks to the following people for their support and contributions to this project:

Andreas Prokoph
Barry Pellas
Carsten Leue
Differences between internal and external facing sites

We start this wiki with a short overview of some of the different considerations for internal Web sites versus external Web sites and the applications that drive them. The focus of this wiki is on using WebSphere Portal to develop external facing Web sites and therefore it is important to keep these differences in mind when reading the content that is presented throughout this wiki.

<table>
<thead>
<tr>
<th>Internal</th>
<th>External</th>
</tr>
</thead>
<tbody>
<tr>
<td>Usually aimed at members of an organisation, so it is built for only a single audience</td>
<td>Open to the anyone who can find the site, however certain users will be more interested in the site so these users can be favoured</td>
</tr>
<tr>
<td>Accessibility can be at a more basic level as there is a specific and defined user group</td>
<td>Accessibility is broader as anyone could be accessing the site</td>
</tr>
<tr>
<td>Tend to have a clear set of applications defined by the user’s role</td>
<td>Tend to have a narrower set of parameters defined by the stakeholders</td>
</tr>
<tr>
<td>Browser compatibility important but may consist of a smaller support list, that is, an organisation may force use of a particular browser</td>
<td>Browser compatibility more important and has a broader user base</td>
</tr>
<tr>
<td>Potential Uses</td>
<td>Potential Uses</td>
</tr>
<tr>
<td>• Collaboration</td>
<td>• Brand Building</td>
</tr>
<tr>
<td>• Content delivery</td>
<td>• Product or Service delivery</td>
</tr>
<tr>
<td>• Internal news</td>
<td>• Knowledge/Information sharing</td>
</tr>
<tr>
<td>• Alerts</td>
<td>• Community building</td>
</tr>
<tr>
<td>• User specific content</td>
<td>• Entertainment</td>
</tr>
<tr>
<td>• Access to online apps</td>
<td>• Customer support</td>
</tr>
<tr>
<td>• Sales management</td>
<td></td>
</tr>
<tr>
<td>• Customer relationship management</td>
<td></td>
</tr>
<tr>
<td>• Tech support</td>
<td></td>
</tr>
</tbody>
</table>

In addition to the above, search engine optimization can be of particular importance for external facing Web sites. See the section on search engine optimization on the Search page of this wiki.

Killer applications

Something that all Web sites need to succeed is a killer application; a function or service that is so critical to the user that the system that has it is more valuable to them than any other alternative. By using the correct framework, such as Lotus Web Content Management (WCM) with WebSphere Portal, the tools are provided to the system to implement such features.

Internal sites

An Internal site has a lot more control of its audience and because the most common use of these sites are from businesses, implementing a killer application is a lot easier. For example if the organization only allows the user to book annual leave through their intranet then they ensure that the site has a stick and carrot to pull in a user. Also an organization does not need to be concerned with a competitive Web site offering the same service.

Due to the nature of internal sites outlined in the table above the emphasis of the system comes down to how easily the site can be navigated. The sites face similar challenges to an external site in terms of creating a positive user experience. However, as these sites are used by a smaller number of people for a longer time any inefficiency is amplified as they can hold up and frustrate a user multiple times.
External sites

The implementation of a killer application in an external site is more difficult, as there is a higher level of competition. Although sometimes it is not important to have, in the case of a brand building site for a product, for example, it may not be as important to pull people into the system as the product itself may be the hook for the user. On the other hand a social networking site will need a killer application, since without something to differentiate it from other sites users will simply not use it.

With an external site the emphasis is placed on how the site is perceived by the user and how positive their experience is. This encompasses aspects such as the look and feel, the usability, how the content is structured, how relevant the content is to them, how impressed they are with the functionality, how easy the site is to use, and if the site looks good on the platform on which they are viewing it.
About successful Web sites

The measure of success for a Web site varies by the type of site it is. For example, an e-Commerce site can measure sales to judge how well it is doing. However, there are points that are relevant to all Web sites such as user retention, user enticement and usability design. To give a site the best possible chance of hitting its goals it is best to lay out best practices in the design phase. In this chapter, some elements of best practices including how to best define a project scope and key considerations in the user interface design will be outlined.

Information architecture and site design

The table below outlines common challenges associated with Web content.

<table>
<thead>
<tr>
<th>Challenge 1</th>
<th>Challenge 2</th>
<th>Challenge 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Keeping your site current</td>
<td>Content Chaos</td>
<td>Bringing visitors back</td>
</tr>
<tr>
<td>Why is it a challenge?</td>
<td>Why is it a challenge?</td>
<td>Why is it a challenge?</td>
</tr>
<tr>
<td>• Only a small group of technical people can change the web site</td>
<td>• Where did that come from?</td>
<td>• Low attention span</td>
</tr>
<tr>
<td>• It doesn’t demand attention</td>
<td>• Who owns it?</td>
<td>• Lots of completion</td>
</tr>
<tr>
<td>• Cross-functional effort</td>
<td>• How old is it?</td>
<td>• Easy to leave – one click and they’re gone</td>
</tr>
<tr>
<td>• Time consuming</td>
<td>• Are we still selling that product?</td>
<td></td>
</tr>
<tr>
<td>• Why are really old events still listed?</td>
<td>• Why are really old events still listed?</td>
<td></td>
</tr>
<tr>
<td>Cost of Outdated Pages</td>
<td>Cost of Chaos</td>
<td>Cost of Losing Customers</td>
</tr>
<tr>
<td>• Lost customer confidence</td>
<td>• The hassle of keeping the site current results in neglect</td>
<td>• If they run into problems, 19% of customers will never come back</td>
</tr>
<tr>
<td>• Damished image</td>
<td>• Site usage declines</td>
<td>• It costs 6 times more to sell to a new customer than to an existing one</td>
</tr>
<tr>
<td>• Lower traffic</td>
<td>• Content re-creation</td>
<td>• You can increase profits 85% by increasing customer retention by 8%</td>
</tr>
<tr>
<td>• Lost opportunities</td>
<td>• Content editor frustration</td>
<td></td>
</tr>
<tr>
<td>• Legal exposure</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The Information Architecture (IA) of a system is all the information located within a system expressed in a defined and structured way. It is critical to the design of a Web site, as it not only organizes the content but also lays the groundwork for the user experience with the system. A strong information architecture ensures that when the site is designed, all critical aspects of the site are addressed and there are no ambiguities in the system. Defining the information architecture ensures that the development team has a good understanding of the content that is to be displayed.

It is important to acknowledge the relationship between the information architecture and the WCM site framework. The WCM site framework structures the information about a published site and is closely integrated with Web site navigation.

To begin, you define an initial WCM site framework with primary and secondary site areas. The WCM site framework is a prerequisite for the design and development of the content management system. In addition, you must develop an initial layout of the home page based on the primary and secondary site areas. Both the framework and the home page layout require review and validation by the project owner, from both an authoring and Web usability viewpoint.

Choosing the right framework for a site dictates how the development teams approach the information architecture. WebSphere Portal provides other things such as security, content aggregation or user
personalization. Information needs to be structured at an early point to ensure that there are no problems further down the line. For example, if a user role is not modeled correctly in the system, then there is a chance that an element of the site that is too rigid, such as the design, may need to be reworked.

**Information architecture**
The following outlines the components and that make up an information architecture.

**How to classify the content**
The content classification shows all the different types of content that will be shown on the site.

Defined with:

- Content audit
- Site framework (only an outline of what the site framework will be is defined here)

**The hierarchical structure of the site**
This gives a clear picture of the way the content fits together in the context of the Web site. It shows how the site content fits into the different areas of the site.

Defined with:

- Site framework
- Site map

**The functionality that is required on the site**
This gives the behavior of the site and what functions the site performs. It also provides a detailed account of the site capabilities.

Defined with:

- Requirements
- Use cases

**The look of individual pages**
This shows how the content fits in with the functionality of the site, how the user flows through the site and details the look and feel of the site.

Defined with:

- Wireframes
- Storyboards

**The flexibility of the architecture to allow the business to evolve**
To ensure that site can grow and progress naturally the ability to evolve has to be taken into consideration in each section of the site. However, there will be times that an area of a site will not be suitable to grow. This should be captured and fed back to the project owners.

**Defining the information architecture**
The following sections describe the key steps to defining and information architecture.

**Requirements gathering**
The best way to ensure that you have an easy to use and intuitive Web site is to have a clear idea of the scope of the site from an early stage. This gives the developer a clearly defined set of goals to work towards and provides a measurable set of parameters for the site. It also defines an end point for the site development.

It is useful at this stage to identify aspects of the demographic that will be targeted by the Web site and how
the requirements relate to the user. There are many methods of defining this relationship but this document shall focus on use case creation and analysis. This provides the ability to map the Web site function and how the user type relates to it.

In most cases, user types will be fairly obvious as a project owner will know their audience fairly well. However, in some cases more analysis will need to be done. It is also difficult to do this without making fairly broad generalizations about certain user groups. For example, if the site is targeted at people over the age of 65, then the designer should assume that the users might have traits common to that age group such as poor eyesight or decreased mobility and should write the requirements to cater to this user group. This is not to say that everyone in this group has these particular challenges, but there is a greater possibility that this is the case. For an external Web site, the user base will tend to be fairly broad as you can never tell who will be accessing the site. It is usually important to ensure that no user feels left out by the system. Certain levels of competence and ability can be assumed if the project owner determines these assumptions can be made.

**Functional / Non-Functional requirements**

Requirements can be split into various sections and priority levels. The most commonly referred to sections are functional and non-functional requirements and the main priority levels are must, should and may. Functional requirements define behavior that the Web site implements such as adding a product to a shopping cart or logging in. A non-functional requirement is a factor that the system has as operational criteria, such as a list of browsers the system will support.

The priority levels of the requirements of a system refer to the importance of that requirement with regard to delivery. ‘Must’ refers to a requirement that the system has to implement, that is, a core part of the system and therefore has to be included in the first phase of the project. ‘Should’ refers to a requirement that the system ought to meet. It will refer to criteria that the system will be able to do without if there is a valid reason not to have it. ‘May’ refers to requirements that are not important to the running of the site but will add additional features to the system.

At the completion point of a system, all of the ‘must’ requirements should be implemented and all the ‘should’ requirements are either met or omitted, provided the omission is justified. The ‘may’ requirements should not prevent the system from going live. However, any of these requirements have to be either not started or complete, as an unfinished piece of functionality should be seen as a bug by the testing team.

**Content audit**

A content audit requires a sample of all types of content and what document or media types need to be displayed. Creating the sample cases, should not take a lot of time as it should primarily encompass picking a sample of content that fits into a particular content type and adding that to the audit sample. For example, all press releases tend to look the same as each other so analyzing them all would be a little redundant.

Once the content sample has been defined, content analysis can be performed to identify the patterns and the links between the sections of content. This provides the definition of the sections of the site that can later be drawn into the hierarchical site framework or taxonomy and include:

- Types of content
- Audience
- Dynamic content vs. Static content
- Connections between content
- Hierarchical structure
- Dependencies

By visually mapping the content with a content map the structure of the site can be marked out in a clear way. It will outline the structure of the site with the project owner in a clear and easy to understand manner, and it will ensure that the content is structured in a scalable and understandable way.

In later stages of project development, a content inventory will be required, but in the planning phase of a project a content audit is necessary. The difference is that where a content inventory is a definitive list of everything on the system, a content audit is a sample of that information.
By the end of this process and by organizing the content in this fashion, the site framework should be confirmed and the outline for creating visualizations of the system should be in place. This makes defining the content a critical point in the development of the system.

**Visualizations**
A key piece of the information architecture include the visualizations of the various pages.

**Wireframes**
A wireframe can be drawn to add definition to the content on a site and to help define how navigation works on the site. Wireframes are a high level tool that allows the architect to design how the content is laid out on the screen without having to worry about the look and feel. This will create a low fidelity version of the site that focuses on the usability of the site and not its presentation. A high fidelity version of the site can be drawn later in the design process, but by creating the wireframe of the site early on, the outline of the site is defined ensuring that the project owner is brought into the design process and can start to picture how the site fits together.

**Storyboards**
A storyboard defines how the user navigates through the system. It also allows the project owner to visualize the way that the user will view the use sequence of a site before the design phase is started. Storyboards tend to model the use cases, as they are event driven and from the point of view of the site user.

**Use Cases**
After defining the different user types that the system will cater to, and the requirements of the site are defined, use cases can be developed. The user types can be defined as actors in the system and the set of steps the actor takes while navigating through the system will become the events in the use case. This provides a testable and measurable plan for the development team. This definition of functionality will also help in the planning stage of the system to identify the dependences between the requirements, and will also highlight the areas of the system that might cause the development team problems.

**Web Content Management deliverables**
At the end of the Information Architecture stage the development team will then be able to draw up some documentation that will allow the development team to start developing. These deliverables give a clear picture of how a developer should implement the system, and also how all the information hangs together. It therefore provides a strong content centric design of the system. They include:

- **Site framework**
  Similar to a site map that would be found in the development of a traditional web site but rather than setting out a series of pages in a tree structure, it defines a set of site areas and content items. This effectively maps out how the user navigates round the site.

- **Libraries**
  A library is a container for any WCM objects that you wish to group together, although it is not necessarily important exactly what the libraries contain it is important that what they contain is logical. For example an Authoring Template can be referenced or copied between 2 different libraries but it may be logical to keep it within a specific container.

- **Authoring Templates**
  An authoring template specifies the input fields, for example text or images, that will be presented to an author when creating a content item.

- **Workflows**
  This defines the process by which a content item is verified before it is published.

- **Security requirements**
  Includes the access rights of the different user types within the system.

- **Category hierarchy (taxonomy)**
  It is important to note the hierarchical categorization of the content on the site. It differs from the site framework in that the site framework refers to the areas of the site but the taxonomy is independent to the location of the content.

**WCM Web site decomposition**
After completing the visualizations and assuming that the project owner has signed off or has provided some design elements such as prototypes, screenshots or existing sites to recreate, it is important to define what components are going to make up the site. The functional parts of the design that are provided can be
broken up and divided into the constituent parts. Then the pages that these components are shown on are identified. That is, if a component is located on every page then it should be identified that it can be re-used. A WCM specialist, while performing this analysis, will be able to identify components that may cause problems and highlight them as areas to address.

**User interface and user experience design**

There are certain web standards relevant to the development of an external site that a system should support, such as having a search function or navigational features like breadcrumbs. These are important points to note while defining the scope of a system as they declare what a system will and will not do to support its users. A well designed user interface (UI) that puts the emphasis on user experience ensures that the user will have a positive experience. This is a key point in supporting customer care which is often lacking on-line, particularly on e-Commerce Web sites, but is good practice with all websites. Good customer service dictates that if a customer has a positive experience they will tell a friend but if they have a negative customer experience they will tell ten, and this logic is no different on-line.

**Usability**

A key item for successful Web sites is usability. The following sections describe some key aspects to usability.

**Identifying the different user types**

To define the demographic there are 3 points that need addressed

- Who is using the system?
- What functions do they require?
- What challenges does this group face?

This gives the design team a clear picture of what issues have to be addressed by the user interface and what limitations have to be imposed on the site. For example, when designing a technical blog about programming, the site usually will not have to cater to people with little technical ability and this can be defined as an assumption about the users. On the other hand when creating an e-Commerce Web site it would have to assume less technical ability as anyone might be interested in buying a product.

**80 20 rule**

When laying out a site, a designer should take into consideration what features a user will require. When it comes to UI design in most cases, 20% of the functionality of the Web site is used 80% of the time, this is known as the 80/20 rule. It is therefore important to identify the aspects of the site that users use most often and give prominence to those features, and to also try to keep them together in strong positions on the page. This gives the user key areas of the screen that they know they can interact with.

**Function and appearance**

Items on the page must look and behave in a way that is obvious and it must be really clear what their function is. Using icons is a common way of defining what function the buttons have, and there is often an obvious choice for what icon to use. A save button for example will almost always have a disk or a file folder as the icon accompanying it.

**User centric design**

It is important to a user that they can quickly find the function that they are looking for, so they don’t have to think about what they are trying to do. Jakob Nielsen states “If your users have many questions, it’s a failure of your primary site design. It becomes not so much customer support, as much as customer complaints”. This illustrates that if a UI does not do the work for the customer then the customer will place the blame with the UI. This becomes a major factor in the success of a site, especially considering that it is much more beneficial for a business to keep an existing customer, than find a new one.

**Content**
The quality of the content on a Web site goes beyond merely making the site interesting. This is of course an important factor, but the site must go further. Targeted content is an important method of pulling users into the site, keeping users on the site and guiding the user to where they want to go. Different sites will require focus on different sections. Unfortunately this is very much dependent on what the function of the web site is.

- The inclusion of keywords will improve search engine ranking
- Readable content and accurate information delivered in a timely and efficient manner will assist user decisions
- Interesting copy will retain users on the site

At times, these factors will conflict with each other. For example a passage of text that is overloaded with keywords will tend to be unreadable. To ensure a positive user experience it is imperative that the content makes sense and is relevant to the user. Appropriate keywords facilitate the external search engines to serve the correct page to the user. In most cases succinct content that is relevant to the subject matter will contain a lot of the keywords and phrases that the end user will search for anyway.

**Look and feel**

To create a positive user experience, the UI should have a strong design, not only from a functional perspective but a graphical perspective. If a design looks untidy or the layout is confusing it will be hard to navigate through, creating a negative experience for the user. The site must therefore concentrate on providing the function that the end user was originally looking for in an attractive and easy to use way. The main consideration for the look and feel of a site should lie with the function of the site. For example, a corporate Web site should look professional and clean where as in the case of an e-Commerce site the focus should be on show-casing the products that it offers.

**Clean design**

A UI should be simple and easy to navigate and a clean design helps to facilitate this as it tends to avoid clutter and superfluous images or text.

Some simple guidelines include:
1. Avoid clutter
By overburdening the site with excess images and over designing elements, the Web site will start
to look cluttered. It is tempting to keep adding things to attract users to different areas of the site.
However, by adding too much, the flow through the site can be disrupted. This can be addressed at
a very early stage in the development process by avoiding the shopping cart effect when
requirements gathering, that is when gathering requirements the project owner is likely to want to
keep adding things to the Web site until there is a saturation point of what is physically possible to
do.

2. Pick a simple color scheme
A color scheme in which each color complements the others should be defined. A strong color
scheme provides definition for the site and ensures that the colors used by the site remain
consistent. It can also differentiate each level of the site design from background down to the color
of the text. Use of color can be very important in guiding the user around the site as it creates a
visual definition between different sections. It should be noted that some colors have meanings for
users. A common mistake in design is to try to attract user’s attention by using red text. This often
collides with the color scheme of the site, is often lower in contrast to its background, and also
tends to have negative connotations for a user. Use of bold text or text with a border has a much
greater impact for the user.

3. Ensure all text is readable
Text should always be high contrast from its background ensuring that it can be read. Text that is
unreadable will tend to get ignored or passed over by the user and does not have the same impact
as text that clearly stands out from its background. Also it is important to ensure that heading text
can be clearly associated with any passage of text it is linked to. the following figure shows two
poor examples of the use of contrast and one good example.

4. Use subtle and consistent effects
Many web sites use gradients or other effects such as drop shadows to provide emphasis and
depth to different sections of the site. Many of these effects suggest a light source somewhere on
the page and often a designer will have two elements in a design that suggest that the light source
is in different places. It is important to ensure that these effects do not overpower what they are
displaying or compete for the user’s attention.

5. Use a tried and tested layout
When laying out the Web site the designer generally should avoid re-inventing the wheel. Most
sites particularly corporate sites should tend to follow existing standards. By using a layout the user
is familiar with, they can easier understand where things are. A designer can follow a different
layout and it is important to note that this point is a judgment call. A bad design that conforms to a
standard provides a worse user experience than a good design that does not conform to standards.
6. Put important information in the strongest part of the design
The top left of a website is usually the strongest part of a design. This is likely to be the location of the corporate logo or the primary navigation on the page. It is also the location of most of the common navigation functions on a browser such as back or forward, and on the menu bar with the most used functions such as file, edit and so on.

The KISS principle
The KISS (Keep it simple, stupid) principle is an important methodology for UI design. The idea is to minimize complexity and make things as intuitive as possible by, as the name suggests, keeping it simple. It is important to constantly put yourself in position of the end user. When developing the interface, concentrate on providing clear visual cues to the user such as icons or labels and avoid giving the user too many options.

Fixed versus fluid
Navigation

Navigation through the site should be obvious. It should be located on the top or on the left side of the screen and should be consistent throughout. Dropdowns should be close to the top level item and navigation should behave in a standard and expected way.

Three clicks rule

The three clicks rule recommends that the user should have to use a maximum of three mouse clicks to get to the content they would like. However, as there is no evidence to support this rule, it is widely suggested that the user will be better served by ensuring that the content that they would like to find is in the most obvious place regardless of how many clicks they took to get there. The key is to balance the two schools of thinking and not to get too hung up on ensuring that the user clicks no more than three times. Well designed navigation and a clear easy to follow site map will provide much greater value for the user, than adding additional complexity to the user to shoe-horn the site structure into a site map that only has three clicks.

Web 2.0

When looking at the human-computer interaction (HCI) of a Web site, Web 2.0 techniques can ensure that the site is easy to use and gives information to the user in a clear way, as well as providing a mechanism for refreshing the parts of the screen the user is interested in more quickly. However the use of Web 2.0 techniques can prove to be a double edged sword, as over use or inappropriate use can lead to confusion or annoyance for the user. For example, if an animation on a pop-up is too slow, the user can lose patience with the system, or if the information they want to get to is hidden, then they will become frustrated at not being able to find it.

The best practices here are to:

1. Decide what library to use
2. Identify the sections of the site that will benefit most from using a Web 2.0 function
3. Deliver the key information that the user is interested in without requiring user action
4. Where possible, give the user a summary that can be expanded to more information
5. Don’t over use a function
6. Ensure that time based effects, such as animations, are kept as short as possible so as to not withhold information from the user
7. Always provide a no-script alternative
8. Keep effects consistent throughout the site

For more information on Web 2.0 and WebSphere Portal, see the Web 2.0 page in this wiki.

**Accessibility**

The Web Accessibility Initiative (WAI) defines accessibility this way. "Web accessibility means that people with disabilities can use the Web. More specifically, Web accessibility means that people with disabilities can perceive, understand, navigate, and interact with the Web, and that they can contribute to the Web. Web accessibility also benefits others, including older people with changing abilities due to aging."

See [http://www.w3.org/WAI/intro/accessibility](http://www.w3.org/WAI/intro/accessibility)

As with Web site usability, accessibility is dependent on the demographic of users of the site. There is some overlap between usability and accessibility in that a user centered design will adhere to the standards set out in these disciplines. However, accessibility does cover some topics that usability does not cover. The key to what these are lies within the ability of the user groups the site is to be made accessible to.

Most sites will require basic accessibility features such as cross browser compatibility. Whereas other sections will require a more detailed approach implementing such features as variable text size, mobile support or screen reader compatibility. It is important to the success of the site to define what these requirements are at the start of the process.

**Accessibility in WebSphere Portal**

Accessibility is important for everyone. Web sites are primarily about communicating information with the widest possible audience, whether it is a news site, a commerce site or a technology organization like IBM; it is about information dissemination and communication. Making a site accessible is about removing the barriers that limit the groups of people that can receive the information the site provides.

The Lotus Web Content Management (WCM) system provides the development team with a platform and components to build an accessible Web site. Using the team’s existing skills and tooling the IBM WCM platform provides you with a foundation and management system to facilitate accessible web design and development.

**WAI**

The Web Accessibility Initiative (WAI) of the World Wide Web Consortium (W3C) is focused upon addressing the needs for web accessibility. WAI provides a repository of the latest web accessibility guidelines and tools.

See [http://www.w3.org/WAI/](http://www.w3.org/WAI/)

**Web content accessibility guidelines**

The WCAG guidelines that have been set out by W3C are primarily intended for four main groups:

- Content developers (content contributors, presentation designers and developers)
- Web authoring tool developers
- Web accessibility evaluation tool developers
- Others who want or need a technical standard for Web accessibility

The WCAG documentation provides guidelines as to how to make web content, such as text, images, forms or sounds, accessible to people with disabilities. WCAG is part of a series of accessibility guidelines, including the Authoring Tool Accessibility Guidelines (ATAG) and the User Agent Accessibility Guidelines (UAAG).

See [http://www.w3.org/TR/WCAG/](http://www.w3.org/TR/WCAG/)

**Standards W3C WAI and section 508**

Making websites accessible is only really possible by adhering to the standards laid out in the WAI (Web Accessibility Initiative) & Section 508 guideline documentation. In addition, it is important to adhere to any regulations that are specific to the country in which the site is being deployed and need to be understood.
and applied appropriately. In the USA, Section 508 covers the legal requirements for web sites.

http://www.section508.gov/index.cfm?FuseAction=Content&ID=3
http://www.w3.org/Consortium/

**Browser compatibility**

There are currently more major browsers on the market than ever before, and ensuring that a site looks the same on all browsers has never been so important. If a site fails to render properly or fails to work properly then it can frustrate users, and may lose their interest entirely. Although most browsers should display standards compliant code in the same way, there is always a chance that a site will look different in each browser.

As new CSS attributes and HTML standards are released there are older tags and attributes that are no longer supported by the browsers. HTML standards are becoming more future proof. For example, XHTML is trying the take all formatting out of the HTML document. So future proofing and keeping the markup current is very important to ensuring that the site renders successfully.

The best way to ensure that there is a high degree of cross browser compatibility is to test the site in all these browsers. There are some things that the development team can predict and make efforts to resolve, however visually confirming that the site looks proper on all browsers is the only real way to be sure.

**Define your support list**

In the early stages of compatibility testing it is important to define the browsers that you wish to support. As new technologies are released and browsers become more advanced inevitably older browsers will become deprecated. There will also be a noticeable drop in the browsers market share as users upgrade to the latest versions of their preferred browser. However this drop off in usage will tend to happen slowly rather than overnight and this can mean that requirements to support older browsers are necessary. It poses the question; to what level should older browsers be supported within the system?

There are three levels of support the development team should consider:

1. Does the system look and function identically in each of the browsers this system is designed to support?
2. Does the system look and function to a high standard but not look identical in each of the browsers this system is designed to support?
3. Is the system navigable in each of the browsers this system is designed to support?

Obviously the more browsers a developer is required to support the longer it will take to develop and test. Also with older browsers there may be aspects of the design that are not possible with the tools supported by the browser. It may be prudent in this case to just ensure that on older browsers the site looks presentable and easily navigable.

At the very lowest level of cross browser compatibility the user should be able to find their way round a site in the same way in each of the browsers on the support list. The look and feel may vary slightly on each browser but this is a secondary to functionality.

When defining the list of supported browsers the development team should first list all the major browsers and versions of those browsers then agree with the project owner how rigorously the browsers should be supported. This information should be fed to the designers of the user interface to ensure that there is nothing in the design that cannot be produced in a supported browser.
Content centric design
This page covers content centric design and the Web Content Management component of WebSphere Portal.

Understanding Web content management
In today's global business world, businesses depend upon visibility in their respective marketplaces. With the pervasiveness of the Internet, the Web has become a prime delivery mechanism of this visibility. Worldwide, organizations use the Internet to deliver their presence, from basic organization and company information to e-Commerce. The Web is a sophisticated and key tool.

A Web site is a collection of all of the text, graphic images, links, sounds, and other content elements that make up a presence on the web. Ultimately, the goal is to present content and information in the most dynamic, up-to-date method possible. Each individual document is called a web page. Web sites have three basic components:

- A home page - The home page is the top-most page in the Web site.
- Local and remotely stored web pages linked to the home page.
- Content - Content includes text, graphic images, and sounds.

Content and its management
Content can mean many different things. The most common understanding of business content can be summarized as:

- Content supports the work that the enterprise does and interacts in many ways with business operations.
- Enterprise content management is the discipline involved with the capture, storage, and management of this kind of content across the enterprise.
- Content management makes information easy to find, use, update, and discard when the time comes.

Content can be categorized into the following types among others:

- High-volume production content, such as document imaging and computer output, archiving, and presentation
- Rich media, such as audio, video, and photos
- Web content, such as Internet, intranet, and extranet
- Collaborative content, such as office documents, discussions, and e-mail

Enterprise content management has the following objectives:

- Provide an efficient and secure solution for managing content within the enterprise, including enterprise-wide content storage, access, search, distribution, and retention. This is especially true with the current focus on corporate accountability and regulatory compliance.
- Build knowledge-based environments to leverage corporate know-how and expertise.

Web content types
When your organization offers information via a Web browser, this material is considered Web content. Web content consists of two parts:

- The content
- The design or presentation of that content

There are two types of content as related to creating, managing, and publishing Web content:
• Static Web content
• Dynamic Web content

**Static Web content**
Static Web content is embedded directly into the design and placed statically on a Web page. This type of content is most likely the result of programming rather than content authoring. Due to its static nature and embedded design, this type of content can be difficult to reuse and change.

The content of static Web sites is difficult to alter as the bulk of the content such as text, graphic, and multimedia content are usually stored in HTML pages and not in databases. Changing the content on multiple Web pages can be (and usually is) tedious and time-consuming. There is also the risk of file overwriting critical content and the possibility of important non-content areas such as the security code of the site getting corrupted.

The basic disadvantage of static Web sites is that the web pages are developed at the time of constructing the site. As such, all static Web site owners have to be abjectly dependent on Web professionals whenever any changes become necessary. Though the initial cost of setting up a static Web site may not be costly, its long term maintenance will be expensive, especially if the site owner wishes to effect changes at regular intervals.

Static Web content solutions often require someone within an IT department, or at least a person who possesses Web design and some Web-based IT skills, to translate content into HTML before it can be posted to a Web site or a portal. Accordingly, static Web content is more difficult to use within a dynamic Web site, which changes frequently and requires up-to-date information to deliver maximum value. Additionally, static Web content is often impossible to reuse due to its embedded design.

**Dynamic Web content**
 Rather than embedding the information directly together with the design, dynamic Web content treats the content and the presentation of the content as two distinct entities. By separating the Web content from the presentation layer, you can manage and deliver content quickly and efficiently, independent of its presentation. The ability to manage both content and its presentation layer as separate entities is enabled most frequently by using Web content management systems.

The other advantage of a dynamic-content Web site is not having to depend on Web professionals for making alterations to the Website. A dynamic-content Web site, though developed by Web professionals, can be maintained directly by the site owner. Unlike static Web sites, the initial set up costs of a dynamic-content Web sites is quite high but there will be enormous saving in subsequent maintenance costs.

The owners will not be required to pay Web professionals every time a change in the Web content is needed. A dynamic Web site design will become necessary for online databases, e-businesses and online shopping sites. A dynamic Web design page has the facility to change its content via a program.

**Web content management**
To thrive in an on-demand environment, you need flexible, cost-effective content management solutions to manage any type of information, including structured data and unstructured content. You need content management solutions that enable data and content to be integrated with multiple applications and processes, distributed or shared throughout and beyond your organization. Furthermore, you need it structured and organized in the way that is best suited for your business.

**Web content management fundamentals**
In order to understand Web content management we must first distinguish between content and the presentation of the content. When a clear separation is made between content and how it is presented, you can appreciate how a single piece of content can potentially be rendered in multiple ways. For example, a single item of content, such as a news article, can be presented in any of the following ways:

• On a Web page in a variety of formats, based on user preferences and personalization
• In a PDF file
• Downloaded to a mobile device
• Streamed as an RSS feed

The content is the same, but the presentation can be adapted to best meet a user’s needs within the context of their specific role or preference. This approach also guarantees a consistent view and usage of a Web site. If the design changes, the content parts remain unaffected. Additionally, the content creators do not have to worry about the presentation of their content. This is important because content creators usually do not have significant knowledge of HTML.

Moving forward with this idea, key concepts and functional areas that apply to web content management systems can be grouped in terms of dynamic presentation or content management.

**Content publication**
After content is authored and approved, the content publication stage is when the content is released for delivery to the live site. The delivery can be a simple process, such as making a file available on a file system to a Web server and advertising the URL. Or it can be a more complex procedure such as moving content through a complex workflow and into a production environment.

**Workflow**
The accuracy and relevancy of content is assured by automating the content life cycle, from creation through approval and delivery to review and archival.

**Content aggregation**
Content aggregation occurs when content from various sources is brought together. In a simple Web site, this occurs manually during the content creation phase. However, in a more dynamic and data-driven environment, the aggregation occurs at an application server level.

**Content delivery**
Content must be delivered to the user in the appropriate (or desired) format. Most frequently, this implies delivery via HTTP by using browsers and similar devices. Alternatively, content delivery can refer to other publish-and-subscribe methods, data feeds, or Web service protocols.

**Integration**
Aggregation, transactional integration, and performance enhancing caching enrich the user experience.

**Content creation and authoring**
Content creation and authoring refer to the creation of content and information that is intended to be delivered. Content creators are guided through the authoring process, by using familiar applications, without having to learn new technical skills.

**The need for Web content management**
It is quite a standard practice for most organizations to use a Web content management system to assist with the process of delivering content to their intranet or Internet site. For example, a product page that details the features and advantages of a new coffee maker. The generation of the words and images can involve several people, including subject matter experts, image designers, proof readers, legal, and IT. If we achieve this all through existing desktop office tools, it can be quite arduous and time consuming. However, by using a modern Web content management system, the process can be streamlined to allow distributed authoring, work flow, preview and ultimately delivery of the finished article to the Internet Web site, notifying the appropriate people. For both intranet and Internet sites, content is king. Without it being current, timely, and appropriately placed, the site can be perceived in a lesser light. From the content delivery process alone, a Web content management system seems like a good choice.

With the content creation, approval and delivery process providing the top capability and benefits from a Web content management system, are there other areas that can lend weight to using a Web content management system? The answer is "Yes". Ensuring that the Web site has a consistent look and feel (branding) is extremely important for the external Internet sites. It is important for a businesses' brand awareness and for ensuring that visitors to the Web site are comfortable with the layout, presentation, and navigation to content. A web content management system provides a structured approach and framework to allow the separation of content and design while ensuring consistency where required. Componentization is also an area that the Web content management framework affords you, by allowing a higher degree of reuse, ultimately building more with less. Again this is another clear area where a Web content management system seems like a good choice.
Content and design are just two important aspects of any Web site (intranet or Internet) that benefit from a WCM system. In addition, the following aspects are important to a Web site:

- **Content**: The creation, approval, and delivery process streamlined to take advantage of your organization's distribution of knowledge and skills.
- **Design**: Separation of design from content to allow for the design process (in-house or agency) to deliver consistent branding of your business, its products, and services.
- **Componentization**: Reuse of assets to allow the business to deliver more with less, and saving money.
- **Standards**: A system to employ best practices where accessibility is required.
- **Integration and delivery**: Providing a framework from which a business can integrate with systems, services, and information including growth from Web sites to portals and beyond.

With these concepts in mind, you can be reassured that you are making the right decision to use a web content management system. While there might be many factors to consider, the following decision tree can guide you to some that might help or provide a starting point to expand upon.
Content authors (people with the knowledge you need to consume) will be distributed around the business.

Yes

The business does not expect the content authors to be conversant with web development or HTML basics.

No

Yes

The business does not expect the content authors to be conversant with web development or HTML basics.

No

Yes

The site needs to be frequently updated, expectation is daily.

No

Yes

The site contains more than 20 pages of information?

No

Yes

The business will be looking to integrate content with applications, ala Portal.

No

Yes

You expect to deliver the content in multiple formats? RSS, XML, etc.

No

Yes

A full WCM system is probably appropriate for your requirements.

No

A STATIC traditional website and tools are probably sufficient for your requirements.

Other features that might drive the need for a WCM system:
- Forums
- Commenting / Rating
- Blogs
- Content Tagging
- Video / Audio streaming
- High degree of organisational change / acquisitions
- Compliance
**Defining a content centric site**

Content-centric applications, currently being adopted by companies, take content management onto the next level. Content combined with business logic is used to influence the behavior of an organization’s key stakeholders - customers, partners, and employees - by persuading them to take actions that best support business objectives. The content displayed in such applications is typically relevant, personalized, and context-aware.

Take the example of a typical corporate intranet, the information is pushed out and presented to users in a completely static form for them to consume. A Web 2.0 intranet, on the other hand, will be dynamic and augmented or even populated by user-generated content and with the flexibility that will enable users to assemble the content in whichever way they want.

Wikis, mash-ups, user-tagging to create folksonomy: social networks are moving interaction far beyond the traditions of business applications. And it is not just the jargon that has evolved: dynamic web development techniques, such as Ajax, are becoming widely adopted; XML is becoming ubiquitous and the whole structure of application delivery is changing.

A content-centric organisation had pushed the use of wikis and blogs to foster improvements in team working. That has had a transformative effect in the way the organisation should think about disseminating information and the way content is delivered for the next stage.

Such work has now filtered through to the way the organisation thinks about all information classes, and how it can squeeze every last drop of knowledge from the vast quantities of operational data it has accumulated.

**Introduction to IBM Lotus Web Content Management**

In today's global business world, businesses depend upon their visibility in their respective marketplaces. With the pervasiveness of the Internet, the Web has become a prime delivery mechanism of this visibility. Worldwide, organizations use the Internet to deliver their presence, from basic organization and company information to e-commerce. The Web is a sophisticated and key tool.

IBM Lotus Web Content Management provides a sophisticated Web content management tool and platform that is designed to accelerate the delivery and management of critical business information. It enables a collaborative approach to content creation for individuals and teams. It allows for approval of processes, management, and assets. Lotus Web Content Management is an extremely powerful and sophisticated tool that is remarkably easy to use.

**Lotus Web Content Management new features and improvements**

IBM Lotus Web Content Management v6.0 was released and generally announced in June of 2006. With the introduction of v6.0, Lotus Web Content Management received many new and significant enhancements that added to its capability as an enterprise content management system (CMS) and introduced further levels of scalability and reliability.

Today, Lotus Web Content Management v6.1.5 has built upon these areas further to deliver an enterprise scalable CMS capable of meeting both business and technical requirements. Version 6.1.5 has introduced new product enhancements in many significant areas across the platform including administration, authoring and management of content, rendering, security, workflow, APIs and many others. The following information explores key new additions to the platform and the benefits gained through their use and implementation.

IBM Lotus Web Content Management v6.1.5 includes new features, such as a new rich text editor and Web content viewers, as well as improvements, such as large file handling.

**New Web content viewers**

Lotus Web Content Management includes a new JSR 286 Web content viewer that integrates the portal's page structure with information from a Web content library. As a result the portal page hierarchy helps to organize the navigation of your content. In addition to leveraging the personalization, presentation, and security benefits afforded by JSR 286, there are additional advantages:
• Web Content pages allow you to attach content directly to a page, so you no longer have to configure a rendering portlet to display content from your content library. Other benefits of the JSR 286 Web content viewer include:
  • Sessions are not required for rendering Web content for both anonymous and authenticated portal users.
  • Performance is improved over the traditional Web content viewer.

The JSR 286 Web content viewer can be configured to create site analysis log entries, enabling you to gather data about how often content is visited.

Portal Search supports the use of seedlists to make crawling Web sites and their metadata more efficient and to provide content owners fine-grained control over how content and metadata are crawled. Support has been added for the new seedlist 1.0 format. With support for seedlist format 1.0 and the JSR 286 Web content viewer, search result links going to content items now display the content items in the context of the portal page where they are rendered. Seedlist format 1.0 is the out-of-the-box seedlist format, however you can configure the portal to use a previous seedlist format if required.

**JSR 286 Web content viewer**

Based on the Java Portlet Specification 2.0 (JSR 286), the JSR 286 Web content viewer is a full-featured viewer that integrates the portal's page structure with information from a Web content library. In addition to leveraging the personalization, presentation, and security benefits afforded by JSR 286, the JSR 286 Web content viewer provides additional advantages:

  • Web content pages are supported. Because you attach content to a Web content page when you create it, the JSR 286 Web content viewer automatically displays that content without requiring configuration when you add the Web content viewer to the page.
  • The portlet does not create a portlet session for rendering Web content for both anonymous and authenticated portal users.
  • Support for portal site analysis is available to measure how frequently content items are visited.
  • Performance is improved over the traditional Web content viewer.

**New rich text editor**

EditLive! is the market leading rich text editor providing rich desktop/office tool capabilities like advanced table editing as well as linking to Web Content Management style sheets, role-based control over the RTE features, accessibility support and more.

**New link resources**

A new link type is now included to allow you to create links to documents stored in an external ECM or document repository, such as Lotus Quickr. When creating a link to an external document in an HTML or rich text element, you are able to select documents to link to from a pre-defined list of external document repositories.

**New syndication feature**

A new tool view been added to the syndication administration to allow you to monitor pending items on a syndicator and manually syndicate those items.

**Large file handling**

Web Content Management now supports documents and other binary files that are greater than 50Mb. Of course, good content design would say you don't want to have a lot of really large content objects that need to be downloaded to your site visitors' browsers.

**Web content feed integration**

The IBM Web Content Integrator is a solution used for integrating externally managed Web content with Web Content Management. Through the use of standard content syndication feed technologies based on RSS 2.0, the Web Content Integrator provides a loosely-coupled mechanism for transferring published content and meta data to the portal after they have been approved in the source system. Once the content and meta data have been transferred to the portal, it is possible to leverage the built-in content management features of Web Content Management to secure, personalize, and display the content to end users.
**Working with IBM Web Content Integrator**
If your file resource element is a file type that can be converted to HTML you can now convert the file to HTML and render the converted HTML directly in your Web content. Examples of supported file types include:

- Word processing documents (*.doc, *.odt)
- Spreadsheets (*.xls)
- HTML files (*.htm, *.html)
- Text files (*.txt)

**New file upload validation plugin**
A new plugin class has been added to allow you to write file validation plug-ins that can be used to validate files uploaded into file resource, image and style sheet elements, and images uploaded into rich text or HTML elements.

**IBM Lotus Web Content Management Architecture**
The Lotus Web Content Management 6.1.5 application is fully integrated with WebSphere Portal 6.1. As such, all required components of the Lotus Web Content Management application are installed with WebSphere Portal 6.1.5. The following figure provides a high level overview of where Lotus Web Content Management fits into the WebSphere Portal infrastructure.

**Architectural frameworks for content in WCM**
The following sections briefly describe various frameworks supported by WCM in WebSphere Portal.

**Portal**
Portal provides the basic framework services for content aggregation, role-based access, personalization and security. The underlying J2EE platform provides low-level middleware services such as enabling security with a user registry and session management. These underlying services coupled with the basic services supported by the portal framework can be effectively used to provide content life cycle management in terms of content authoring and content delivery. Different roles, such as author, reviewer, and approver, can participate in their capacity to move the content from authoring to delivery. The content management and delivery framework embedded into Portal can be used in tandem with the other features,
such as personalization and content aggregation, which provides an effective layer for presentation services in the architecture. The information architecture can be a combination of the content management and delivery interspersed with other portal navigable pages to render content from different back-end systems.

**Traditional**
Traditionally, content is prepared in pure HTML format by using different HTML editors, and the content is hosted on a Web server for the presentation. The URLs are exchanged with the reviewers and approvers for verification and validation over e-mail. The hosting platform is lean and provides no functionality in terms of content management. Dynamic applications came into existence with CGI scripts but had lots of limitations. The evolution of the J2EE platform has brought in a new dimension to display content on Web sites. Static Web content is coupled with the dynamic content generated through servlets and JavaServer Pages (JSPs). One aspect that cannot effectively be provided through this framework is content management.

**Internal facing**
Internal facing sites are those meant for the employees of an organization. Apart from being simple but easily accessible (through effective navigation), there must be an important application for the employees to visit the site. Typical entities on such an intranet are news, alerts, and role-specific content. Apart from the content, in terms of production, such sites benefit immensely if the intranet users are provided access to applications that are useful for day-to-day activities, such as sales management, defect tracking, customer relationship management, or technical support. The challenge is to intersperse the content by using a content management framework, such as Lotus Web Content Management, with Portal (pages and portlets).

**External facing**
On external facing sites, the content must be updated because the public view decides the image of the organization. The taxonomy of the site should be well defined, and the content should be easily accessible either through browsing or search. The site needs to be fast enough and within the limits of the users’ response time perceptions. Any response times more than 5 seconds are perceived to be slow by the users. The framework should support caching techniques at various levels to enhance the response times for different scenarios. The user can have a unique ID with the site, and the content can be personalized according to the profile of each of these users. The architectural and design frameworks should consider how these content management, search, caching, and personalization services can be provided to the user, with the goal of attracting and keeping the attention of the users.

**Information architecture and site design**
When designing a Web site, organize information in a way that is readily accessible to users. The structure in which information is organized is called the information architecture. In this section, we provide an approach for determining the information architecture for the top levels of your Web content manager (WCM) Web site. You structure the information that is published on a Web site within a site framework that is accessed through Web site navigation. In this section, we lead you through the decision making process by discussing and defining the information architecture, key considerations, decision processes, and information design. Additionally, we explain how to define criteria for site acceptance.

It is important to acknowledge the relationship between the information architecture and the site framework. The site framework structures the information about a published site and is closely integrated with the Web site navigation. To begin, you define an initial site framework with primary and secondary site areas. The site framework is a prerequisite for the design and development of the content management system. In addition, you must develop an initial layout of the home page based on the primary and secondary site areas. Both the framework and the home page layout require review and validation with business stakeholders, from both an authoring and Web usability viewpoint.

**Defining the information architecture**
Content accessibility on an intranet can have a direct impact on an organization's overall productivity. If users can find the information they are looking for before they even are aware that they are searching, then you have accomplished the goal of organizing the information correctly. The primary reason for sharing information on a Web site is to make it available for users. However, it is often the central reason why Web sites fail, which include for the following reasons:

- Information is often voluminous and widespread over different divisions within an organization, making it difficult to maintain a consistent oversight of the structure of information and to determine a structure that is easy to understand and to use.
• There is rarely a single person or department with a consistent oversight of all the information that an organization wants to share.
• Information is used everyday, and as a result, it is often not obvious which pieces of information are most valuable or how they should be structured.
• By using a content management system, content providers can share more information and gain more experience, which can unfortunately result in a strain on the primary organization of the content.

The information architecture defines how the information on a Web site should be organized and linked so that users can access content. An organization should investigate, analyze, design, and implement the information architecture for a site. Then, it must face the challenge of presenting an image that enhances the way in which the user experiences the information.

You might find reasons why a specific path to information supports your business needs. The experience that a user gets can be the key to success and often reflects the organization’s philosophy. Frequently, an organization builds its success on a unique customer experience that cannot be ignored when architecting a Web site. In addition, the audience can vary such that it becomes necessary to separate users into categories to get a good acceptance of the way information is presented on the Web site. For example, consider the following diverse offerings:

• A food company might offer both traditional, home style food and more contemporary fast food.
• A vehicle manufacturing company might have diverse offerings that range from passenger cars, to industrial trucks, to motorcycles.
• A technology company might offer a range of products and services, from technical consulting services to consumer electronics.

Because of this diverse range of products, brands often get their own Web sites with there own information architecture and Web address (URL). In this case, a master (parent) Web site, which includes links to detached Web sites that represent the specific branding, is the best choice. Defining the information architecture is often the most underestimated part of a content management project. Organizations frequently spend a lot of time finding the right content management system and then deciding on the best information architecture for that system.

When planning the information architecture, an organization must determine the following information:

• The hierarchical structure of the site
• The functionality that is required on the site
• The look of individual pages
• How to classify the content
• The flexibility of the architecture to allow the business to evolve

The information architecture determines the structure of the site, how navigation is derived, and the ease of navigating the site. You develop the following information architectures when designing a content management system:

• Site framework
• Category hierarchy
• Document type hierarchy

Defining the information architecture lays much of the groundwork for how content is organized on a site. Regardless of where the content resides, you need a good understanding of the content that is to be displayed.

**Physical architecture best practices**
In this section, we look at physical architectures that you can use to build the Lotus Web Content Management infrastructure.
In real-world installations of Lotus Web Content Management, a variety of common infrastructure designs are in place. The primary differences in these designs are due to variations in several basic assumptions:

- **Site size and complexity:** For a relatively small Web site, it might not be necessary to have individual servers dedicated to all four types of Lotus Web Content Management environments.
- **Funding:** In many cases, there is a limitation in regard to the funding provided to build out the Lotus Web Content Management infrastructure. In these cases, you must construct your environment as soundly as possible within the budget. However, the budget is likely to force you to reduce the infrastructure.
- **Corporate standards:** Your company standard might not allow for the creation of certain types of servers. For example, it is quite common for smaller companies to omit the staging environment because it requires time and resources to perform the content review in this stage.

**Basic architecture**
The basic architecture (illustrated in the following figure) is the smallest acceptable design for an IBM Lotus Web Content Management infrastructure.

In the basic architecture, a single Lotus Web Content Management server is responsible for all Lotus Web Content Management activities. Site visitors access this single server for content when they visit the Web site.

An HTTP server is placed in the DMZ to receive all requests for site content. This HTTP server acts primarily as a proxy to shield the Lotus Web Content Management server from direct access. If you use IBM HTTP Server (or Apache), the WebSphere Portal and Lotus Web Content Management server can generate an HTTP plug-in to ease configuration of the HTTP server.

While this architecture is technically feasible, there are a variety of issues that make it sub-optimal.

**Design advantages**
Design advantages include simplicity. This design is simple to implement due to the limited number of components that are involved.

**Design disadvantages**
This architecture has the following design disadvantages:
- Failover or redundancy: This environment has no failover or redundancy in any layer. If any one part of the system fails, then the entire site appears to be down.
- Load balancing: This environment has no load balancing capabilities. There is only one Lotus Web Content Management server, and it must handle the entire workload at all times.
- Maintenance: To perform site maintenance, the site must be unavailable.

**Intermediate architecture**

The intermediate architecture (illustrated in the following figure) attempts to correct the primary run-time deficiencies of the basic architecture.

In the intermediate architecture, multiple Lotus Web Content Management servers share the responsibility for Lotus Web Content Management activities. Site visitors can access any of these servers for content when they visit the Web site.

Multiple HTTP servers are placed in the DMZ to receive requests for site content. These HTTP Servers perform two primary functions:

- **Proxy:** The HTTP servers act as a proxy to shield the Lotus Web Content Management servers from direct access. If you use IBM HTTP Server (or Apache), the WebSphere Portal and IBM Lotus Web Content Management server can generate an HTTP plug-in to ease configuration of the HTTP Server.
- **Load balancing:** The HTTP plug-in can be configured to provide load balancing based on a variety of algorithms. If your Lotus Web Content Management servers are not similar in processing capabilities, you can set up the plug-in to load balance traffic according to server capacity.

While this architecture improves upon the basic architecture, still several issues make it sub-optimal for larger environments.

**Design advantages**

The intermediate architecture has the following design advantages:
• Simplicity: This design is simple to implement due to the limited number of components that are involved.
• Load balancing and redundancy: If you are using IBM HTTP Server (or Apache), this environment has basic load balancing capabilities. Because there are duplicates of all components, a basic level of redundancy exists in this design.
• Maintenance: With multiple servers available to deliver content, site maintenance can be performed without the entire site becoming unavailable.

**Design disadvantages**
The intermediate architecture has the following design disadvantages:

• Administration: As more Lotus Web Content Management servers are added to the infrastructure, the maintenance effort to make changes increases in a linear fashion. Changes must be manually made to each server because the servers are not clustered in this design.
• Failover: While we have redundancy with this design, there is no automated failover in the event that a specific component fails. If any one part of the system fails, there is a likelihood that the site is down to some subset of the users.

**Advanced architecture**
The intermediate architecture still contains one major deficiency. That is, there is no failover within the environment. To resolve this issue, we look at building a slightly more complex infrastructure that includes clustering some of the servers.

When you cluster WebSphere Portal servers, they share an identical configuration. This configuration is managed from a WebSphere Deployment Manager server (not pictured). The Deployment Manager is responsible for synchronizing the WebSphere Portal configuration across all servers in the cluster. Thus, when you deploy a Lotus Web Content Management portlet or create a new portal page, the Deployment Manager synchronizes all servers in the cluster, which is a reduction in administrative time and effort.

In addition, when you cluster the WebSphere Portal servers, the Deployment Manager can create a plug-in for IBM HTTP Server and Apache that allows the HTTP servers to gracefully failover in the event that one of the portal servers stops functioning. While setup and maintenance of a cluster take time and effort, this effort
is typically small compared to the ongoing benefits provided to the infrastructure.

**Design advantages**
The WebSphere Portal and Lotus Web Content Management integrated architecture has the following design advantages:

- Load balancing and redundancy: If you are using IBM HTTP Server or Apache, this environment has basic load balancing capabilities. Because there are duplicates of all components, a basic level of redundancy exists in this design.
- Failover: With the addition of a cluster, the HTTP Server can provide failover for the WebSphere Portal servers. If any one part of the system fails, the entire site does not appear to be down.
- Maintenance: With multiple servers available to deliver content, site maintenance can be performed without the entire site becoming unavailable.
- Local rendering portlet: If IBM Lotus Web Content Management is running on all portal servers, you can use the Local Rendering Portlet. By not having to communicate with an external Lotus Web Content Management server (as required with the Remote Rendering Portlet), you eliminate a potential performance bottleneck.
- Administration: As more IBM Lotus Web Content Management servers are added to the infrastructure, the maintenance effort to maintain the infrastructure does not increase significantly. Changes to server configurations are performed from the central Deployment Manager console.

**Design disadvantages**
The WebSphere Portal and Lotus Web Content Management integrated architecture has the following design disadvantages:

- Complexity: This design is fairly complex. The learning curve for installing and administering a clustered environment servers should not be underestimated.
- Cost: Enabling Lotus Web Content Management on every WebSphere Portal server quickly increases your infrastructure cost.

**WebSphere Portal and IBM Lotus Web Content Management separated**
In the WebSphere Portal and IBM Lotus Web Content Management separated architecture, we no longer enable Lotus Web Content Management on all of the WebSphere Portal servers. Instead, we create a separate tier of servers with the sole task of serving Lotus Web Content Management content as illustrated in the following figure.
As with the other advanced architecture, this configuration clusters the portal servers. However, in this configuration, we create a separate cluster for the Lotus Web Content Management servers.

When you cluster the WebSphere Portal or Lotus Web Content Management servers, they share identical configurations within their respective cluster. This configuration is managed from a WebSphere Deployment Manager server (not pictured). The Deployment Manager is responsible for synchronizing the configuration across all servers in each cluster. This means that, when you deploy an updated Lotus Web Content Management portlet or create a new portal page, the Deployment Manager synchronizes all servers in the appropriate cluster, resulting in a reduction in administrative time and effort. In addition, when you cluster the WebSphere Portal or IBM Lotus Web Content Management servers, the Deployment Manager can create a plug-in for IBM HTTP Server or Apache that allows the HTTP Servers to gracefully failover in the event that one of the portal servers stops functioning. This is how both sets of HTTP Servers can provide failover for WebSphere Portal and Lotus Web Content Management requests.

**Design advantages**
The WebSphere Portal and IBM Lotus Web Content Management separated architecture has the following design advantages:

- **Failover:** The addition of a cluster allows the HTTP Server to provide failover for each cluster. If any one part of the system fails, the entire site does not appear to be down.
- **Maintenance:** With multiple servers available to deliver content, site maintenance can be performed without the entire site becoming unavailable.
- **Administration:** As more WebSphere Portal or IBM Lotus Web Content Management servers are added to the infrastructure the maintenance effort to maintain the infrastructure does not increase significantly. Changes to server configurations are performed from the central Deployment Manager console.

**Design disadvantages**
The WebSphere Portal and IBM Lotus Web Content Management separated architecture has the following design disadvantages:
• Complexity: This design is fairly complex. The learning curve for installing and administering a clustered environment servers should not be underestimated.
• Remote Rendering Portlet: If IBM Lotus Web Content Management is running on a separate set of servers all portal servers, you must use the Remote Rendering Portlet. Because this portlet communicates with an external IBM Lotus Web Content Management server, there is a potential performance bottleneck if the network connection is poor.

**IBM Lotus Web Content Management system components**

This section defines the key components of IBM Lotus Web Content Management.

**IBM Lotus Web Content Management Server**
The IBM Lotus Web Content Management Content Server is the core of the IBM Lotus Web Content Management application. All requests for content are ultimately processed by the Content Server.

The IBM Lotus Web Content Management Content Server leverages Virtual Member Manager as its user repository. This repository is used for both authentication as well as determining group membership for authenticated users.

**IBM Lotus Web Content Management Content Repository**
When you first enable IBM Lotus Web Content Management, it uses WebSphere Portal’s embedded Apache® Derby database as its content repository.

If desired, you can switch your IBM Lotus Web Content Management repository to any of the supported databases.

**IBM Lotus Web Content Management Authoring Portlet**
The user interface for working with IBM Lotus Web Content Management is provided via a Portlet running within WebSphere Portal.

The Authoring Portlet is (more accurately) your sole graphical user interface for interacting with IBM Lotus Web Content Management. Content creators use the portlet to author content. Content approvers use this same portlet for managing content.

Developers use this portlet to create technical assets (for example, Presentation Templates, Workflows, Syndication components, and so forth). The Authoring Portlet allows for very granular user interface security controls.

**IBM Lotus Web Content Management Connect Servlet**
The Connect Servlet is used to deliver Web content outside of a WebSphere Portal environment. You would use this servlet when you need to deliver a traditional standalone Web site. Site visitors can access content by requesting the HTML directly from the Connect Servlet.

It is important to note that even if you only use Connect Servlet to deliver stand-alone (non-portal) Web sites, the core IBM Lotus Web Content Management™ application always runs on WebSphere Portal.

**IBM Lotus Web Content Management Rendering Portlets**
IBM Lotus Web Content Management provides two Portlets that can display your content in WebSphere Portal. These portlets require no Java coding - only simple configuration by the portal administrator.

**IBM Lotus Web Content Management Local Rendering Portlet**
A local rendering portlet displays Web content on the same portal server as the instance where IBM Lotus Web Content Management is installed. This portlet can only be used when deployed to a WebSphere Portal Server that also has a live instance of IBM Lotus Web Content Management.
The Local Rendering Portlet relies on the IBM Lotus Web Content Management API to interact with the IBM Lotus Web Content Management Content Server.

**IBM Lotus Web Content Management Remote Rendering Portlet**
A remote rendering portlet displays Web content on a different portal server than the instance where IBM Lotus Web Content Management is installed. Unlike the Local Rendering Portlet, the Remote Rendering Portlet uses an HTTP connection to interact with a remote IBM Lotus Web Content Management Content Server at run time. The configuration of this portlet contains some additional fields to facilitate this type of connection.

**IBM Lotus Web Content Management Public API**
While IBM Lotus Web Content Management provides a solid set of Web content management capabilities out-of-the-box, there are times when the standard capabilities of IBM Lotus Web Content Management do not meet your exact needs. For situations such as this IBM Workplace Web Content Management provides a Java API. It is important to note that the IBM Lotus Web Content Management API does not expose all the capabilities of the IBM Lotus Web Content Management application. The JavaDocs should be reviewed for a complete set of the features available using the API. The Javadoc HTML files are located under the was_profile_root folder. In this path name, was_profile_root is the profile root for WebSphere Portal Server.

**IBM Lotus Web Content Management JavaServer Pages Tags**
In addition to the Java API, IBM Lotus Web Content Management provides a JavaServer Pages Tag Library that you can use when developing Portlets and other J2EE applications. The tags in this library make it very easy to access your IBM Lotus Web Content Management content from a JSP page. These tags rely on the API for their functionality so they do not provide any capabilities beyond those of the API.

**Understanding roles in IBM Lotus Web Content Management**
Different types of users are responsible for different tasks when deploying a Web Content Management system or when creating and managing Web sites and Web content.

**System wide roles**
Web Content Management is a component of WebSphere Portal. As such, the deployment process for Web Content Management systems is a sub-process of the overall WebSphere Portal deployment process. Many of the overall system planning and deployment tasks will be performed by users with system-wide responsibilities.

**Web content administrator**
A Web content administrator is responsible for configuring the Web content servers within a WebSphere Portal system.

**Web content information architect**
The information architect is responsible for determining the architecture of a site including things like library structure, the site framework, categories and site navigation as well as delivery methodologies.

**Web content graphic designer**
A Web content graphic designer is responsible for designing the layout and style of the Web pages and graphics in a Web site.

**Web site creator**
A Web site creator is responsible for building a Web site by creating presentation templates, authoring templates, site areas, components and categories. They are also responsible for creating content management items such as folders and workflows. The items created by a Web site creator are based on the designs created by the information architect and graphic designer.

**Web content developer**
A Web content developer is responsible for extending Web Content Management by using the Web Content API, developing JSP components and creating Web content plug-ins.
Web content author
A Web content author is responsible for creating Web content for the sites developed and managed using Web Content Management.

Web content approver
A Web content approver is responsible for approving Web content items that use workflows.

Web content manager
A Web content manager is responsible for performing ongoing site maintenance activities.

Web content tester
A Web content tester is responsible for performing functional and performance user acceptance testing (UAT) within UAT environments.

Web content viewer
A Web content viewer is the person viewing your Web site. They could be an internal viewer within a company intranet, or an external viewer such as a customer of a commercial Web site.

Case Study of moving River Bend to a Portal content centric site

This section provides information and examples related to the movement of a fictional site (River Bend Coffee) to a Portal content centric site.

River Bend is a fictional coffee company created as part of another wiki: Building a website using WCM 6.1: http://www-10.lotus.com/ldd/portalwiki.nsf/dx/WCM-Master-Table-of-Contents

For the purpose of providing a realistic business context to this wiki, we reuse the River Bend Tea and Coffee Company as the basis for the development scenario. River Bend Coffee and Tea Company, a subsidiary of WWCorp, is a fictitious company that uses Lotus Web Content Management software.

It operates a chain of 20 retail stores in 12 cities worldwide. In addition, the company runs an Internet-based retail operation, offers small-scale catering services, and has launched a certification program for employees and customers who wish to become skilled roast masters.

The figures below illustrate highlights of their current site and provides a brief overview of the components we build in this section.
The current economic landscape in the coffee market pushed River Bend to invest strongly in their Web site to increase customers awareness and one of the key initiatives that the IT department will be taking on is migrating the current WCM based site to a Portal centric solution that would provide these advantages:

- Improved branding through Portal themes
- User self-care and personalization
- Rich user interface through client-side aggregation
- Integration of external services consumption such as widgets and google gadgets
- Access to online services
- Integration of social software services like blogs and wikis
This figure shows the new Portal based site that provides a stronger platform to achieve River Bend business goals while improving the old user interface with Portal rich user interface features:

**Migration process**

The following sections cover the steps in the migration process.

**Import River Bend library**

The River Bend WCM library was imported in a fresh Portal 6.1.5 installation by using the standard JCR import tool.

**Adapt presentation artifacts for Portal**

Most of the work that had to be done was related with adapting the old user interface to be Portal based. River Bend presentation templates were based on WCM components to display the banner, navigation elements and footer so it was necessary only to remove those elements from the existing templates to make existing content ready to be displayed through portlets.

This figure shows how a content item was displayed on the old River Bend site:
This figure shows the same content item displayed in two separate portlets, one for the launch content and the other for a menu of items:

Our beverages are second to none, and the unique River Bend experience only adds to the flavour.

When we receive our coffee beans and teas we work our magic turning the typically odourless base product into a world beating beverage through roast or steaming them. If your not a tea and coffee connoisseur then or store stock a range of juices, smoothies, soft drinks (cola, lemonade, etc.) all made locally and sometime in store.

Where ever possible we take in suggestions and influences from in country locals like yourself so you can always rely on a flavoursome, and familiar beverage at a River Bend store.

The following shows the code for a WCM based presentation template:

```html
<! DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd" >
<html>
<Component name="html-header" htmlencode="true" />
<body>
<!-- set bg colour, x-y overlay image and fonts -->
<div id="mainBg" >
<!-- main transparent background 946px -->
<div id="fullWidth">
```
And this is the same presentation template adapted to be shown in a portlet:

```html
<div class="lotusui">
  <div class="lotusContent">
    <div class="lotusHeader">
      <h1><IDCmpnt context="current" type="content" field="title"/></h1>
      <h3><Element context="current" type="content" key="Summary"/></h3>
    </div><!--end header-->
    <div>
      <Element context="current" type="content" key="Body"/>
      <p>
        <img src="<Element context="current" type="content" key="Image" format="url"/>">
        weight="200" height="200" />
      </p>
    </div>
  </div><!--end content-->
</div>
```

Something you should notice is that Portal theme css styles are being used in presentation templates to keep a consistent user interface.

**Create a Portal theme**

As we mentioned earlier, the site branding will be managed by a Portal theme so you must create or configure an existing Portal theme to generate a look and feel similar to the old WCM based site.

**Create a Portal page hierarchy**

Once you have adapted your presentation templates and menu components to be displayed in a portlet you should create a page hierarchy that links part of your site area with the Portal navigation tree.

One thing to keep in mind is that not all your site areas need to be converted into Portal pages. Alternatively, you can use a portlet to display a menu or navigator component that lists the last levels of navigation just like River Bend did in the Beverages area:
It's strongly recommended to use both WCM based pages and JSR 286 Web content viewers to allow portal to automatically link content between different pages while browsing, previewing or displaying search results. To create a Web content page proceed as follows:

- From Portal administration click on **New page from**

  ![Web content page creation](image)

- Enter page name, URL mapping and click on Web content folder and Select
Select the WCM area that will be linked to this Portal page.
• This will create a new Portal page linked to the WCM site area. Now it is time to add a WCM JSR 286 viewer to display some content on it. To do this just open the page, put it in edit mode and drag and drop a Portlet instance:

• The portlet will display the default content for the area linked with this page without further configuration

Next steps
Portal provides over WCM an extra aggregation layer that provides three big advantages: improve personalization and customization, integration with other services and rich user interfaces.

This features might be leveraged by River Bend to create a web 2.0 portal that adds value to their customer on top of their existing content.

This figures and the next sections provide a description of the kind of services that can be used to improve the existing services:

**Branding through themes**
Portal themes will allow River Bend to separate the site branding from the content making it easier to:

- Test and update the site design more frequently
- Use different look and feels for different brands or countries
- Allow the end user to personalize some part of the interface such as the color palette
- Reuse Portal theme widgets to give rich user interface theme components like navigation menus or search boxes.

**Site navigation and dynamic layout management**
Managing the navigation of the site through Portal will make it easier to create new pages and mix Web content with external and internal services such applications, widgets or information that comes from feeds.

Dynamic layout management will allow River Bend Web masters to update the content and structure the home page easily making the site more attractive to customers with just a few clicks.

**User self care and personalization**
Portal customization allows the end customer to personalize their Portal experience in different ways:

- Customize page contents and structure
• Updating their profile to reflect their topics of interest

River Bend Web masters can use personalization campaigns to show or hide parts of the Portal navigation as well as specific page components.

**Rich UI features**

Portal rich user interface features such as client-side aggregation or live text will give River Bend customers a better user experience with minimal changes to the existing content. WCM presentation templates can leverage Dojo widgets to build rich components such as image galleries or graphical effects.

**Integration of services**

Current WCM content can be mixed in a Portal page with services and applications either internal or external to provide customers role-based, contextual pages. Some examples are:

• Widgets and Google Gadgets
• Wikis, Blogs, bookmarks, activities, communities or files hosted on Lotus Connections
• Team spaces in Lotus Quickr
• Federated content from ECM systems displayed through personalization elements
• Transactional services such order management through portlets.
Web 2.0

Web 2.0 is a term that has many aspects. Of course one aspect is the technology that can be used to generate the next generation of web based application and services. In addition, Web 2.0 encompasses new approaches to IT such as services as a product and cloud computing. It also covers business models and trends that are being used to gain competitive advantage such as self-established communities, user contribution and allowing users to take control and customize applications.

How do Web 1.0 and Web 2.0 sites differ

In the Web 1.0 scenario, the web master runs the site and the human users consume it. The Web site provides information and services, although the amount of information/content is relatively small. The information flows in only one direction and is organized into a fixed taxonomy.

Conversely, in the Web 2.0 scenario the information flow is bidirectional. The Web site provides content and applications, but now the users collectively contribute to the Web site in addition to consuming it. They do this by creating, editing and rating content. This results in a much larger amount of information and content that is categorized with flexible tagging. Users of the Web site are not limited to humans; other applications can also be consumers.

Separating user interface from services

A traditional J2EE Web 1.0 application follows a pattern of request from the browser with a full page response from the server:

Since browsers became capable of making asynchronous HTTP requests, the pattern for Web 2.0 applications has changed. The browser makes requests to one or more services and the Web page can be updated partially with the result of each service.
Providing fine grained services like this brings the possibility to create new breeds of application such as mashups.

**Web 2.0 technologies**

At a technical level, the services are often created leveraging technologies such as REST (Representational state transfer), JSON (JavaScript Object Notation), and consumed by Ajax (Asynchronous JavaScript and XML) applications. These concepts are well documented elsewhere and are not specific to Portal so are only briefly summarized below:

- **REST** - a server side architectural model that relies on the semantics of the HTTP protocol to provide access to services. This makes such services very easy to consume in browsers (or other clients) and hence they are very popular in the context of Web 2.0 applications using Ajax. REST provides a resource-oriented approach to services, as opposed to a Remote Procedure Call (RPC) centric approach. For example, all resources are addressable through relative URLs, for example /riverblend/employees or /riverblend/employees/Michael.

- **JSON** - often the payload returned by a REST service when being consumed by a Web browser. The service may emit alternative payloads for different clients such as XML, RSS or ATOM. JSON is the native format of JavaScript that makes it very easy to consume in Ajax applications. The JSON payload can be directly assigned to a JavaScript object providing easy access to its data structure. Compare this to calling a Web service from a browser. While this is technically possible, it is much more difficult for the browser to manage the XML response as constant navigation of that XML document object model would be required.

- **Ajax** - a programming pattern for creating richer and more responsive Web pages. It is often used to build Web clients in a Service Oriented Architecture allowing the browser to directly access services deployed to any kind of server. Ajax is built upon existing open technologies and standards such as JavaScript and XML and does not require any proprietary browser plugins. Because more computation is done in the browser, an Ajax application can reduce the load on the server. A key feature of Ajax is the ability to make an asynchronous request from the browser, and update part of the page with new markup when the call returns.

- **Dojo** - an Open Source DHTML toolkit written in JavaScript. It is very useful for developing Ajax applications with rich user interfaces as it simplifies JavaScript development and provides reusable user interface widgets. Dojo is used in WebSphere Portal to provide Web 2.0 features. It is an IBM supported toolkit for creating custom portlets that require Web 2.0 features.

Some of these technologies will be explored in more detail in the case study later in this chapter that demonstrates how to develop portlets with Web 2.0 features.

**Considerations for using Web 2.0 with Portal**

When a page is updated with an Ajax request, the URL in the browser toolbar is unaffected. This can cause problems for standard Web browser features that rely on the URL to rebuild the page. For example, back and forward buttons, bookmarking, reverse proxy caching and crawlability could all be affected if an Ajax request had modified the page. These problems are valid for both Web and Portal applications. However, in
Portal, the problem is even more prevalent due to the fact that a Portal page is an aggregation of several applications (usually portlets).

The URL for a Portal page includes information about navigational state. The navigational state includes information about the entire page (page selection, theme template, and so on) and information about all portlets including their rendering parameters. Therefore if a single portlet used Ajax to update its view, all links held in that portlet and all other portlets would conceptually need to be updated to encapsulate the changed view. One way to tackle this problem is to use a Portal aware JavaScript library that can manage URLs. This feature will be explained in detail later, but for now let's summarize how it can be used to address the challenges introduced in this section:

<table>
<thead>
<tr>
<th>Capability</th>
<th>Portal Web 1.0</th>
<th>Portal Web 2.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bookmarking</td>
<td>A user can create a browser bookmark from any link on a page. Each link contains the latest navigational state of all portal artifacts visited so far.</td>
<td>This can be supported with a special &quot;Bookmark this page&quot; link. A user would have to select this link which asynchronously invokes a server side method to return a URL that is 'bookmarkable'.</td>
</tr>
<tr>
<td>Browser Back/Forward/Reload</td>
<td>A user can navigate using the browsers back, forward and reload button with the expected behavior (for example, like static html pages).</td>
<td>Javascript Frameworks such as Dojo support back and forward button in a limited fashion. They enable back and forward events to be captured by the portlet which can maintain its own history of URLs. However this will not be reflected in the browser history as would be expected in Web 1.0.</td>
</tr>
<tr>
<td>Browser and Reverse Proxy Caching</td>
<td>The full portal page is cached in a reverse proxy or browser cache. The page expiration is calculated by Portal to be the shortest expiration of all artifacts on the page.</td>
<td>Using the Portal JavaScript API, each artifact on the page is loaded on demand from within the browser. Therefore, the reverse proxy and browser cache hold each artifact individually gaining the maximum benefit from caching. If an artifact never expires it will not be loaded again with follow-on requests. The caching potential is actually increased with Web 2.0 compared with Web 1.0 where the whole page would expire after the portlet with the shortest caching time had expired.</td>
</tr>
<tr>
<td>Crawlability</td>
<td>The markup that is returned from a portlet is analyzed by the search engine. The best practice is to return special markup with 'normalized URLs' when a portlet is</td>
<td>Depending on the extent of Ajax usage, this could be tricky to achieve as current internet search engines do not understand JavaScript.</td>
</tr>
</tbody>
</table>
If the Portal simply renders a 'bootstrap' page that then loads each portlet with Ajax, the search engine will not understand this and the portlet content will not be crawled. This limitation should be considered when comparing Portal's server side and client side aggregation themes (discussed later).

---

**WebSphere Portal Web 2.0 features**

WebSphere Portal leverages the Web 2.0 technologies mentioned in the previous section to provide new out of the box Web 2.0 features and capabilities. The following topics will be expanded upon in this section.

- Portal REST Services
- Client Side Aggregation (CSA) Theme
- Ajax Proxy
- Semantic Tagging with Live Text API
- Google Gadgets
- Lotus Mashups
- Client side feed consumption

**Portal REST services**

Portal provides a rich catalog of REST services to facilitate remote access to core portal data. They separate the portal user interface from the portal data.

The REST services are all public APIs and can be accessed from Portal applications running within the WebSphere Portal framework and Web-based applications, external to the Portal. GET, POST, PUT and DELETE HTTP methods are supported via a URL addressable interface.

- Content Model - Allows you to obtain and modify the content topology and the properties of content nodes such as pages, labels, and content URL’s.
- Layout Model - Allows you to obtain and modify the layout of a page, that is, the topology of layout elements of a page, and the properties of layout elements, such as layout containers and layout controls.
- Navigation Model - Allows you to obtain the navigation topology only, as the navigation model is implied by the structure of the content model.
- Portlet Model - Allows you to obtain, create, update, and delete portlets.

When accessing a REST service to get information or to modify a portal resource, the response and in these cases also the request works by means of a feed. A feed contains information about one or more portal resources in a specific format as exposed by portal models.

In order to obtain a feed that contains certain portal resources, send an HTTP GET request to the following URL.

`/wps/[my]contenthandler[/<vpmapping>]?uri=<model-uri>+&mode=<verb>+&name=<value>`

- contenthandler - The name of the servlet for unauthenticated access. This is the equivalent to /wps/portal/, where anonymous access is allowed.
- mycontenthandler - The name of the servlet for authenticated access. This is the equivalent to /wps/myportal/, where only authenticated access is allowed.
• vpmapping - The optional name of the URL mapping to a virtual portal. If no mapping is given, the default virtual portal is assumed, otherwise the ID of the addressed virtual portal is associated with the current thread.
• model-uri - The identification of the addressed resource, as described further below. The uri parameter can appear only once.
• page-oid - This is the portal object ID of a content page. When you address portlet model resources, this is required for portlet windows, but it must not occur for any other portlet model resource identification.
• navigationnode-oid - This is the object ID of a navigation model node.
• layoutnode-oid - This is the object ID of a layout model node.
• portlet-oid - This is the object ID of a layout control, portlet window, portlet entity, or portlet definition.
• oid - This is the serialized string that represents a portal object ID. This is URI-escaped with UTF-8 encoding. The character @ is also escaped by using %40. You can also use unique names instead of object IDs. Unique names also have to be URI-escaped, and the @ character must be escaped. Note that every object ID has the defined scheme 'oid:'.
• verb - An optional specification for the mode of access. By default mode=download is assumed.
• value - An arbitrary set of parameters. These include additional query parameters. For details refer to the topics further below.

If modifying an existing portal resource it is recommended to send an HTTP GET request and first retrieve that portal resource. The response will be an existing XML feed of a page which can then be edited and used as a template for changes. It will also come in handy if there is a need to backup to a previous version.

To obtain a feed of the layout node with the object id 'l_7_0830M4HTFF0SHFCQ_2BV' on the content page with the object id 'l_6_0830M4HTFF0SHFCQ_4D', send an HTTP GET request to the following URL:

/wps/mycontenthandler?uri=lm:oid:_7_0830M4HTFF0SHFCQ_2BV@oid:_6_0830M4HTFF0SHFCQ_4D

Client side aggregation themes
Client side aggregation (CSA) is the new aggregation model that provides an improved user experience through faster response and performance. Themes that are CSA enabled are rendered from JavaScript and XSLT transformations that are executed on the client. CSA considers portlets as fragments that are reloaded individually. There are two main advantages for using a CSA enabled Portal theme:

• Improved user experience by eliminating full page refreshes. Only individual portlets are refreshed as users navigate a page.
• Improved server-side performance and scalability.
• Reduced Server-side processing - rendering workload is off-loaded to the browser.
• Reduced bandwidth requirements between the server and browser.
• Improved HTTP caching - all artifacts can be cached independently.
• Reduced client side processing - fragmented reloads, fewer full-page reloads

The PortalWeb2 theme should be used as the starting point for all CSA themes. Make a backup copy of this first and then gradually adopt the existing PortalWeb2 theme. Portal themes are located in the following directory:

<was_profile_root>/installedApps/<cell_name>/wps.ear/wps.war/themes/
The PortalWeb2 theme directory structure is described in the figure above. The CSA theme makes heavy use of the dojo APIs; therefore it is packaged separately, under the dojo directory to keep the main complexity outside of the PortalWeb2 theme.

The CSA theme should always support the server side aggregation (SSA) framework. Not every portlet will work with CSA due to JavaScript conflicts, non-standard coding practices, or insufficient browser capabilities.

SSA mode can be enforced via:

- A portlet initialization parameter in portlet.xml
- com.ibm.wps.web2.renderMode=force_SSA
- Page parameter / meta data in edit page properties
- RenderMode = force_ssa
- Portlet blacklist (../themes/PortalWeb2/js/portletsForServerSideOnly.js)
- Add the portlet definition UID (defined in portlet.xml) to the list

**Portal versus PortalWeb2 theme**

The CSA enabled theme is not much different from the SSA theme; a few files were added or modified. When a CSA theme is rendered, a skeleton HTML document containing placeholders is rendered first. A JavaScript bootstrap function gets called after the skeleton finishes loading. This bootstrap process calls the CSA framework to initiate the CSA rendering process that replaces the placeholders with HTML markup.

The modified files are as follows:

- head_extras.jspf - Initializes CSA and initiates bootstrap
- js.jsp - Loads the CSA specific libraries such as dojo
- status_bar.jspf - Defines HTML anchor for the status bar

The new files and folders are as follows:
• xsl folder - Contains XSL templates producing the widget markup
• CSAProperties.jspf - Loads resource bundle strings needed by CSA
• head_themePolicy.jspf - Stores current theme policy in a javascript variable

Ajax proxy
The Ajax Proxy is a server side component that acts as a http forwarding proxy, which can be used to control access and route requests to different domains. It allows security administrators to control portlet access to external domains.

To prevent cross site scripting in such Web applications, browsers introduced the so called same-origin policy. This policy prevents client side scripts, in particular JavaScript, from loading content from an origin that has a different protocol, domain name, or port.

The proxy configuration can be set as global or application specific.

• A global proxy configuration applies to every application deployed on the portal, regardless if the application has its own proxy configuration file. In large deployments with many deployed portlets, using only the global proxy configuration creates less administration overhead. In such environments, application specific configurations can create more administration overhead due to the fact that a security administrator needs to maintain multiple proxy-config.xml files spread across different WAR and EAR files.

• In the application specific proxy model, each portal application packages its own proxy configuration file. The portal application proxy takes precedence over the global proxy configuration.

The application specific proxy configuration is specified in a proxy-config.xml file, that must be placed in the WEB-INF directory of the portlet WAR file.

The global proxy configuration is also in a proxy-config.xml file. This file is located within the WEB-INF directory of the wp.config.proxy.war file in an EAR called AJAX Proxy Configuration.ear. See below:

<profile_root>/installedApps/<cell_name>/Ajax Proxy Configuration.ear/wp.proxy.config.war/WEB-INF/proxy-config.xml

To enable the global proxy configuration specify the following configuration setting in the portal Config Service.

proxy.enforce.global.config = true

The following link details the process of modifying the ConfigService:

To update the global proxy configuration, redeploy the enterprise application archive that contains the modified proxy-config.xml file and restart it. You do not need to restart the entire portal server.
When a request is received by the Ajax Proxy, the proxy locates a corresponding policy url. The asterisk (*) in the above example denotes that requests for any target URL is accepted. However, only GET and HEAD HTTP requests are allowed.

```xml
  <proxy:mapping contextpath="/proxy" url="*" />
  <proxy:policy url="*">
    <proxy:actions>
      <proxy:method>GET</proxy:method>
      <proxy:method>HEAD</proxy:method>
    </proxy:actions>
  </proxy:policy>
  <proxy:meta-data>
    <proxy:name>max-connections-per-host</proxy:name>
    <proxy:value>5</proxy:value>
  </proxy:meta-data>
</proxy-rules>
```

In the configuration above, GET, HEAD, POST, and PUT HTTP requests are allowed solely for the http://www-01.ibm.com/software/lotus/ URL and sub-paths below it.

Meanwhile for http://www.cnn.com/, a user must be authenticated, and only GET and HEAD requests are allowed. In addition, only the specified cookies and mime-types will be accepted by the proxy.

```xml
<proxy-rules xmlns:xsi="" xmlns:proxy="">
  <proxy:mapping contextpath="/1bm" url="http://w3.ibm.com" />
    <proxy:actions>
      <proxy:method>GET</proxy:method>
      <proxy:method>HEAD</proxy:method>
      <proxy:method>POST</proxy:method>
      <proxy:method>PUT</proxy:method>
    </proxy:actions>
  </proxy:policy>
  <proxy:policy url="http://www.cnn/*">
    <proxy:actions>
      <proxy:method>GET</proxy:method>
      <proxy:method>HEAD</proxy:method>
    </proxy:actions>
    <proxy:mime-types>
      <proxy:mime-type>text/html</proxy:mime-type>
      <proxy:mime-type>application/xml</proxy:mime-type>
    </proxy:mime-types>
    <proxy:cookies>
      <proxy:cookie>LTPA</proxy:cookie>
      <proxy:cookie>CoreID</proxy:cookie>
    </proxy:cookies>
    <proxy:users>
      <proxy:user>AllAuthenticatedUsers</proxy:user>
      <proxy:user>Admin</proxy:user>
    </proxy:users>
  </proxy:policy>
</proxy-rules>
```
How the Ajax proxy works

1. A user accesses a portal page with a portlet on it that requires markup from an external datasource.
2. Portlet renders through on a portal page.
3. The JavaScript code from the portlet application attempts to access an external site (for example: www.cnn.com) through a well defined URL pattern.
4. The Ajax proxy applies its security check.
5. If the domain is a trusted domain then the request is forwarded to the target server at the target domain.
6. When the request is returned from the target domain, it gets passed through the server back to the browser.

Semantic tagging with Live Text API

Live Text provides elements embedded in portal pages that become active in the browser and are enhanced with additional functionality by JavaScript libraries. For example, if you include portal user IDs in your portlet output and mark them as live text, users can click on these IDs in the browser and see a person’s information card or a context menu that allows them to send an e-mail to the person, initiate a chat, view their blog, or make use of many other extensible menu-option features.

Live text has the following advantages:

- Live Text triggers can be declared by any HTML fragment on a page, including portlets, theme components or simple HTML clippings.
- Adopt new portal content within your company more easily, as it is now easier to handle portal tags. For example, you can write tags and make them available centrally, and UI developers can reuse these tags for in their portlets for various purposes.
- Content editors can add meaningful live text elements to portlets without requiring portlet development knowledge.
- Embed content from other sources, for example, from a HTTP or .NET server.
Google Gadgets
IBM Portlet for Google Gadgets is a JSR 168 portlet that enables WebSphere Portal users to integrate Google Gadgets for Web pages with their portal pages. Google Gadgets for Web pages are remotely accessible services that provide access to online content and applications designed to be aggregated into the context of a Web page. These include language translators, maps, YouTube videos, and Wikipedia. A full listing of the 15,000+ available gadgets can be found here:
http://www.google.com/ig/directory?synd=open

The Google Gadget portlet has been updated to facilitate inter-portlet communication. Data can be exchanged between the IBM Portlet for Google Gadget portlet instances and other portlets. This feature enables other portlets on a page to interact with the selected gadget(s). The gadget portlet reacts in response to an event triggered in a different portlet. For example, a user clicks on the name of a city in a portlet application. The portlet application sends, transparent to the end-user, the city information to a Weather Google Gadget portlet. The weather gadget then automatically shows the current conditions for that city.
The figure above is a picture of the Google Gadget selection interface within WebSphere Portal.

**Lotus Mashups**
A Mashup is a lightweight web application created by combining information or capabilities from multiple existing sources to deliver new functions and insights. IBM Mashup Center is a lightweight mashup platform, supporting line of business assembly of simple, flexible, and dynamic web applications with the management, security and governance capabilities IT requires.

Below is a list of advantages of using Lotus Mashup Center

- Assemble applications by reusing existing services
- Unlock Enterprise, Web, Personal and Departmental Information
- Create interactive widgets that encapsulate existing services
- Discover and share mashups, widgets, and feeds
- Transform and mix information into new feeds
- Explore different combinations to uncover new insights

The figure below is an example of a business application created using IBM Mashup Center.
Lotus Mashups integration into Portal
IBM Lotus Mashup Center is a platform for rapidly creating, sharing, and discovering reusable assets called widgets. Mashups consist of one or more widgets. Widgets can run within the context of WebSphere Portal using any of the following deployment options.

- Deploy widget to WebSphere Portal via a pre-built, reusable JSR 168/286 portlet. This portlet is available on the WebSphere Portal Business Solution catalog here:


The WebSphere Portal Widget Portlet provides customizable, personalized, and managed access to widgets that are compliant with the iWidget Specification, thus extending the existing capabilities of WebSphere Portal to provide flexible integration with web components, independent of their technology. With the IBM WebSphere Portal Widget Portlet, users can leverage widgets from Lotus Mashup Center on their portal pages side by side with portlets, Google Gadgets or other web components, enabling the creation of portal-based mashups tailored to the situational requirements of an enterprise.

The Widget Portlet provides the following widget integration capabilities. A widget can be added to Portal by:

- Pointing at a URL
- Browsing the Lotus Mashups Toolbox
- Browsing the Mashup Center Catalog
The screenshot above displays this interface as seen in WebSphere Portal.

- **Deploy a widget locally to WebSphere Portal**
  
  If a widget is implemented as a J2EE components, which are typically Web archive (WAR) files containing components like servlets, portlets, HTML pages and others, they can be deployed to WebSphere Portal. The following steps list the procedure to deploy a widget from Lotus Mashup Center and install it natively on WebSphere Portal.
  
  - A portal administrator downloads the widget from the IBM Lotus Mashup Center catalog as a WAR file.
  - A portal administrator deploys the WAR file using the WebSphere Application Server administration interface.
  - A portal administrator uses the Widget Portlet to integrate the locally deployed widget as a portlet.

- **Display widgets and mashups in any web page (Portal or non-Portal)**
  
  With a few simple clicks widgets and mashups created in IBM Mashup center can be displayed in any web page. In this scenario the widget or mashup still resides on the Mashup server. The hosting web page the knows to call the Mashup server through a script tag added to the web page source.
The figure above from Mashup Center shows how a user can create the one-line script to be inserted into a web page where the mashup or widget is to be displayed.

Widgets in the Mashup Center toolbox can also be embedded into web pages with this same technique.

- Promote Mashups to WebSphere Portal. (future capability)

In early 2010, users will be able to install their Mashups on WebSphere Portal from IBM Mashup Center with the click of a button. It will no longer be necessary to manually export the Mashup as a WAR file. This high-level process is described in the figure below.
Ajax portlet development

Portlets are reusable Web modules that can be developed separately providing access to Web-based content, applications, and other resources. Portlets are intended to be assembled into a larger portal page, with multiple instances of the same portlet displaying different data for each user. Portlets can use different technologies (DHTML, IFrames, Plugins or even Applets) and leverage Ajax/Rich Internet Application (RIA) techniques and libraries directly, or encapsulated in frameworks.

Portal Ajax scenarios
In a portal, Ajax is typically used to enhance the user experience with two scenarios:

1. Data retrieval

In this scenario, the browser client makes an asynchronous request to load data using a data encapsulating protocol such as XML or JSON. JavaScript is used to processes the data and update a view component. This update is displayed without reloading the whole page. Examples of this scenario are:

- Typeahead feature - A user enters text into a text field and receives a list of alternatives about the entered text.
- Refreshing stock quotes - A portlet displays the latest stock quotes every five minutes.

2. Markup Retrieval

As an alternative to loading data, the client can load markup (e.g. HTML) and apply it directly to DOM elements of the current page.

An example of this scenario:

- A link or button is clicked on the page that triggers an asynchronous request to load markup from the server. The markup is received and is used to directly replace the current view of a DOM element.
Approaches to AJAX portlet development

Ajax support in portlets differs depending on which Java Portlet Specification level being used:

- **JSR 168** - Ajax is supported and the most common use cases use external servlets to provide a data source for serving markup or data.
- **JSR 286** - Introduces a new feature called resource serving that enables portlets to serve independent markup within the portlet context using resource requests.

At each specification level, there are alternative design patterns that can be considered.

**JSR 168 portlets**

Ajax support in JSR 168 is provided through a servlet implementation. In this scenario, the portlet WAR file packages one portlet and one servlet. The servlet acts as a proxy that connects the portlet to an external data source. This servlet is needed because there is no way to address the portlet directly to retrieve fragments of data or markup. The servlet is packaged in the same WAR file so the servlet and portlet can exchange data using the HTTP session (application scope). For example, in response to an asynchronous browser request, the servlet might access an email id in session, and return an email message.

This Ajax scenario can be used to update backend data or clean server-side caches if necessary. However this scenario has some limitations:

- The Ajax call cannot use portlet APIs or affect the portlet state.
- View changes are typically lost in case of a full page refresh.
- Implementation of the servlet could be complex, particularly if it is used to connect to several different data sources.
- The servlet actually needs to act like a HTTP proxy which requires handling of cookies, redirects, request /response headers etc.

Be aware that more that one portlet instance can coexist in the same page so avoid conflicts in library versions, callback functions, variable names and avoid loading JavaScript libraries more than once. Some techniques to achieve this will be demonstrated in the case study.

The simplest approach for Ajax handling is to use Portal's Dojo implementation which is included in the default themes. WebSphere Portal contains an instance of the IBM Dojo Toolkit, a JavaScript library based on the Dojo toolkit (http://dojotoolkit.org). The bundled Dojo is packaged in the /installedApps/node_name/wps.ear/wps.war/themes/dojo/portal_dojo directory. It is intended to be usable by both IBM and non-IBM components. Dojo is used in Portal's themes so in order not to adversely affect them, it is important to refer to the guidelines in the Portal InfoCenter and adhere to best practices on how to use Dojo in portlets. More information on this can be found at:
JSR 268 portlets
This version of the portlet standard introduced resource serving, a key capability for Ajax development. This enables portlets to serve resources, for example, markup or data, from within the portlet context using resource requests. This means a separate servlet is no longer required and encourages a better programming model.

The new callback interface is triggered by resource URLs and the portlet has a new method: serveResource(RESOURCE_REQUEST, RESOURCE_RESPONSE). Resource serving allows portlets to render stand-alone content with no portal page aggregation, no theme content and give the portlet full control of the entire output while maintaining access to portal APIs and state data, such as the portal session.

Resource serving can be used to implement scenarios like generating binary content (PDF, SVG, and so on) from a portlet, to be displayed in pop-up windows. For other Ajax use cases, resource serving portlets have a way to return XML, JSON, HTML fragments or any other content.

The main advantages over the servlet scenario are:

- Ajax requests can now be handled inside the portlet where there is access to Portal API, session, and so on.
- Current state (navigational state, preferences, portlet session) can be accessed.
- Updates can be made to the portlet preferences or session via a POST request.

However there are some limitations to resource serving as it is not possible for the serveResource method to update navigational state or send events to other portlets. We will see how to overcome these limitations in the next section concerning the Portal client side programming model.

Client side portlet programming model
While most portlet Ajax scenarios can probably be addressed using JSR286 resource serving, the most flexible Ajax support requires coordination between portlet and Portal. To achieve this, Portal provides an Ajax library for portlets that allows server side features to be updated from the client. This includes a mechanism to update portlet preferences, navigational state or send events. Thus changes made in the client with Ajax are retained even if the user clicks outside the current portlet.
**Accessing the API**

This is a short summary of the main JavaScript API classes and methods. You can find a full copy of the API documentation at: [http://download.boulder.ibm.com/ibmdl/pub/software/dw/wes/0608_wp6javadoc/portal_61js.zip](http://download.boulder.ibm.com/ibmdl/pub/software/dw/wes/0608_wp6javadoc/portal_61js.zip)

**ibm.portal.portlet.PortletWindow**

This object is the entry point to the API providing JS functions access to the outlined functionality and defining useful constants for portlet modes, window states, response status and errors.
This object provides XMLHttpRequest functionality for portlets.
ibm.portal.portlet.XMLPortletRequest
Provides XMLHttpRequest functionality for portlets

dojoportaldeclare="ibm.portal.portlet.XMLPortletRequest", null, {
  // Properties to read the ready state as well as register callback functions
  readyState: 0,   /* int */
  onreadystatechange: null, /*Function*/
  onportletStatereddy: null, /*
  Function(ibm.portal.portlet.PortletState) */
  // Properties to access the response body (either text or XML)
  responseText: null, /* String */
  responseXML: null, /* Document */

  // Properties to read the response status
  status: null,   /* int */
  statusText: null, /* String */

  // Functions to open, send, and abort a synchronous or asynchronous request
  open: function| /*String*/ method, /*String*/ uri ) {...},
  open: function| /*String*/ method, /*String*/ uri, /*boolean*/
  async ), {...},
  send: function| /*String or Document*/ data ) {...},
  abort: function( ) {...},

  // Functions to set request headers and read response headers
  setRequestHeader: function( /*String*/ header, /*String*/ value )
  {...},
  getAllResponseHeaders: function( ) {...},
  getResponseHeader: function( /*String*/ header ) [...]}
}

ibm.portal.portlet.PortletState
Allows portlets to work with the render parameters, portlet mode and window state (navigational state) of the portlet window. Changes are persisted when calling setPortletState() on the PortletWindow object.
ibm.portal.portlet.PortletPreferences

Allows portlets to work with the configuration data (persistent state) of the portlet. Modifiable and read-only preferences are distinguished according to the portlet mode. Changes are persisted when calling setPortletPreferences() on the PortletWindow object.
Ajax portal tooling
Rational Application Developer v7.5 contains Portal tooling to support Web 2.0 features:

- JSR 286 support for creating portlets that perform resource serving
- Client-side programming model support. The new portlet wizard provides an option to import and initialize the client side programming libraries.
- Client-side click-to-action support. Automatic code generation is available to enable cooperation between portlets using client-side click-to-action.
- Person tagging support. Automatic code generation is available to create the Person menu, and extend the Person menu by writing JavaScript actions.
- Ajax proxy tooling is available to enable and configure this HTTP proxy for portlet projects.

Portal Web 2.0 case study
In this section some of the portlet development scenarios described previously will be demonstrated:

- Applying the PortalWeb2 theme to an existing Portal page / portlets
- Using client side Click to Action semantic tags and JSR286 resource serving with server side aggregation

Case study scenario
The River Bend coffee company now offers a ‘Farmers Market’ facility where customers can learn about the origins of their coffee products, and add individual coffees to a custom blend. The facility is aimed at both ethical coffee buyers who wish to support certain famers, or coffee connoisseurs who wish to create custom blends.

Applying the PortalWeb2 theme to an existing Portal page or portlet
In this scenario, the difference between server side and client side aggregation will be demonstrated. In the attachments to this wiki page you will find a Web module that can be imported into Portal. It contains two

```javascript
dojoportal.declare("ibm.portal.portlet.PortletPreferences", null, {
    // Returns a preferences map [object] [name: string, values: string[], readonly: boolean]
    getMap: function() {...},

    // Returns the names of the available preferences [String[]]
    getNames: function() {...},

    // Functions to read preference values (String / String[])
    getValue: function( /*String*/ key, /*String*/ default ) {...},
    getValues: function( /*String*/ key, /*String[]*/ default ) {...},

    // Checks whether the preference with the given name is a read-only preference
    isReadOnly: function( /*String*/ key ) {...},

    // Functions to set /reset preference values
    reset: function ( /*String*/ key ) {...},
    setValue: function ( /*String*/ key, /*String*/ value ) {...},
    setValues: function( /*String*/ key, /*String[]*/ value ) {...},

    // Function to clone the PortletPreferences object
    clone: function() {...}
});
```
portlets that provide a basic realization of the case study scenario.

![Image](image_url)

After importing the Web module, the portlets should be added to a page with the portal theme "Portal" that provides traditional server side aggregation. A wire should then be created to enable the inter-portlet communication. The wire is added by selecting from the drop down lists, clicking the "+" button, then clicking Done:

These portlets provide typical Web 1.0 functions:

- Users can page through the fifteen available coffee products from three different farmers.
- Products can be added to the shopping basket.
- The farmer's name can be clicked to display more detailed information in the farmer details portlet. This uses JSR286 events to perform the inter-portlet communication.

Note that the farmer details portlet intentionally takes a few seconds to respond to simulate a backend call that causes a 'white screen' effect. This could be reduced by moving the pause to the processEvent method instead of doView. However, the time taken to load the page where the user cannot perform any functions is the same.

All these functions result in a full round trip to the server with a page refresh. Note that as the navigational state is managed by the server, the render parameters always maintain information about which page is currently being viewed. The same page is always displayed, for example after adding a product to the basket. This is a subtle but important feature we will strive to maintain when discussing addition of Web 2.0 features in the case study.

The full page refresh of the Web 1.0 portlets does not provide a very satisfying user experience as the user
sees a blank screen after paging, or viewing farmer details. In addition, with every operation the full page refresh means the server must render all portlets even if only one has changed.

Portal’s Web 2.0 theme can provide a simple way to introduce Web 2.0 features without requiring any code changes to existing portlets. Change the Portal theme for the page containing the two portlets:

Title:

Coffee Farmers Market

Unique Name:

Note: If the unique name you entered for this page already exists, it will not be created or updated.

Friendly URL name:

Theme:

PortalWeb2

Theme Style (Theme Policy):

Inherit Parent Theme Policy

Now the user experience is improved as a click on the farmer name will result in the farmer details portlet updating its view independently of the coffee list portlet:

Also note that after adding a product to the basket or refreshing the page, the list of products does not default to the first page but shows the previously displayed page. This is due to an important feature of the Portal Web 2.0 theme that uses the Portal JavaScript library to automatically update navigational state. The portlet maintains a paging offset value as a hidden parameter in its form. The portlet stores this as a render parameter when the form is submitted during the action phase. This render parameter is then used to shift the starting page of the products during the rendering phase. After clicking the paging links, even though there is no page refresh to update the render parameter with the new page offset, the Portal Web 2.0 theme does this automatically.

Using client side Click to Action semantic tags and JSR286 Resource Serving with Server Side Aggregation

In this scenario, Portal’s semantic tags will be used to provide client side ‘click to action’ function, even when using a server side aggregation theme. In the attachments to this wiki page you will find a web module that can be imported into Portal. It contains an evolution of the two case study portlets. Replace the existing portlets by clicking Administration->Portlet Management->Web Modules. Search for the web module and click the update icon (the icon with arrows):
Modify the page on which the portlets are deployed to use the default Portal theme:

Title:

Coffee Farmers Market

Unique Name:

Note: If the unique name you entered for this page already exists, it will not be created or updated.

Friendly URL name:

Theme:

Portal

First let’s check the modified behavior of the portlets before investigating the code updates. The portlets were tested with IE7 and Firefox 3.5 on Portal 6.1.0.2 + APAR PK91594 (and its prerequisites, PK84631 & PK84603). At the time of writing the portlets will not work with Portal 6.1.5 until APAR PM02564 becomes available.

Note how the farmer names are now click to action (C2A) sources and hovering over them causes a menu to pop up. The JSR286 events code used in the previous case study example has now been removed. C2A does not require portlets to be wired.

When clicking the C2A source, there is just one available action, although more could be configured. The action is to show details for each farmer. This triggers a client side C2A event, which sends the farmer’s name to the details portlet. A full page refresh is not made and the farmer’s name can be seen immediately:
Next a spinning icon appears under the farmer's name, and the remaining details are queries are loaded using an Ajax request using JSR286 resource serving. After a moment the farmer details appear, again without a full page refresh:

**Client side click to action**

When the portlets were created using Rational Application Developer, the **Add Web 2.0 features** box was checked. This added a couple of new declarations to the JSP of a generated sample portlet that provide access to the Portal JavaScript API.

```
<portlet-client-model:init>
  <portlet-client-model:require module="ibm.portal.xml.*"/>
  <portlet-client-model:require module="ibm.portal.portlet.*"/>
</portlet-client-model:init>
```

The Click to Action source and targets are very easy to add using Rational Application Developer portlet tooling. Simply drag a Client Side Click to Action Source/Target Property from the Portlet palette, or use the Insert menu. The source or target can simply be dropped onto a field in the JSP editor and a wizard is used to gather information:
The JSP is modified like this:

```
<div class="c2a:source">
  <span class="c2a:typename" style="display:none">http://farmersmarket#MyDatatype</span>
  <span class="c2a:value" style="display:none">aCoffeeProduct.getFarmerName()</span>
  <span class="c2a:anchor" style="display:none">aCoffeeProduct.getFarmerName()"</span>
</div>
```

The source property has been replaced with the `<div>` element below which uses ‘micro formatting’ (a special class name) that Portal uses to identify the source property (c2a:anchor) and cause it to appear underlined. In addition there are some further `<span>` elements which are not visible to the user due to the applied style. However, they are used by Portal to perform client side C2A so must be left in place.

The Rational Application Developer portlet tooling can be used in a similar way to define a C2A target on the target portlet. This will generate markup similar to this:
On the target portlet’s JSP, a form is defined which contains an input element. The C2A framework uses Ajax to update the value of the input element with the value defined in the source portlet. Once the value is in place, the C2A framework triggers a form submit.

If the form’s ‘action’ pointed to a server side resource (for example, a portlet action URL), server side logic could be triggered to update the view. In this case, we want to use JavaScript to trigger an Ajax request to update the view. Therefore the form action is "javascript: void(0);" and the onsubmit JavaScript handler is used to trigger a function. Also note that entire form is hidden from view with style="visibility: hidden". In this scenario we want to receive the C2A value and be notified of this by the form’s onsubmit event, but we do not want to show a form with an input box. Instead, the JavaScript will access the C2A value directly from the Document Object Model (DOM). The next section describes this in more detail.

**JSR 286 resource serving with Ajax**

The farmer details are updated in the view using JSR resource serving. The portlet now references a custom JavaScript file in directory _FarmerDetails/js that uses Dojo to declare a JavaScript object acting as the client side controller.

The following code from FarmersDetailsPortletView.jsp imports the JavaScript file:

```javascript
<script language="JavaScript"
src="<%=renderResponse.encodeURL(renderRequest.getContextPath()+"/_FarmersDetails/js/FarmerDetailsController.js")%>"/></script>
```

There are some important concepts to note when using JavaScript in a portlet. The script statement loads the JavaScript file using the renderRequest object to get the context path for the current portlet, and renderRequest is used to encode the entire URL. The FarmerDetailsController.js contains a Dojo JavaScript object that begins like this:
Take note of how the typeof keyword is used to avoid declaring the same JavaScript object more than once. That could happen if the same portlet was placed on a page twice, or if multiple portlets shared common JavaScript files.

Next, an instance of the Controller class is created in the JSP:

```javascript
if (typeof(FarmerDetailsController) == "undefined") {
    dojo.declare("FarmerDetailsController", null, {
        constructor: function (/\*String\*/ windowid, /\*String\*/ namespace, /\*String\*/ resourceUrl) {
            this.windowid = windowid;
            this.namespace = namespace;
            this.resourceUrl = resourceUrl;

            // initialize the portlet window for this portlet instance
            this.portletWindow = new ibm.portal.portlet.PortalWindow(windowid);
        },
        getFarmerDetails: function() {
            this.showFarmerReading();
            var params = this.getFormDataParams();
            this.sendPortletResourceRequest(this.resourceUrl, this.getCallbackFunction(), params, this.namespace);
        },

        // remaining functions removed
    });
}
```

Each variable name is prefixed with `<portlet:namespace/>` that returns a unique namespace for each portlet. This ensures no JavaScript naming collisions if the page contains multiple portlets.

When the C2A framework has populated the target portlet’s input field, the surrounding form is submitted. This triggers the JavaScript method getFarmerDetails on the Controller object, (see listing above).

At a high level, this function performs the following actions:

1. Accesses the DOM to copy the hidden C2A target value (the farmer’s name), from a hidden INPUT to a DIV.
2. Makes visible another DIV that contains the “Loading farmer details” text, and spinning progress icon.
3. Prepares a String of POST parameters, i.e. the farmer’s name which will be posted to the portlet’s serveResource method using Ajax.
4. Calls the sendPortletResourceRequest function to perform the Ajax POST request, providing a callback function.
5. When the serveResource has returned its markup, the browser invokes the callback function which updates the DOM with the returned HTML.
The `sendPortletResourceRequest` function uses Dojo to access the JavaScript XMLHttpRequest object which is used to make asynchronous requests. The Portal JavaScript API also provides an object named `XMLPortletRequest` which is a wrapper for the XMLHttpRequest. However, the `XMLPortletRequest` object should not be used to make requests to a resource URL. The `XMLPortletRequest` should only be used to make requests to Action or Render URLs.

The `sendPortletResourceRequest` function looks like this:

```javascript
sendPortletResourceRequest: function ("String"/ resourceUrl, /*Function*/ callbackFunction, /*String*/ params, /*String*/ namespace)
{
    var request = dojo._xhrObj();
    request.open("POST", resourceUrl, true);
    request.onreadystatechange = function () {
        if (request.readyState == 4) {
            if (request.status == 200) {
                callbackFunction(request.responseText,
                namespace);
            } else {
                this.portletWindow.logError("The request failed");
            }
        }
    };
    request.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
    request.setRequestHeader("Content-length", params.length);
    request.setRequestHeader("Connection", "close");
    request.send(params);
}
```

This function uses Dojo to create an XMLHttpRequest which can be used to address the portlet using Ajax. The open function is called that opens an HTTP connection to the `resourceUrl` originally generated in the JSP. The true parameter indicates the request should be made asynchronously.

Next, the HTTP headers are set to identify the type of the POST parameters and finally the parameters are sent with `request.send`. This triggers the `serveResource` method on the portlet which can retrieve the POST parameters and return appropriate markup or data.

With every change of state, the browser calls the inner function referenced by the `request.onreadystatechange` variable. When the relevant status is met, this indicates the portlet's `serveResource` has returned and the data is available to the browser. The callback method then updates the UI to hide the spinning icon and populate a DIV with the returned markup.

Finally, let's inspect the `serveResource` method on the target portlet:
public void serveResource(ResourceRequest request, ResourceResponse response)
    throws PortletException, IOException {
    String responseText = new String();

    if ("farmerName".equals(request.getResourceID())) {
        String farmerName = request.getParameter("farmerName");
        Farmer selectedFarmer = lookupFarmersDetails(farmerName);

        responseText = createFarmerTable(selectedFarmer);

        // Set current farmer to the session to cater for browser refresh
        FarmersDetailsPortletSessionBean sessionBean = (FarmersDetailsPortletSessionBean) request.getPortletSession().getAttribute("SESSION BEAN");
        sessionBean.setFarmersDetail(selectedFarmer);
    }

    response.setContentType("text/html");
    response.getWriter().println(responseText);
}

It inspects a ‘resourceID’ of the resource request. This is similar to the action name of an Action URL; a means of differentiating requests. Next the farmer’s name is retrieved and this is used to look up a Farmer object and create a HTML table String that is written back to the client.

Also notice that the appropriate Farmer object is also written to the session - this is to facilitate browser refresh. In the JSP, if a Farmer object is found in the session, it is used to output the details. Imagine a scenario where the user triggers a C2A event to initially show farmer details and subsequently presses the browser refresh button. If the Farmer was not stored to the session the details portlet would be blank as only the C2A event triggers the Ajax request to populate the empty portlet. To solve this problem, the information is retrieved from the session and displayed until a further C2A event is received in which case it is replaced.
UI Frameworks

This page describes the primary server side frameworks that can be used for designing a portlet application for IBM WebSphere Portal. The main focus will be frameworks supported by IBM tooling that are based on IBM WebSphere Portlet Factory, Struts and Java Server Faces.

Generally, WebSphere Portal doesn't depend on a particular User Interface (UI) framework. So, Portal developers are always driving hard to live up to the hype that new technologies claim to provide. The fact is that no single framework provides the solution for all requirements. Having a strong knowledge of the various frameworks is important before initiating the development process. Portal specialists are always weighing the merits and demerits of one framework over another framework. WebSphere Portal supports many different frameworks such as Struts, JavaServer Faces, Spring and others.

The portlet development frameworks are based on a portlet/servlet Bridge/Portal Bridge implementation. These bridges comply to portlet APIs such as JSR 168 or JSR 286. Pure servlet based frameworks like Struts are very difficult to use in a Portlet environment because they only support and understand a single request event. For a Portlet, the request processing has to be handled in two separated action and render events. Furthermore, the Portlet specification doesn't allow servlet access during the action request event.

A distinctive feature of the Portal Bridge is that it removes these barriers and thereby allows the mapping of an action request event to the underlying servlet handler implementation.

The importance of frameworks

Development frameworks have evolved over a period of time based on the best practices of enterprise application development. They use well known design patterns and wrap them with appropriate classes that are shipped with the frameworks. Therefore, application developers need not worry about the low level implementation details of the objects and can focus on the high end design aspects such as the required business logic, user interface and so on.

A framework also clearly identifies the discrete areas of focus for the application development. For example, using the well known Model-View-Controller (MVC) architecture, user interface developers can focus on the view aspect of the application using UI-related technologies such as JSP, Javascript and so n, while Java developers can focus on the model do not need to have UI development specific skills.

Moreover, there exists tooling such as IBM Rational Application Developer and IBM WebSphere Portlet Factory that provide easy development with the use of known frameworks. For example, IBM RAD provide wizards for development with the use of popular frameworks and upon using these features of RAD, several key artifacts related to the frameworks are created and wired automatically by the tool. Developers then just need to customize or add additional code to an automatically generated code base created by the tool. This speeds development immensely.

IBM supported frameworks

The following is a list of the primary IBM tools that enable portlet development using the various frameworks:

- Java Server Faces (JSF) 1.1
- JavaEE standard
- IBM Rational Application Developer (RAD) v7.0 / 7.5
- Struts 1.1
- IBM Struts Portlet Framework
- IBM Rational Application Developer (RAD)v7.0 / 7.5
- IBM WebSphere Portlet Factory
WebSphere Portlet Factory

IBM WebSphere Portlet Factory (WPF) provides a development and customization tool of portlets for WebSphere Portal. It is positioned as a development environment, parallel to Rational Application Developer and it offers unique capabilities for developers to develop JSR 168 and JSR 286 compliant portlets.

The tool is based on wizards that developers need to configure to develop various pieces of application logic. These wizards are called builders in the WPF context and they are integrated block by block to fully develop the complete portlet application. Therefore, developers of all skill levels can rapidly build multi-page, complex applications without writing Java code. Java/J2EE skills are not required for developers to develop portlet applications and the engine generates J2EE compliant code behind the scenes.

Secondly, since no code is required to be written, the time to develop the portlets is reduced. Moreover, it becomes very easy to customize these portlets as developers need to tweak the builders or add more builders to get the customized portlets. A single code base creates portlets that run in WebSphere Application Server, WebSphere Portal, Expeditor, and Lotus Notes 8 clients with very minimal changes.

Lastly, high-level design patterns are available in the Portlet Factory during automatic development of code and configuration of builders by end users. It exposes various toolkits for consuming enterprise data from various databases and expose them as services for further consumption. The presentation layer can consume these services to present data through a custom user interface.

Key concepts

The following sections describe some of the key concepts of WebSphere Portlet Factory based on the following figure.

**Builders**

Builders are the building blocks for creation of application elements and artifacts like pages, schemas and data, server-side and client-side actions and logic, backend connectivity, and so on. They have easy to use, wizard-like interfaces, capture design patterns and automate the creation of code. Examples include, a page builder is used to define HTML markup, a select builder for implementing the drop down menu functionality in a user interface, a service definition builder for defining a set of services and so on. New builders are constantly being added in WPF and there is a process for creating your own custom builders as well.

**Models**

Models are containers for aggregating builders. Developers work iteratively with builders in a model to create the presentation as well as service layer logic of the portlet. A model is typically used to create a service or presentation pages. In a typical development scenario, models are also used to organize groups
of builders for implementation of a block of functionality for portlets and these models can be further integrated to achieve the complete portlet.

Profiles
Profiles enable the generation of multiple application variations from a single source model. The values of all variables in WPF can be mapped to select their values for profiles. These profiles can represent variations in roles or groups of users accessing the application, customers, partners, or regions or portlet customization by administrators or end users. For example, a common use case is to expose some variables as a configuration mode of the portlet. This enables users to customize the portlet based on values assigned in the configuration mode. WPF enables this functionality with a builder (called Portlet Adapter builder) and such functionality can be achieved very quickly without requiring additional coding by developers.

Portlet Factory support for advanced capabilities
Portlet Factory enables Web 2.0 characteristics in application development in the following ways:

1. Rich user friendly user interfaces: Portlet Factory provides extensive set of builders for generating highly customized and interactive user experiences by providing Dojo and Ajax capabilities. Here are some examples of Client side processing, thereby reducing server side requests:
   - Client side sorting of table column data
   - Pagination of table data
   - Type-ahead options in the search text box
   - Contextual pop-up windows with additional details as user hovers cursor over UI elements
   - Drag and drop UI elements onto drop zones
   - In-line editing of data

   All of these are handled as a part of builder configuration.

   To see some samples, visit this site: http://www-10.lotus.com/ldd/pfwiki.nsf/dx/ajax-type-ahead-sample-using-database-filtering

2. User participation: Leverage the social aspects of Web 2.0 provided by Lotus Connections for integrating Profiles, Blogs and Dogear in applications.

   Samples can be viewed at: http://www-10.lotus.com/ldd/pfwiki.nsf/dx/ibm-lotus-connections-integration-samples

3. Data access through web services standards: REST and Web Services provide a standard method for accessing data and functions regardless of platform, operating system and programming language. WPF provides support for RSS 1.0/2.0 and ATOM 1.0 feeds. WPF provides support to consume and visualize WS-I 1.1 basic profile web services. It can also generate WS-I 1.1 basic profile web services to expose data from enterprise systems.

   See samples at: http://www-10.lotus.com/ldd/pfwiki.nsf/dx/06182008070049PMWEBUZ3.htm

Integration with other IBM products and backend repositories
One of the best practices for WebSphere Portlet Factory application development is to develop different models for back end data integration (known as service provider or service model) and the front end user interface (known as service consumer or presentation model).

Service Provider: The service provider model focuses on integration with back end enterprise repositories with the help of various out-of-box integration builders such as SQL Call, SAP Function Call, Web Service Call, and others from WPF. The builders generate the Java code for integrating with these data repositories behind the scenes. If any data transformation is needed, we can make this a part of the service Model as well.

The table below shows the key data sources and services that can be integrated with WebSphere Portlet Factory by out-of-box builders.
<table>
<thead>
<tr>
<th>System or Data Source</th>
<th>Builder</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relational databases</td>
<td>SQL Call</td>
</tr>
<tr>
<td>SAP</td>
<td>SAP Function Call</td>
</tr>
<tr>
<td>Siebel</td>
<td>Siebel Business Component</td>
</tr>
<tr>
<td>PeopleSoft</td>
<td>PeopleSoft Component Interface</td>
</tr>
<tr>
<td>IBM Lotus Domino</td>
<td>Domino Data Access</td>
</tr>
<tr>
<td>WSDL/SOAP Web service</td>
<td>Web Service Call or Web Service Multiple Operation</td>
</tr>
<tr>
<td>REST-style service, Lotus Connections, Lotus Quickr, InfoSphere MashupHub</td>
<td>REST Service Call</td>
</tr>
<tr>
<td>IBM WebSphere Process Server tasks</td>
<td>Human Task Processing Action and Human Task Query</td>
</tr>
<tr>
<td>IBM Information Server</td>
<td>IBM Information Service Call</td>
</tr>
<tr>
<td>XML Data</td>
<td>Import to XML</td>
</tr>
<tr>
<td>Java beans</td>
<td>Linked Java Object</td>
</tr>
<tr>
<td>EJB</td>
<td>EJB Call</td>
</tr>
<tr>
<td>Microsoft Excel</td>
<td>Excel Import</td>
</tr>
<tr>
<td>IBM Workplace Web Content Management</td>
<td>Lotus Web Content Management Access</td>
</tr>
<tr>
<td>IBM Lotus Forms</td>
<td>Lotus Forms Launch and Embed</td>
</tr>
</tbody>
</table>

**Service Consumer:** The consumer layer consumes the data returned by the services layer to present to the portal front end. This is shown in the diagram below.
The key advantages of using such a paradigm are:

- Promotes independent development and testing of front end and data layers.
- Automates back end connectivity provided by WebSphere Portlet Factory builders.
- Enables reuse of assets. The service models can be reused by several presentation models as required. Also, the presentation model can be reused to retain a similar user interface pattern, while changing the consuming service model.

Support and documentation

Some key points that may be useful for customers while considering development using WebSphere Portlet Factory include:

1. WebSphere Portlet Factory provides an environment for immediate development of applications. The installer provides Eclipse and WebSphere Application Server Community Edition out of the box. WebSphere Portlet Factory includes an integrated set of tutorials including creating a Web Application Project, creating a simple portlet, creating data-driven portlets, and a quick start with WAS CE.
2. Users can leverage the expertise of the Portlet Factory community. There is a best in class wiki which provides over 50 downloadable samples and techniques. It is one of the best sources of information that developers should utilize. [http://www-10.lotus.com/ldd/pfwiki.nsf](http://www-10.lotus.com/ldd/pfwiki.nsf)
4. WebSphere Portlet Factory 6.1 has increased their platform support to SUSE and Red Hat Enterprise Linux. WPF Designer can also be installed as a plugin into Eclipse 3.3.
5. WebSphere Portlet Factory has increased support to deploy JSR 286 portlets to WebSphere Portal 6.1 and deploy portlets to WAS JSR 168 containers.

Example implementation

The following sample uses the service provider and service consumer paradigm of WPF. This example demonstrates a simple use case of developing a form, submitting a value to the portal server and fetching values from the server, based on the submitted value. Therefore the main functions of service consumer and provider are as follows:

- Service consumer: Form UI, submitting the form input to the server and displaying the result of the form input on the UI.
- Service provider: Accepting the form input and returning data from the back end based on the form input.

**Step 1**

Install WPF. The default installation selection will include a supported version of Eclipse. Alternatively, WebSphere Portlet Factory can be installed into a supported version of one of the following IDE’s:

- IBM Rational Application Developer
- IBM Rational Software Architect
- IBM Rational Web Developer
- Eclipse (Eclipse SDK versions only)

Prior to installation, you must install IBM WebSphere Portal or one of the supported application servers.

For WPF installation information refer to this infocenter link: [http://publib.boulder.ibm.com/infocenter/wpfhelp/v6r1m2/topic/com.bowstreet.designer.doc/install/install.htm](http://publib.boulder.ibm.com/infocenter/wpfhelp/v6r1m2/topic/com.bowstreet.designer.doc/install/install.htm)
**Step 2**
Creation of a WPF project. The next step is to create a WPF project that results in the creation of the portlet application. It may have several portlets and is the first step in the formation of a single application war file.

![Create WebSphere Portlet Factory Project](image)

**Step 3**
Creation of WPF model. The next step is to create a model within the WPF project created above. Models are containers for builders. There are some template models which have typical builders populated to give developers a head start for using the model of a particular template.
Step 4
Builders can be added to a model. Here is the dialog to add a builder call to the current model:
These are some of the builders used in the example:
1. View and Form builder (Service Consumer): Used for calling a backend method and display the result on a page. It also enables creation of input form.
2. Portlet Adapter(Service Consumer): It exposes the model to be used as a portlet.
3. Service Definition (Service Provider): Creates a data service, that can be exposed as a WSDL based web service as well. operation can be added to this service definition.
4. Service Operation (Service Provider): Creates a data operation to be added to the service definition.

The following figure shows a snapshot of the WPF workspace showing the WPF project, models and builders.
1. WPF project - RiverBendSample
2. Consumer model named consumer.model
3. Provider model named data.model
4. Various builders in consumer.model
5. Service consumer builder.
6. Builder editor (This is the wizard like interface to customize builders).

**Step 5**
Execution of the portlet war on a portal server: The Portlet application can be automatically deployed to a pre-configured portal server. Optionally the portlet application can be extracted as a war file and can be deployed to the server through portal administration. [Here is the link to the portlet war.](#)

Below are the screenshots of the demo portlet that is deployed on WebSphere Portal 6.1. The user interface can be customized further. Note that to see the internal structure of the portlet and to customize it, the WPF archive needs to be imported in the WPF. [Here is the link to the portlet archive.](#)
Struts Portlet Framework

The Struts Portlet Framework (SPF) is a WebSphere Portal specific framework built on top of the Struts framework contributed by Apache as part of the Jakarta project. Rational tools provide Struts development tools for both Web applications and portlet applications. The primary goal of SPF is to enable Struts programmers to easily create portlets based on Struts conventions without requiring a great deal of knowledge about portlets. Struts is a popular open source project for implementing Web applications using a Model-View-Controller design pattern. Struts have an active development community and continue to evolve. The user community is equally active and many Web applications have been implemented using Struts. The Struts framework can be used to effectively design the Web application. The framework supports both large and small development teams as well as organizations that need to split the development work between specialists (for example, between user interface designers and programmers). The reasons that Struts has become popular for the servlet environment apply for portlets as well. Developers that have worked with Struts in the servlet environment should adapt easily to the Struts Portlet Framework. The packaging of a Struts portlet application is very similar to a Struts application in the servlet environment. However, WebSphere Portal also introduces additional concepts, such as portlet modes, multiple device support, and portlet communication, that might need to be addressed by the Struts application.

Runtime

Struts Portlet Framework provides modified TLD files so that appropriate classes can be referenced with in the Portlet JSP files. The implementation of these classes resides in additional JAR files included in Struts Portlet Framework. In the WebSphere Portal 6.x distribution of the Struts Portlet Framework, all of the TLD files are now contained in the wps.struts.tlds.common.jar file. You can reference these TLDs in the JSP. These taglibs will be added automatically when you use the Web Page wizard in a Struts Portlet Project containing WebSphere Portal Server v6.x Struts Portlet Framework. For adding additional taglibs, you can use the Import Custom menu option to select the TLDs available in the Page Designer's JSP.
Toolset
The Struts Portlet Framework support in the Rational Software Development Platform simplifies the process of wiring Struts Portlet applications and eliminates the need to manage many of the underlying requirements of Portlet applications.

The Struts Portlet tooling supports development of Portlet applications based on all the JSR 168 API and IBM Portlet API. There are differences in the run-time code included with projects, tag libraries supported, Java class references, and configuration architecture, but unless otherwise noted, these differences are managed by the product tooling.

The following are the list of high level activities involved in developing Struts Portlet applications:

- Create a Struts Portlet Project
- Design the Struts Portlet application, typically using the Web diagram editor.
- Create and modify artifacts that are associated with a Struts Portlet application.

You will use various wizards to generate these artifacts, including Struts Portlet specific JSP and Java files. If you use the Web diagram editor, you can use the standard menu options and drag-and-drop from available palette drawers to create node representations of the artifact that you need. These artifacts are created for you automatically after you enter the name for the node. You will have the option to reference an existing artifact by clicking on the button that is available while you are editing the name.

When you realize a node by putting the name in Edit mode, you can press the button to launch a selection
dialog launching the Create new Object wizard. You can also click **Enter** after you finish editing the name. This will automatically generated the artifact that you need with default options.

- Navigate to related artifacts and view the project in a logical manner, using the project explorer.
- Run and test the Struts Portlet application.
- Set up related preferences, if necessary.
- Validate that the Project is correct.

Rational tools provide a set of wizards that help you to create Struts-Portlet related artifacts. These wizards are the same wizards used to create standard Struts artifacts. Based on the development context, Portlet-specific model options are provided as defaults. However, in some cases you may need to select a model value that specifies Portlet-specific file and code-generation behaviors. To summarize how the wizard behaviors vary between Portlet and non-portlet models, see the below list:

**Action Class wizard**

- Provides support for the enhanced Struts Portlet Framework action class, Struts Action, that hides details of the Struts action class that do not map well to execution Portal run-time.

**Action Mapping wizard**

- Supports the Struts Portlet Framework changes added to the Action class wizard.

**Struts Configuration File wizard**

- Add the required `<controller>` section that specifies the `com.ibm.wps.portlets.struts.WpsRequestProcessor` processor class when creating the configuration file (for an IBM API Portlet). For the JSR 168 API or JSR 286 API Portlet the `com.ibm.portal.struts.portlet.WpsRequestProcessor` processor class is used.

**Struts Module wizard**

- For an IBM Portlet API, the `<init-param>` entry that specifies a module is added under the `WpsStrutsPortlet` servlet entry instead of the `ActionServlet` servlet entry. For a JSR 168 API Portlet, the module is specified in the `Portlet.xml` file as part of the Struts Portlet definition.
- The Struts configuration files specified by modules include the required `<controller>` section.

### Developing a Struts Portlet using Struts Portlet Framework

Struts portlet projects share similar functionalities with standard Portlet and Struts projects, but there are some differences that you should be aware of. The Struts portlet project structure and related resources are dictated by the Struts Portlet Framework support provided by WebSphere Portal and included in the Rational Software Delivery Platform.

You can create Struts portlet projects by using the New Portlet Project wizard. By default Struts-type portlet and a Web diagram file will be added in the process of creating the project. The wizard automatically generates Struts portlet configuration files and the necessary updates to the `web.xml` file, `portal.xml` file, and adds all of the Struts Portlet Framework tag libraries and JAR files to the project, in the directory structure that is required.

Steps to create a Struts Portlet Project using Rational Software Delivery Platform v7.5 include:

1) **Select File -> New -> Project.** In the New Project wizard, select **Portal -> Portlet Project** and then click **Next.** The New Portlet Project wizard opens.

2) Provide values, as appropriate, for the following fields:
• In the Project name give the name for the new portlet project.
• In the Project content section, the location in the file system where the project will be created. Accept the default workspace, or change the file system location by using Browse.
• In the Target runtime section, select the appropriate WebSphere Portal version for your portlet project. Struts portlet is available for WebSphere Portal Server V6.1 target run-times. The SPF version will be the one distributed for WebSphere Portal server V6.1. **Note:** Struts portlets are not supported on the WebSphere Application Server run-times.
• In the EAR Membership section select this check box for Add project to an EAR project if you intend to deploy the Portlet as an EAR module.
• In the EAR project Name section, name of an enterprise application project (EAR project) that the portlet project should be associated with for purposes of deployment. The information used to deploy the portlet project to WebSphere Portal in the test environment will be added in this project. All portlet applications associated with a single EAR project will run on a single session at the test environment. You may separate portlet applications into multiple EAR projects. You should use the same EAR project for other projects that are related.
• In the Portlet settings section select the JSR 168 Portlet for the Portlet API.

To create a portlet with this project, select the Create a portlet checkbox. Specify a name for the portlet and choose Struts for the portlet type.
Click the **Show advanced settings** button to access facet and run-time options using this wizard.  
**Note:** If you want accept the defaults associated with a portlet project, click **Next**, then click **Finish**. If you prefer to customize more project options, click **Next**.

3) In the Portlet Settings page, define the following:

- In the Content types and modes section, specify the content type and its modes.
- Struts portlets always support HTML markup. You can add Compact HTML (chtml) or WML 1.3 (wml) markup support to the project. These markups are not currently supported by Rational Software Delivery Platform.
• Select the appropriate check boxes for any additional modes that you want to enable for the portlet. View mode is always provided by default.
• For Code generation options, generate a custom portlet class is not available for Struts, as the tool does not support creating additional subclasses of a Struts portlet.
• Package prefix - a combination of the default package naming preference plus the project name.
• Class prefix - a combination of the portlet-name suffixed by "Portlet".
• Super class - a base class for the portlet class being generated
• For Locale specific information, choose the default, or click Add to create a new locale. Click Next.

4) Update any Struts portlet-specific settings in the Struts Portlet Settings page. By default, the wizard will create resource bundles. The Java package name default is built using the project name that you specified.

5) Click Finish. A new portlet project is created with a Struts portlet definition in web.xml and portlet.xml.

Note: The following sample configuration and deployment descriptor file snippets show representative tagging used within Struts Portlet (JSR 168 API based) projects:

**struts-config.xml**

The following controller section is added to all struts configuration files:

```xml
<controller processorClass="com.ibm.portal.struts.portlet.WpRequestProcessor "/>
</controller>
```

**web.xml**

This is an example of additional tag libraries to be added to the web.xml file:

```xml
<taglib>
<taglib-uri>/WEB-INF/struts-html.tld</taglib-uri>
<taglib-location>/WEB-INF/struts-html.tld</taglib-location>
</taglib>
```
portlet.xml

This shows a sample Struts portlet added to the portlet.xml file:

```xml
<portlet>
  <portlet-name>MyFirstStrutsPortlet</portlet-name>
  <display-name>MyFirstStrutsPortlet</display-name>
  <portlet-class>com.ibm.portal.struts.portlet.MyFirstStrutsPortlet</portlet-class>
  <!-- Struts configuration -->
  <init-param>
    <param-name>config</param-name>
    <param-value>/WEB-INF/struts-config.xml</param-value>
  </init-param>
  <init-param>
    <param-name>struts-servlet-mapping</param-name>
    <param-value>*.do</param-value>
  </init-param>
  <!-- end of Struts configuration -->
  <!-- Here is an example of edit mode initialization: -->
  <init-param>
    <param-name>config/html/edit</param-name>
    <param-value>/WEB-INF/struts-html-edit.xml</param-value>
  </init-param>
  <!-- The following snippets are generated by the portlet API. -->
  <expiration-cache>0</expiration-cache>
  <supports>
    <mime-type>text/html</mime-type>
    <portlet-mode>view</portlet-mode>
    <portlet-mode>edit</portlet-mode>
  </supports>
  <supported-locale>en</supported-locale>
  <portlet-info>
    <title>MyFirstStrutsPortlet</title>
  </portlet-info>
  <portlet-preferences>
    <!-- example of setting the first page of the html view mode -->
    <preference>
      <name>com.ibm.struts.portal.page.view.html</name>
      <value>index.jsp</value>
    </preference>
    <!-- example of setting the first page of the html edit mode -->
    <preference>
      <name>com.ibm.struts.portal.page.edit.html</name>
      <value>html/edit/index.jsp</value>
    </preference>
  </portlet-preferences>
</portlet>
```
Demonstration
The following example uses the same scenario that was used to develop the sample using WPF. Here are the screenshots of the portlet that was developed using RAD 7.5 using the Struts Portlet Framework. The war file can be imported into RAD to see the file structure and the source code. Here is a link to the sample war file.

Struts 1.x
Struts 1.x is a controller based framework. It is not a standard one. It is recommended for legacy Struts Applications. Struts 1.x do not have the following features:

- Built-in UI component model
- Built-in event model for UI components
- Built-in state management for UI components
- Built-in support of multiple renderers

Struts 2.0
The Strut 2.0 framework is designed for the compilation of the entire development cycle including building, developing and maintaining the whole application. It is very extensible as each class of the framework is based on an Interface and all the base classes are given an extra application and even you can add your own. The basic platform requirements are Servlet API 2.4, JSP API 2.0 and Java 5. Some of the key concepts of Struts 2.0 are described below.

Jsri68Dispatcher
The Controller portlet that handles all the requests to the portlet application. It is Similar to ActionServlet in Struts 1.0 framework but adjusted to a portal environment. The portlet is configured through the portlet.xml descriptor file.
**Action**
Action is a Combination of ActionForm, Action and Model class. Jsr168Dispatcher will pass the request to configured Action to handle request. This is also referred as a location in framework where you write your business logic.

**Interceptors**
Interceptors are one of the most powerful features of Struts 2.0 framework. It is similar to Filters in servlet environment, they allow you to execute your code before and after Action. In the Portlet application if you have a common functionality like setting the parameters and the workflow for validation can be implemented as interceptor.

**Value stack**
The Value Stack is a set of Objects. It is used for carrying data from the Action class to JSP page. By default the Action instance is pushed on the top of the stack, because of this you can access the action instance object, along with all the HTTP objects (session, request, application, context, and so on).

The following sequence diagram shows the request flow for forwarding Action.

The following sequence diagram shows the request flow for how the forward is happening to a JSP page.
Struts also supporting different view technologies like Freemarker, Velocity, and Java Server Pages. So it will give a flexibility to choose any view technologies where the developers have the skills. It also provides additional features like type conversion and a validation framework.

**Spring Framework integration**

In Struts many popular optional features of the particular framework are distributed as plug-ins. Integrating the Spring framework with Struts allows the Actions, Interceptors and Views to be created by Spring. The Objects created by Struts can be auto wired by Spring. It provides two interceptors that auto wire actions if not using the SpringObjectFactory. We can do the auto wiring by adding the StrutsSpringObjectFactory entry in the struts.properties file.

```
struts.objectFactory = org.apache.struts2.spring.StrutsSpringObjectFactory
```

We need to keep the entries as below in the struts-config.xml and applicationContext.xml file to map the class attribute of `<action>` tag with the id attribute of the `<bean>` tag.

**Struts-config.xml**

```xml
<action name= "myAction" class="myContact">
    <result>/WEB-INF/jsp/list.jsp</result>
</action>
```

**applicationContext.xml**

```xml
<bean id="myContact" class="com.strutssample.action.MyAction">
    <property name="contactDAO" ref="contactDAO"></property>
</bean>
```
Java Server Faces Framework

At its core, JavaServer Faces is a standard Java framework for building user interfaces for Web applications. Its key advantage is that it simplifies the development of the user interface, which is often one of the more difficult and tedious parts of Web application development. Although it is possible to build user interfaces by using foundational Java Web technologies (such as Java servlets and JavaServer Pages) without a comprehensive framework designed for enterprise Web application development, these core technologies can often lead to a variety of development and maintenance problems. JavaServer Faces avoids these problems by offering a robust, "best of breed" framework with well-established development patterns, built upon the experience of many pre-existing Java Web development frameworks.

JavaServer Faces is designed to simplify the development of user interfaces for Java Web applications in the following ways:

- Provides a component-centric, client-independent development approach to building Web user interfaces, thus improving developer productivity and ease of use.
- Simplifies the access and management of application data from the Web user interface.
- Automatically manages the user interface state between multiple requests and multiple clients in a simple and unobtrusive manner.
- Supplies a development framework that is friendly to a diverse developer audience with different skill sets.

Beyond these specifics, JSF offers another important benefit. It takes the best elements found through years of experience in Web application development and combines them into a single, comprehensive, and standard API for building Java Web application user interfaces. Furthermore, it brings unprecedented ease and productivity without sacrificing power and flexibility to J2EE Web application development.

JSF features and benefits

The following list describes the key features and benefits of using JSF for Web application design and development:

- Standards-based Web application framework:
  - JSF is a standards-based Web application framework. JavaServer Faces technology is the result of the Java Community process JSR-127 and evolved from Struts. JSF addresses more of the model-view-controller pattern than Struts, in that it more strongly addresses the view or presentation layer through UI components, and addresses the model through managed beans. Although JSF is an emerging technology and is likely become a dominant standard, Struts is still widely used.
  - JSF is targeted at Web developers with little knowledge of Java and eliminates much of the hand coding involved in integrating Web applications with back-end systems.
- Event driven architecture:
  - JSF provides server-side rich UI components that respond to client events.
- User interface development:
  - UI components are de-coupled from their rendering. This allows for other technologies such as WML to be used (for example, mobile devices).
  - JSF allows direct binding of user interface (UI) components to model data.
  - Developers can use extensive libraries of prebuilt UI components that provide both basic and advanced Web functionality.
- Session and object management:
  - JSF manages designated model data objects by handling their initialization, persistence over the request cycle, and cleanup.
- Validation and error feedback:
  - JSF allows direct binding of reusable validators to UI components. The framework also provides a queue mechanism to simplify error and message feedback to the application user. These messages can be associated with specific UI components.
- Internationalization:
  - JSF provides tools for internationalizing Web applications, including supporting number, currency, time, and date formatting, and externalization of UI strings.
**JSF application architecture**
The JSF application architecture can be easily extended in a variety of ways to suit the requirements of your particular application. You can develop custom components, renderers, validators, and other JSF objects and register them with the JSF run-time. The focus of this section is to highlight the JSF application architecture depicted in the figure below.

- **Faces**: JSP pages are built from JSF components, where each component is represented by a server-side class.
- **Faces servlet**: One servlet (FacesServlet) controls the execution flow.
- **Configuration file**: An XML file (faces-config.xml) that contains the navigation rules between the JSPs, validators, and managed beans.
- **Tag libraries**: The JSF components are implemented in tag libraries.
- **Validators**: Java classes to validate the content of JSF components, for example, to validate user input.
- **Managed beans**: JavaBeans defined in the configuration file to hold the data from JSF components. Managed beans represent the data model and are passed between business logic and user interface. JSF moves the data between managed beans and user interface components.
- **Events**: Java code executed in the server for events (for example, a push button). Event handling is used to pass managed beans to business logic.

Rational Application Developer allows developers to easily use JSF components for portlet development. It enables rapid application development and standards-based code generation. The following is a list of JSF features that you can use to develop portlets in Rational Application developer:

- Visual page layout of JSF components using Page Designer
- Web Diagram Editor for defining the flow of a JSF application
Lotus Redbooks Wiki – Creating External Facing Web Sites with WebSphere Portal

- Built-in Component Property editor
- Built-in tools to simplify and automate event handling
- Page navigation defined declaratively
- Automatic code generation for data validation, formatting, and CRUD functions for data access
- Multiple faces configuration file support for different purposes, such as navigation and managed beans
- Relational database support
- EJB support
- Web services support
- Data abstraction objects for easy data connectivity using JPA
- Data objects can be bound easily to user interface components
- Support for runtime page templates with Tiles
- Creating Portlet project with JSF support
- Customizable property templates for data types (such as Date)
- 3rd party library integration with customizable palette
- Custom component library building
- Integration with JPA to easily create Web applications with persistence

Rational Application Developer has additional rich and powerful components for JSF. These components include:

- Menu Bar (displays a menu bar of buttons and/or hyperlinks)
- Panel - Dialog (creates a block panel that behaves like a modal or modeless dialog box)
- Panel - Form Box (creates a block panel that contains a header area and one or more form label/field pairs)
- Panel - Section (creates a block panel that has a header that can be used to expand/collapse the display of the panel's content)
- Select - Color (displays a drop-down combo box from which the user chooses a color)
- Select - Calendar (adds small calendar to the page)
- Progress Bar (displays an animated progress bar)
- Link - Request (generates an HTML link with a URL that can pass parameters and navigate to a page by passing a string to JSF navigation rules)
- Data Iterator (iterates over rows of model data allowing values from each row to be used in child components).

One of the most elegant design aspects of the JavaServer Faces specification is that it completely relies on existing J2EE Web technology at its foundation. This means that a JSF application is really just a standard J2EE Web application with a few specific configurations. These are:

- An entry in the Web application's web.xml file, that enables the Faces Controller servlet when a certain URL pattern is specified, such as /faces/*.
- A JSF configuration file, faces-config.xml, that allows for configuration of all elements of a JSF application. This file is treated as a peer of the web.xml file and is usually located in the Web application's WEB-INF/ directory.
- A WEB-INF directory with the following Java libraries:
  - The actual JSF libraries: jsf-api.jar and jsf-impl.jar.
  - Additional Apache "Commons" libraries: commons-beanutils.jar, commons-collections.jar, commons-digester.jar, and commons-logging.jar. Although not part of the core JSF technology, these libraries are relied upon by JSF and are thus required to be in the application's WEB-INF/lib directory.
  - JSTL jar files: jstl.jar and standard.jar.

Once a portlet is properly configured for JSF, you can construct the View using, but not limited to,
JavaServer Pages. Building JSF applications with JavaServer Pages is done by using JSF-enabled JSP tag libraries. For a JSP page to be JSF-enabled, it must first contain JSF JSP taglib directives provided by a JSF implementation. The following taglib directives are for the Core and HTML libraries from Sun's reference implementation:

```jsp
<%@ taglib uri="http://java.sun.com/jsf/core" prefix="f"%>
<%@ taglib uri="http://java.sun.com/jsf/html" prefix="h"%>
```

In the body of the portlet JSP, you must then add a `<f:view>` tag. This will become the base UI component of what will become a component tree in memory on the server when the page is requested for viewing. If the page processes form input, as opposed to just displaying output, you'll need to add a `<h:form>` tag as a child of the `<f:view>` tag. Subsequent children tags of the `<h:form>` tag will become the form elements such as `<h:inputText>`, that renders an input field, and `<h:commandButton>`, that renders a form submission button.

To understand how JavaServer Faces creates and manages a server-side tree of components in memory that directly corresponds to the components included in a page, consider the following JSF-enabled portlet JSP page:

```jsp
<%@ page contentType="text/html"%>
<%@ taglib uri="http://java.sun.com/jsf/core" prefix="f"%>
<%@ taglib uri="http://java.sun.com/jsf/html" prefix="h"%>
<f:view>
<html>
<body>
<h:form >
<h2>
A Simple Portlet JSF Page
</h2>
<h:inputText value="#{portletBean.portletName}"/>
<h:commandButton value="List Portlets"/>
</h:form>
</body>
</html>
</f:view>
```

When a JSF-enabled JSP page is requested or when the user invokes an action on a UI component in a JSF-enabled JSP page, it is important to understand the exact sequence of events that occur on the server in order to fulfill the request to view or submit a JSF page. The sequence of events that are triggered during requests to JSF pages is known as the JSF request processing lifecycle or sometimes simply as the JSF lifecycle. This is shown in below figure.

---

Like Struts, JSF follows a Model-View-Controller design paradigm. Recall that an MVC application is segmented into three distinct application components:

- The Model, that contains the business logic or non-UI code.
- The View, that is all the code necessary to present a UI to the user.
- The Controller, that is a front-end agent that directly handles the user's requests and dispatches the appropriate view.

These three elements, also depicted in the figure below, combine to produce an architecture that yields distinct, separately maintainable code.

![Architecture Diagram]

The JSF Navigation Model is an elegant solution for keeping track of all navigations in the entire JSF application. This greatly improves the manageability of the application because it is easier to maintain a central navigation model rather than having to update multiple page links in multiple pages. The central location of the navigation model in an XML file is also very "tool friendly" in that Rational Application Developer offers visual ways to easily define JSF navigation models.

The navigation model is based on a set of "navigation rules," that define a "from" page (from-view-id) and one or many "to" navigation cases. Each navigation case has an associated "outcome" and "to" page (to-view-id). For example, to navigate from page1 to page2 when the outcome "success" is encountered, the following rule is specified in the faces-config.xml:

```xml
<navigation-rule>
  <from-view-id>/portletPage1.jsp</from-view-id>
  <navigation-case>
    <from-outcome>success</from-outcome>
    <to-view-id>/portletPage2.jsp</to-view-id>
  </navigation-case>
</navigation-rule>
```

A second navigation case can be defined for a "failure" outcome that will route the viewer to portletPage3.jsp.

```xml
<navigation-rule>
  <from-view-id>/portletPage1.jsp</from-view-id>
  <navigation-case>
    <from-outcome>success</from-outcome>
    <to-view-id>/portletPage2.jsp</to-view-id>
  </navigation-case>
  <navigation-case>
    <from-outcome>failure</from-outcome>
    <to-view-id>/portletPage3.jsp</to-view-id>
  </navigation-case>
</navigation-rule>
```

**JSF advantages and disadvantages**

Advantages and disadvantages are sometimes based on the perspective of the individual or the functional requirements. From a portlet development perspective the main advantages are:

- JSF offers a clean separation between behavior and presentation.
The separation of logic from presentation also allows each member of a portlet development team to focus on his or her piece of the development process, and it provides a simple programming model to link the pieces. It removes the whole notion of having to process incoming Web requests. Instead, the Web developer can rely on the JSF lifecycle to handle back-end plumbing automatically and can use the JSF event model to jump in and do custom processing only when needed.

- Code reusability is high.
- It is easy to upgrade.
- It has very extensible component infrastructure.
- Developing complex portlets including navigation, complex forms, validations is very easy.
- It has embedded plugin set in Rational Application Developer.

Disadvantages are:

- It adds overhead to developing simple portlets.
- The learning phase is not short. You need to create your own Ajax based components or you can use open source JSF based frameworks like icefaces.

**Spring 2.0**

Spring also supports JSR-168 and JSR-286 API Portlet development similarly servlet-based Web development. The Portlet MVC framework is a mirror image of the Web MVC framework, and also uses the same view abstractions and integration technology. Most other portlet MVC frameworks try to hide the two phases from the developer and make it look much like traditional servlet development. It will make you to think this approach removes one of the main benefits of using portlets. So, the separation of the two phases is preserved throughout the Spring Portlet MVC framework.

This framework is designed around a DispatcherPortlet that dispatches requests to handlers, with configurable handler mappings and view resolution, just as the DispatcherServlet in the web framework does. The Portlet MVC does not support the local resolution and theme resolution. These two areas are in the scope of the portal/portlet-container and are not appropriate at the Spring level. Regardless, all mechanisms in Spring that depend on the locale will still function properly because DispatcherPortlet exposes the current locale in the same way as DispatcherServlet.

**Model**

Spring Portlet MVC supports beans whose lifecycle is scoped to the current HTTP request or HTTP Session. This is not a specific feature of Spring Portlet MVC itself, but rather of the WebApplicationContext container(s) that Spring Portlet MVC uses.

**View**

All the view rendering capabilities of the servlet framework are used directly via a special bridge servlet named ViewRendererServlet. By using this servlet, the portlet request is converted into a servlet request and the view can be rendered using the entire normal servlet infrastructure. This means all the existing renderers, such as JSP, can still be used within the portlet.

**Controller**

The default handler is a very simple Controller interface; it offers just these two methods:

- void handleActionRequest(request, response)
- ModelAndView handleRenderRequest(request, response)

The framework also includes most of the same controller implementation hierarchy, such as AbstractController, SimpleFormController, and so on. Apart from these the Data binding, command object usage, model handling, and view resolution are all the same as in the servlet framework.
The DispatcherPortlet
This is a controller portlet that handles all requests to the portlet application similar to ActionPortlet or ActionServlet in the Struts framework. Portlet MVC framework designed around a portlet that dispatches requests to controllers and offers other functionality facilitating the development of portlet applications. The DispatcherPortlet, does more than just that. This controller portlet is completely integrated with the Spring ApplicationContext and allows using every other features of the Spring.

To process the requests and render the appropriate views DispatcherPortlet using few special beans. The Spring framework includes these beans and these can be configured in the WebApplicationContext, just like any other bean would be configured. During the process of the request exceptions might be thrown. These exceptions can be picked up by any of the handler exception resolvers that are declared in the WebApplicationContext. So, by using these exception resolvers we can define custom behavior in case such exceptions get thrown. We can also customize the Spring's DispatcherPortlet by adding context parameters in the portlet.xml file or portlet init-parameters.

The ViewRendererServlet
In Portlet MVC the rendering process is a quite more complex than Web MVC. You must convert the PortletRequest and PortletResponse to HttpServletRequest and HttpServletResponse and then call the render method of the View in order to reuse all the view technologies from Spring Web MVC. To achieve this, DispatcherPortlet uses a special servlet called ViewRendererServlet.

Controllers
This is similar to the Action class in Struts. The action request and the render request are the two methods that handle the two phases of a portlet request by Portlet Controller interface. The action phase should be capable of handling an action request and the render phase should be capable of handling a render request and returning an appropriate model and view. Spring Portlet MVC provides a several controllers like ParameterizableViewController, SimpleFormController and AbstractWizardController which already contains lot of the functionality you might need.

Handler Mappings
The Handler Mapping will decide which Controller needs to be called. There are several out-of-the-box implementation available for this. We can map the incoming portlet requests to appropriate handlers with the help of handler mapping. A Handler can be any Object that can handle portlet requests. The Controllers are an example of Handlers, and they are the default handlers mapping. The DispatcherPortlet will hand it over the request to the handler mapping to allow inspect the request and come up with an appropriate HandlerExecutionChain. After that DispatcherPortlet will execute the handler and interceptors in the chain. The concept of configurable handler mappings that can optionally contain interceptors will be very powerful.

The most commonly used handler mapping in Spring Portlet MVC are PortletModeHandlerMapping, ParameterHandlerMapping and PortletModeParameterHandlerMapping.

ViewResolver
The Spring Portlet MVC uses the same ViewResolver implementations. If the portal expect the result of rendering the portlet to be a HTML fragment then using JSP, Freemarker, Velocity and XSLT make sense. But the views that return other document type will not make sense in the portal context. And you can not use a relative URL to access the other resources because in portlet you have no idea what the current URL looks like.

The example sequence diagram below shows how the Render works in Spring Portlet MVC
Developing a Spring Portlet using Spring MVC framework
You can create Spring Portlet projects by using the New Portlet Project wizard in Rational Software Delivery Platform. By default Rational Software Delivery Platform will not provide any specific wizard to develop the Spring Portlets, so you need to create an empty JSR 286 Portlet and on top that you need to customize the basic project structure by updating the deployment descriptor files and adding the tag libraries, JAR files to the project.

Steps to create a Spring Portlet Project using Rational Software Delivery Platform v7.5 include:

1. Open the Rational Software Delivery Platform v7.5 and create a new Portlet Project as per the screen shot below.
2. Create an Empty JSR 286 Portlet Project.
Default directory structure
3. You need to modify the directory structure as like the below screen shot.
You can use the resources which is available from the attachment section.

4. Update the web.xml file with below content.

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE web-app PUBLIC
    "-//Sun Microsystems, Inc.//DTD Web Application
    2.3//EN"
    "http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>
    <display-name>River Bend Coffee Company Portlet Application</display-name>
    <description>River Bend Coffee Company Portlet Application</description>
    <listener>
        <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
    </listener>
    <servlet>
        <servlet-name>ViewRendererServlet</servlet-name>
        <servlet-class>org.springframework.web.servlet.ViewRendererServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>ViewRendererServlet</servlet-name>
        <url-pattern>/WEB-INF/servlet/view</url-pattern>
    </servlet-mapping>
    <taglib>
        <taglib-uri>/tags/struts-bean</taglib-uri>
        <taglib-location>/WEB-INF/tld/struts-bean.tld</taglib-location>
    </taglib>
    <taglib>
        <taglib-uri>/tags/struts-html</taglib-uri>
        <taglib-location>/WEB-INF/tags/struts-html.tld</taglib-location>
    </taglib>
    <taglib>
        <taglib-uri>/tags/struts-logic</taglib-uri>
        <taglib-location>/WEB-INF/tags/struts-logic.tld</taglib-location>
    </taglib>
    <taglib>
        <taglib-uri>/tags/struts-nested</taglib-uri>
        <taglib-location>/WEB-INF/tags/struts-nested.tld</taglib-location>
    </taglib>
    <taglib>
        <taglib-uri>/tags/struts-tiles</taglib-uri>
        <taglib-location>/WEB-INF/tags/struts-tiles.tld</taglib-location>
    </taglib>
</web-app>
```

5. Update the portlet.xml file with below content.
Demonstration

Here are the screenshots of the Spring Portlet that was developed using Rational Software Delivery Platform v7.5. The war file which is attached can be imported into Rational Software Delivery Platform v7.5 to see the file structure and the source code.

River Bend Coffee Company - Spring MVC Portlet

<table>
<thead>
<tr>
<th>Farmers List</th>
</tr>
</thead>
<tbody>
<tr>
<td>Farmer</td>
</tr>
<tr>
<td>Marcelle Diver</td>
</tr>
<tr>
<td>Mathius Bramly</td>
</tr>
<tr>
<td>Sebastian Bravo</td>
</tr>
</tbody>
</table>
Adobe Flex

JSR-168 Portlet has support for using Adobe Flex. Similarly it will also support the JSR-286 API with some additional features like support for asynchronous requests, public render parameters and events. Adobe Flex is an Open Source UI framework that allows creating SWF files using XML markup and Action Script. We can also integrate the Flex controls in the Themes and Skins of WebSphere Portal. Java developers will easily become familiar with Flex Builder. The Flex Builder is the development environment used for this. The SWF file generated by the mxmlc compiler will be rendered on the client side by the Flash Player plug-in that is available in most web browsers.

To embed the Flex Control in to the application, we have to use data exchange interfaces such as Ajax bridges, HTTP services and SOAP services.
For more information and an example of using Adobe Flex, see the following link: [Using Adobe Flex in JSR-286 Portlets](http://www-10.lotus.com/ldd/portalwiki.nsf/dx/17.09.2008050832WEBCQV.htm)

**Deciding on the right framework**

The frameworks described above are designed to let projects quickly implement the business logic of an application without worrying about the underlying plumbing of the application. All three provide similar functionality although they differ in the implementation. Both Jade and Struts are application server-independent, while WebSphere Portal simply supports WebSphere Application Server.

For a useful article on choosing the right framework, see the IBM developerWorks article: [Modeling WebSphere Portal Portlets with UML -- Part 2](http://www.ibm.com/developerworks/websphere/library/techarticles/0211_bernal/bernal.html)
Personalization

Personalization involves using technology to accommodate the differences between individuals. Web pages may be personalized based on the characteristics of an individual. Personalization implies that the changes are based on implicit data, such as items purchased or pages viewed. The term customization is used instead when the site only uses explicit data such as ratings or preferences.

Web personalization models include rules-based filtering, based on “if this, then that” rules processing, and collaborative filtering, that serves relevant material to customers by combining their own personal preferences with the preferences of like-minded others. Collaborative filtering works well for books, music, video, and so on.

The benefits of personalization include:

1. Focused content delivery based on the interests of the Web site visitor.
2. Increase in site usage by the visitor, that may lead to more sales.
3. Customers get what they want to see and keep coming back to the site.
4. Enterprises get an opportunity for more focused campaign and marketing by targeting the right audience.

Some examples of personalization include:

1. Recommended products based on previous activity by visitors.
2. Self service pages where users login to a site and see content meant for them.
3. Marketing campaigns such as personalized e-mail.
4. Portal sites such as MyYahoo and MyGoogle.

Personalization components

Portal personalization provides automatic customization of Portal-based web sites for individual users and user groups.

A Portal personalization engine can be used for the following tasks:

- Recognize a specific user based on a profile or determine characteristics of a user based on previous purchases, products or pages viewed, and so forth.
- Select content that is appropriate for each profile, for example:
- If a person has a high salary range, personalization can be configured to retrieve information about a commercial Web site’s premium products.
- If an individual belongs to a particular geographic region, content specific to that region may be targeted to the individual.
- Assemble pages with the proper personalized information so the user sees her personalized page.

Portal personalization engine (often referred as PZN) is composed of five components:

Personalization browser

The Personalization user interface is a Portal based application where authorized users:

- Register resource collections
- Author rules, campaigns, and content spots
- Map rules into content spots for a particular campaign
Rules engine
The rules engine evaluates rules created in the Personalization browser.
In a Portal environment, there are different ways to invoke rules:

- Personalization rules may be invoked through the Personalized List portlet and presented as a list of text and links
- Web Content Management (WCM) Personalization components allow you to invoke a selection rule or content spot and format the results by using WCM tags
- Hide rules associated with pages or portlets through Portal administration are automatically triggered when a user navigates a Portal page

LikeMinds recommendation engine
Personalization contains a dynamic recommendation system based on LikeMinds. LikeMinds analyzes user interactions that occur on your Web site and generates real time predictions and recommendations to your Web site users.

Resource engine
The resource engine resolves the queries produced by rules into content pieces to be returned. Content for personalization is created using whatever content management tool you choose, or may come from an LDAP or any other database. Content is accessed via a set of Resource Collection classes.

Logging framework
The logging framework is used to record information about Web site usage to the feedback database and the recommendation engine. It is entirely up to the site developers to decide what information is logged.

Four approaches to personalization
Some approaches to personalization are described in the following sub-sections.

Security
A site usually filters content based on the access rights of visitors. For example, if a site visitor is in the Human Resources department, the site provides access to content containing Human Resources policy manuals.

User preferences
The user pulls relevant content from the system by telling what categories are the most important for him. The system will save that preferences to drive content lists in the future.

**Rules engine**
The site owner defines a set of business rules that determine what category of content is shown when a user with a certain profile type visits the site. An example would be: Display all four wheel drive SUVs to visitors in the northeast in the 21 to 35 age group. This approach has the advantage of driving the site's behavior with the business objectives of the site owner.

**Collaborative filtering**
A site visitor rates a selection of products, explicitly or implicitly. Those ratings are compared with the ratings offered by other visitors. Software algorithms detect similarities. For example, a visitor receives book recommendations based on the similar purchases of others.

**Rules driven personalization**
Rules are used to define how your Web site interacts with individual and groups of Web site visitors. Rules are composed of easy-to-read logic statements that, in their final form, specify how to evaluate various conditions and what actions to take based on those conditions.

WebSphere Portal supports several kinds of rules:

**Profiling**

**Function**
Create dynamic profiles based on user attributes, date, browser capabilities or any other information managed by PZN

**Sample**
Selection

Function
Selects a list of content from

- Lotus Web Content Management SQL
- LDAP
- Atom
- Custom

Sample
**Binding**

**Function**
Pulls all the other rule types together:

- Conditions come from profiles
- If conditions are met, select or recommend content, perform updates, send emails.
**Update**

**Function**  
Updates the current user object or the state.  
Updates may be persisted as with the current user or  
Updates may be transient or application managed such as a shopping cart  
May also set portlet or HTTP request and portlet or HTTP session attributes

**Sample**
Email

Function

Ability to trigger email from a page
Personalized email body
Email may run rules to select content for the email
Email campaigning using a scheduled email rule
Examples: targeted newsletters, emails triggered to verify or send receipt of user actions

Sample
Content spot

Function
Creates a virtual content area that will be managed by a personalization campaign. The campaign will define which content is displayed in this area by binding the content spot with different selection rules.

Sample

---

[Image of Personalization Editor interface]

**Description**
This is the content spot to show tea and coffee campaigns.

**Output Type**
WcmContentResource

**Default Mapping**
Campaign1/SelectTeaContent

---

An e-mail action will send a personalized e-mail to a set of users which is defined by personalization separately for each user.
Campaigns

Function
Campaigns define time or frequency constraints to bind rules to spots.

Sample

![Personalization Navigator](image)

Recommendation

Function
Allow recommended content based on user navigation, item affinity or user preferences. Requires the user to first make calls to the feedback or LikeMinds API to log transactions.

Sample
Visibility
Function
Allows for showing or hiding pages and portlets depending on the result of a rule.

Sample
**Extending personalization**

There are a series of articles in IBM developerWorks that shows how to extend WebSphere Portal Personalization capabilities to make data from external sources available to your portal using Personalization rules:

<table>
<thead>
<tr>
<th>Part</th>
<th>Description</th>
<th>URL</th>
</tr>
</thead>
</table>

**Sample scenarios**

This section shows the implementation steps of two personalization scenarios. The WCM artifacts for the RiverBend Web site are reused. All personalization artifacts, such as business rules, content spots, and so on are created in the personalization section of WP. They are organized in two folders, named as "Campaign1" and "Campaign2" for personalization scenarios 1 and 2 respectively.
Personalization scenario 1
In this scenario, a campaign for recommended products is run for a fixed duration in the "Recommended products" portlet in the home page. There are two sets of content related to coffee and tea products. Their display needs to be divided in an equal ratio. Therefore, for one request to the Web site, content 1 is displayed and for the next request, content 2 is displayed. Here are the implementation steps for Personalization scenario 1.

Step 1: Prepare two categories one for Tea and one for coffee. These categories are used to classify content.

Step 2: Profile the content with the two categories. For example, we profiled content titled "Earl grey" with category tea and "Coffee cioccolato" with category coffee. These are stored in the location "River Bend/Beverages/Teas" and "River Bend/Beverages/Coffee" as shown in the figures below.
To profile a content with a particular category, open the content in the WCM authoring portlet in edit mode, select profile section and choose the particular category to profile the content.

Step 3: Prepare selection rules: We go the personalization section of WP, go to the folder “Campaign 1”. We create a new selection rule "SelectTeaContent" to select content titled "Earl grey". Similarly, we create a new selection rule "SelectCoffeeContent" to select content titled "Coffee cioccolato".
Step 4: We create a content spot "ContentSpot_teacoffee" with default rule binding as "SelectTeaContent".

Step 5: We create a campaign "Campaign_teacoffee" whose validity date is from Nov 10, 2009 to Nov 12, 2010. We ensured that the current date falls within the validity period.
Step 6: We create a rule mapping to map the campaign with a content spot and rules. We have assigned a split of 50 to give each rule mapping an equal weight to display. Therefore, 50% of the time, rule mapping 1 would be executed and 50% of the time, rule mapping 2 would be executed.

Step 7: We create a personalization component in the WCM authoring portlet and map it to the content spot "ContentSpot_teacoffee". The component uses WCM custom tags to render the menu search result.
Step 8: We render the personalization component into the portlet named "Recommended products" in the homepage.

On first request, here is the view of the portlet.

Design for each menu search result:
Lotus Redbooks Wiki – Creating External Facing Web Sites with WebSphere Portal

Recommended product

Coffee cioccolato

Indulgent we know, but coffee and chocolate go together so well.

Indulge in both your vices at the same time, a 'Cafe Cioccolato' is just what the Dr. ordered. 80% pure dark chocolate flakes and and coffee ground together with a touch of whipped cream. A perfect date.

Logout and login to the portlet again. Now the view of the portlet changes to show the second campaign.
On first request, here is the view of the portlet. Logout and login to the portlet again. Now the view of the portlet changes to show the second campaign.

**Personalization scenario 2**
In this scenario, we are going to show personalized content based on location of the registered users. Based on a location attribute filled in on the registration page of the registered user, the location specific content is delivered in the "Campaign" portlet in the homepage.

Step1: Lets add the location attribute in the "Edit my profile" page for users and let us populate this attribute.
In this example, we populate the countryName of the registered users with countries either India or Spain.

Step 2: Verification of appropriate categories in WCM: Contents are segregated for two countries - India and Spain. Therefore we ensure that there are two categories in the WCM authoring portlet.
Step 3: We have country specific content in the site area "River Bend/Company/Where we are":

We need to assign content related to India and Spain titled "India Bangalore" and "Spain La Ramblas" to the two categories created in Step 2.

Step 4: Creation of profile rules: This is used to profile users based on their countyName attribute in their profile. As shown in the figure, the profile of users is "visitor_India" when the countyName attribute of the users is India.
Step 5: Creation of select content rules: This rule selects content for a specific country. For example, as shown in the figure, the “SelectSpainContent” selects content whose category is “Spain.”
Step 6: Creation of binding rules: This rule binds the profile rule (created in step 4) with the select rules (created in step 5).
Step 7: Creation of Content Spot: As shown in the figure below, the default mapping of this content spot is the binding rule, created in step 6.
Step 8: Creation of Personalization component: Personalization component maps the content spot, created in step 7. The component uses WCM custom tags to render the menu search result.
Step 10: Viewing in WCM rendering portlet:

We add a "WCM Content Viewer" portlet to the home page of the RiverBend Web site. We then map the personalization component, created in step 9, to the rendering portlet.
Here is the screenshot of the portlet seen by the user from Spain.

We then sign up a new user with CountryName attribute as India. Upon logging in by that user, here is the view of the portlet.
**Personalization demonstration sample**

This section includes steps to download and execute the sample code used in the development of this page.

**Step 1:** Import the personalization artifacts. Here is a link to more detailed information on importing artifacts: [http://publib.boulder.ibm.com/infocenter/wpdoc/v6r0/topic/com.ibm.wp.ent.doc/wpf/pzn_stage_prod.html](http://publib.boulder.ibm.com/infocenter/wpdoc/v6r0/topic/com.ibm.wp.ent.doc/wpf/pzn_stage_prod.html)

Links to sample files:

- Campaign1.nodes
- Campaign2.nodes

**Step 2:** Import the WCM library named "RiverBend". Click [HERE](http://publib.boulder.ibm.com/infocenter/wpdoc/v6r0/topic/com.ibm.wp.ent.doc/wpf/pzn_stage_prod.html) for more information on importing libraries.

Link to sample file:

- JCR_export.zip

**Step 3:** Change the portal theme. Unzip the file to the location `\installedApps\Enhanced_Theme.ear\wp.theme.enhancedtheme.war` Please take a backup of the previous themes folder.

Link to sample file:
themes.zip

4. Import the portal page through Import XML in portal administration.

Link to sample file:

pageExport.xml

5. Navigate to the link "RiverBend" in the portal theme as shown in the screenshot below. The RiverBend portal page should be up and running.

Summary

This page has provided a brief overview of various personalization techniques and considerations. We have also provided two simple examples of setting up customized content through WebSphere portal.
Search
This page covers various topics related to search. There are two primary areas of discussion. The first covers the challenges and possible solutions for having content in the portal successfully show up on Internet search engines. The second is on integrating search engine capabilities and results within a Portal-based Web site.

Search engine optimization
In the past, one of the challenges for external facing Portals was ensuring the public pages appear as expected in the search results of Internet search services such as Google or Yahoo. Some optimization of the Portal is required to achieve this.

Overview of Search Engine Optimization
Search engine optimization (SEO) is defined in Wikipedia as the process of improving the volume or quality of traffic to a web site from search engines via "natural" or un-paid ("organic" or "algorithmic") search results as opposed to search engine marketing (SEM) which deals with paid inclusion.

Having key marketing Web sites appear in search engine results has become a sub-discipline of web site design. Actually a very mature one, so there are many documents, tools, and search engine optimization (SEO) sites to help sites do well in search engines.

This huge amount of information is very helpful, but in some cases can complicate things, especially if you just want to start looking from scratch how to optimize an existing site.

If you have an accessible and usable site, you are already on your way. If not, this quick overview will help by providing a comprehensive list of SEO techniques taken from some of the best online resources and combining it with some advice about how to implement them in WebSphere Portal.

Definition of terms
The following terms are important to understand. These definitions are from the following article:

Directory - A directory is a human-compiled search. Most directories rely on submissions instead of spiders.

Keywords, keyterms, and keyphrases - The words you want your Web site to rank well for in search engine results pages (SERPs).

Link farm - In SEO, a link farm is a page full of links that have very little to do with each other and exist just as links without any real context. People use link farms to increase the number of links to a page in hopes of fooling a search engine, such as Google, into thinking the page is more link-worthy than it actually is.

Organic listings - Organic listings are the free listings in the SERPs. SEO for organic listings usually involves improving the actual content of your Web site, often at the page or infrastructure level.

PageRank - PageRank is a measurement that the Google-obsessed use to test their rankings in Google. SEO and search engine marketing (SEM) professionals also use the term to describe your ranking in the SERPs and the ranking algorithm points given to your site by Google. No matter how you define it, PageRank is an important part of your SEO success.

Paid listing - Like the name implies, paid listings are paid for in search engines. Depending on the search engine, a paid listing can mean paying for inclusion in the index, pay per click (PPC), a sponsored link, or other ways of making your site show up in the SERPs for targeted keywords and phrases.

Ranking - A ranking is where your page is listed in the SERPs for your targeted keywords. The goal of SEO is high rankings for the keywords that your Web pages target.

Ranking algorithm - A ranking algorithm is the set of rules that a search engine uses to evaluate and rank the listings in its index. The ranking algorithm is what determines which results are relevant to a specific query.
Search engine optimization (SEO) - SEO involves creating Web pages that are picked up by the search engines through optimizing your content for search engine attractiveness and visibility. SEO is mostly used to increase the rankings of your organic listings. We'll use the term SEO to describe the techniques I recommend, although many of these techniques also fall under the umbrella of SEM.

Search engine results page (SERP) - SERPs are the listings, or results, displayed for a particular search. SERP is sometimes defined as search engine results placement. For the purposes of this article, we'll refer to it as a page rather than a placement. In the world of SEO, a good showing in the SERPs is what it's all about.

Spamming - Spamming is a method of SEO that attempts to trick a spider and scam loopholes in the ranking algorithm to influence rankings for targeted keywords. Spamming can take many forms, but the most simple definition for spam is any technique a Web site uses to misrepresent itself and influence ranking. The two methods of SEO are based on whether you want to spam or not.

- Black hat SEO: Spamming the search engines. Black hat SEO is lying, cheating, and stealing your way to the top of the SERPs.
- White hat SEO: Optimizing your site so it serves the user, as well as attracts spiders. In white hat SEO, anything that leads to a good user experience is considered also good for SEO.

Spider - A spider crawls through the Web looking for listings to add to a search engine index. It is sometimes referred to as a Webcrawler, robot, or bot. When optimizing your page for organic listings, you are catering to the spider.

How search engines work
This information is based on content from http://www.seomoz.org/files/articles/beginners-guide-to-search-engine-optimization.sxw.

Search engines collect information about the information published on the web and build a huge database that relates pages with the terms they contain through a four steps process:

1. Crawling the Web
Search engines run scheduled processes called "bots" or "spiders" that use the hyperlinks found on the Web to "crawl" pages and documents.

2. Indexing Documents
Once a page has been crawled, its contents are analyzed and keywords are extracted to build a database of documents that makes up a search engine's "index".

3. Processing Queries
When a request for information comes into the search engine, the engine retrieves from its index all the document that match the query.

4. Ranking Results
Once the search engine has determined which results are a match for the query, an algorithm runs calculations on each of the results to determine which is most relevant to the given query. They sort these on the results pages in order from most relevant to least so that users can make a choice about which to select.

More on Google
Google is one of the most popular search engines. Understanding a bit more about Google can be very helpful in SEO.

Google’s search index accounts for a majority of the entire search-related traffic so to start by optimizing your site for Google can make a lot of sense. Google ranks sites by link analysis; if Google isn't led to your site by other sites to be indexed, Google might never give you a high ranking.
Google optimization basics
The key to ranking well in Google is optimizing the visible keywords on a page.

A successful keyword strategy has two steps:

- **Keyword selection**: Determine which words your potential audience might use to search for your page and create keywords based on those words.
- **Keyword optimization**: Apply these keywords to the appropriate pages (3 - 5 keywords per page is the recommended amount) and optimize them from the top left, and then down. Often this will be the first 200 words on your page -- title tag, headings, abstract, and such.

Users will initially view your Web site the same way the spider does, so emphasizing the keywords from the top-left-down is a good Web design practice as well.

**Other factors that will affect your ranking**
Organic SEO goes beyond a good keyword optimization strategy. This table presents other optimization techniques extracted from Google's Search Engine Optimization Starter Guide Version 1.1 (http://www.google.com/webmasters/docs/search-engine-optimization-starter-guide.pdf)

<table>
<thead>
<tr>
<th>SEO goals and tasks</th>
<th>Portal tasks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create unique, accurate page titles:</td>
<td></td>
</tr>
<tr>
<td>• Accurately describe the page's content</td>
<td>Use Portal pages metadata to allow end users to enter meaningful description meta tags</td>
</tr>
<tr>
<td>Create unique title tags for each page</td>
<td></td>
</tr>
<tr>
<td>• Use brief, but descriptive titles</td>
<td></td>
</tr>
<tr>
<td>Make use of the &quot;description&quot; meta tag</td>
<td></td>
</tr>
<tr>
<td>• Accurately summarize the page's content</td>
<td>Use friendly URLs and URLs mappings to create semantic URLs to your pages</td>
</tr>
<tr>
<td>• Use unique descriptions for each page</td>
<td></td>
</tr>
<tr>
<td>Improve the structure of your URLs</td>
<td></td>
</tr>
<tr>
<td>• Use words in URLs</td>
<td>Use your page hierarchy as simple as possible in Portal</td>
</tr>
<tr>
<td>• Create a simple directory structure</td>
<td>Keep your page hierarchy as simple as possible in Portal</td>
</tr>
<tr>
<td>• Provide one version of a URL to reach a document</td>
<td>Keep your page hierarchy as simple as possible in Portal</td>
</tr>
<tr>
<td>Make your site easier to navigate:</td>
<td></td>
</tr>
<tr>
<td>• Create a naturally flowing hierarchy</td>
<td>Incorporate the breadcrumb component to your themes</td>
</tr>
<tr>
<td>• Use mostly text for navigation</td>
<td></td>
</tr>
<tr>
<td>• Use &quot;breadcrumb&quot; navigation</td>
<td></td>
</tr>
<tr>
<td>Offer quality content and services</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Use portlets and WCM to create dynamic content and keep it updated.</td>
</tr>
<tr>
<td></td>
<td>Define validation rules for content elements in WCM to ensure that content is properly</td>
</tr>
<tr>
<td><strong>Write better anchor text</strong></td>
<td>Use validation rules in WCM to ensure link text is relevant. Provide contextual help in authoring templates to remind content contributors of the importance of keep doing it.</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>--------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Use heading tags appropriately:</strong></td>
<td>Define guidelines for using heading tags in portlets and WCM templates and share them with your developers</td>
</tr>
<tr>
<td>• Imagine you're writing an outline</td>
<td></td>
</tr>
<tr>
<td>• Use headings sparingly across the page</td>
<td></td>
</tr>
<tr>
<td><strong>Optimize your use of images:</strong></td>
<td>Use validation rules in WCM to validate the use of alt text in images and restrict the file types for images. Use Ephox accessibility functions to ensure that images and media content include alt text.</td>
</tr>
<tr>
<td>• Use brief, but descriptive filenames and alt text</td>
<td></td>
</tr>
<tr>
<td>• Supply alt text when using images as links</td>
<td></td>
</tr>
<tr>
<td>• Store images in a directory of their own</td>
<td></td>
</tr>
<tr>
<td>• Use commonly supported filetypes</td>
<td></td>
</tr>
<tr>
<td><strong>Make use of free webmaster tools:</strong></td>
<td>Follow the instructions provided in this chapter to configure and use Portal sitemap portlet Review your portal site with external tools to be sure that everything is indexed as expected</td>
</tr>
<tr>
<td>• See which parts of a site Googlebot had problems crawling</td>
<td></td>
</tr>
<tr>
<td>• Upload an XML Sitemap file</td>
<td></td>
</tr>
<tr>
<td>• Analyze and generate robots.txt files</td>
<td></td>
</tr>
<tr>
<td>• Identify issues with title and description meta tags</td>
<td></td>
</tr>
<tr>
<td>• Understand the top searches used to reach a site</td>
<td></td>
</tr>
<tr>
<td>• Get a glimpse at how Googlebot sees pages</td>
<td></td>
</tr>
<tr>
<td>• Remove unwanted sitelinks that Google may use in results</td>
<td></td>
</tr>
<tr>
<td>• Receive notification of quality guideline violations and file for a site reconsideration</td>
<td></td>
</tr>
</tbody>
</table>

**Sitemaps**
The following information is based on content from [http://www.sitemaps.org/](http://www.sitemaps.org/)

Sitemaps are an easy way for webmasters to inform search engines about pages on their sites that are available for crawling.

In its simplest form, a Sitemap is an XML file that lists URLs for a site along with additional metadata about each URL (when it was last updated, how often it usually changes, and how important it is, relative to other URLs in the site) so that search engines can more intelligently crawl the site.
Web crawlers usually discover pages from links within the site and from other sites. Sitemaps supplement this data to allow crawlers that support Sitemaps to pick up all URLs in the Sitemap and learn about those URLs using the associated metadata.

Using the Sitemap protocol does not guarantee that web pages are included in search engines, but provides hints for web crawlers to do a better job of crawling your site.

WebSphere Portal provides a sitemap portlet to assist in meeting the best practices recommended by most of the internet search engines. You can find further information about Portal sitemap portlet in the Sitemap Portlet section of this chapter.

**How Internet search engines work**

When requesting that an Internet search service includes results from a Web site, typically the Web site administrator will specify the URL of the home page or a site map and the search service will crawl or traverse the hyperlinks, indexing information from each page. Typically only the hyperlinks are followed and information stored in JavaScript functions or HTML meta data is ignored by the search engine crawler. The search engine crawler is sometimes referred to as a 'robot'.

**Challenges with portal crawlability**

When the Internet search crawlers encounter a Portal site, the nature of the Portal URLs is not what they are expecting. The problem is that Portal encodes information into the URL called ‘navigational state’ as a string of encrypted characters. The navigational state contains information about the state of the portal, for example, current page and theme template in use. It also contains information about portlet state, for example, portlet mode (edit, view, help), window state (minimized, maximized), and render parameters. The main reason for including all this information in the URL is to support a bookmarking of Portal pages, maintaining the exact layout and view information at that time. The exact problem for search engine crawlers is illustrated by the following figure.
The URL to any page contains information about the source page, so in the illustration URL-A and URL-D point to the same page, yet they have different URLs. This means that a Portal with say one hundred pages can actually involve more like one thousand unique URLs. Search engine crawlers expect that a URL to a page will be unique. When crawling Portal, the crawler will typically encounter more unique URLs than it deems reasonable, give up and terminate the crawl. Depending on the search engine, this results in none or only some of the Portal pages being included in the search engine’s index meaning the Portal site is difficult to find in Internet search results.

**Portal crawler awareness and normalized URLs**

To address the challenges of crawling Portal, it is now ‘crawler aware’. When a crawler visits a web site, it identifies itself using a ‘user agent’ string in the HTTP header, just as Web browsers do enabling the Web site to determine the browser type. Portal is pre-configured to recognize around fifty common search engines and more can be added.

When Portal recognizes that it is being visited by a crawler, it automatically adjusts the URLs for each page to be ‘normalized’. This means the Portal filters out most of the information normally stored in the navigational state and only the mandatory information required to display the page will remain. With the normalized URLs, if there are links to the same page from different source pages, they will now be the same. If the crawler encounters a link to a page for a second time, it will now be able to identify that the page has already been visited and move on to the next one. Additionally, Portal removes any ‘action URLs’ from the pages when being crawled. This is ensures there are no HTTP POST actions that could perform potentially undesirable operations, such as a link saying “Delete this document”. Even though the normalized URLs may still contain some navigational state information and be quite lengthy, there is no problem for internet search engines to crawl this type of URL.

**Sitemap portlet**

The sitemap portlet is provided by Portal to assist in meeting the best practices recommended by most of the Internet search engines. They suggest either pointing the crawler directly at a site map, or at least having a link to the site map somewhere on the Web site’s home page. In this way, the crawler will be sure to index the most important pages before it reaches the finite limit of pages that most search engines will adhere to. The best practice for using Portal’s site map portlet would be to place a link to it near the top of the page, for example inside the Portal theme.

As the generated site map is likely to contain all the pages and content that should be crawled, additional information called robot directives can be added to a page instructing the crawler not to crawl anything more than the site map page. The robots directive is meta data included in the HTML, or a special robots.txt file, that gives instructions to the crawler defining if crawling sections of the site is allowed or disallowed. The semantics of the robots directives rely on URLs describing the site in a structured way. For example, there might be an instruction to allow /home/public/* but have an exception by disallowing /home/public/employees. Portal URLs are less well structured and although they can start in a ‘friendly’ structured way (for bookmarking), after clicking further links the URL they become more complex and unstructured. For this reason, the best practice for using robot directives with Portal is to place them as HTML elements in the theme, with logic to allow or disallow individual pages. For example, if the Portal has a site map, the theme would output this:

```
<meta name="robots" content="noindex,follow">
```

This would ensure the crawler followed all links from the site map page, but did not include the actual sitemap page in the results. For all other pages, the theme would output:

```
<meta name="robots" content="index,nofollow">
```

This would ensure the pages were added to the index, but links not followed, as the crawler will have already done this via the sitemap page instead.

**External crawlability of Portal with WCM content**

If the Portal includes WCM content, extra consideration is required to ensure all content is indexed and the results show the content in the correct context, that is, on the correct Portal page. If the relationship between
content and pages is one to one, that is, each page shows exactly one piece of content; the sitemap portlet and robots directive approach detailed previously will be very effective. However, if the page includes components to select further content (such as a WCM navigator), the sitemap approach with the robots directives described previously will not work well.

When a WCM navigator is used with a content rendering portlet, the URL to the Portal page contains a request parameter that the rendering portlet uses to determine which piece of content to display. To index all content referenced by the WCM Navigator, the crawler would be visiting the same page many times, each time with a different WCM request parameter. The robot directives discussed in the previous section would prevent this and should therefore not be used in this scenario.

If using a WCM navigator or any other component that relies on the WCM request parameter, it is necessary to reconfigure the way Portal presents normalized URLs. As discussed previously, when Portal detects it is being visited by a search engine crawler, it adjusts the links in every page to remove navigational state, and links that could trigger actions. It also removes request parameters and without them the WCM rendering portlet would simply show its default content. To counter this problem, the extent of URL normalization can be customized; this will be demonstrated in the case study later on this page. More information can be found in the Portal InfoCenter:

Web 2.0 and search
Since version 6.1, Portal has provided a client side aggregation theme called PortalWeb2. Because of the significant use of JavaScript in this theme, the crawlability of the Portal will be compromised. Please see the Web 2.0 chapter of this Redwiki for more information.

Case study using the sitemap portlet
The following figures containing example screens illustrate the steps to configure and add the sitemap portlet and robots directive to an externally facing Portal.

The sitemap portlet lists links up to a maximum figure per page and then moves to the next page. The maximum links value should be configured according to the recommendations of Internet search engines, and according to the typical number of links on the Portal's pages. Typically, this will be between seventy-five and two hundred. The value can be configured as follows:

Use Portlet Management->Portlets and search for wps.p.Sitemap. Click the wrench icon to configure the parameter:
In a default Portal 6.1.5 install, the sitemap can be found on the site map page. It looks like this:

<table>
<thead>
<tr>
<th>Preference</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXPORT</td>
<td>false</td>
</tr>
<tr>
<td>MAX_LINKS</td>
<td>75</td>
</tr>
<tr>
<td>portlet_icon_name</td>
<td>sitemap.gif</td>
</tr>
<tr>
<td>portlet_icon_path</td>
<td>/images/palette/portlets/</td>
</tr>
</tbody>
</table>
If it is acceptable to make the site map visible to users via the Portal’s navigation, the sitemap portlet can be placed on any public page and this would be the crawl starting point for the Internet search engines. If however it is more preferable to hide the site map from the navigation, there are two similar options:

- Place the sitemap portlet on a separate page, and omit that page from the portal navigation. This specific page should be used as the crawler starting URL.
- Place a link to the hidden sitemap page in the theme of the portal home page. Now the regular Portal homepage can be used as the crawler starting URL.

For this case study, both methods will be illustrated.

**Adding sitemap portlet to a page outside of the default navigation**
Use the manage pages portlet to create a new page under the "Context Root".
Assign the new page a unique name and friendly URL name:
Add the sitemap portlet to the new page by clicking the pencil icon, searching for portlet "wps.p.Sitemap" and placing on the page.

Edit the access permissions for the page to allow anonymous access:
Now edit the access permissions for the sitemap portlet so it can be viewed by anonymous users. Locate the sitemap portlet using Portlet Management->portlets and search for "wps.p.Sitemap". Click the key icon and permit anonymous users to perform the "User" role:

If you exit the administration pages and click Portal home, the Travel Site Map page will be visible in the navigation:
To hide the page from the navigation, a property of the page must be amended. This is only possible by editing the page’s XML properties. Use the manage pages portlet to locate the Travel Site Map page, click the export button and save the XML definition to the filesystem:

Edit the exported XML file and add a new piece of meta data that will instruct Portal to omit this page from the navigation (you will see existing <parameter> elements for other properties):

```xml
<parameter name="com.ibm.portal.Hidden" type="string" update="set"><![CDATA[true]]></parameter>
```

The modified XML file must now be imported using the XMLaccess command line tool, replacing the values in <> with the values for your Portal:

```bash
```

Now the Travel Site Map page no longer appears in the navigation and it can now only be viewed via the friendly URL specified when the page was created. In this case, the URL to the page for an anonymous user is http://luxor.hursley.ibm.com:10040/wps/portal/travelsitemap and this URL could be provided to the Internet search engines as the starting point for their crawling.

Note that in a default Portal installation, there are no other pages visible to anonymous users so to test the site map page, add some additional pages under the "Home" label and allow all anonymous users to view
them:

Now to an anonymous user, the Travel Site Map page looks like this:

Adding sitemap portlet to the theme
If it is more desirable to use the Portal's normal home page as the starting point for the Internet search engines to crawl from, a link to the public site map page can be added to the Default.jsp of the theme:

```html
<portal-navigation:urlGeneration contentNode="ibm.portal.Travel Site Map" />
```

This will render an invisible link that the crawler can follow from the Portal web site's home page. Note the contentNode value relates to the unique name of the site map page made previously. The site map page can be either included or excluded from the Portal navigation.
Configuring robots directives
As discussed previously, it is good practice to use robot directives to force the Internet search crawlers to the site map page, and disallow any further access to the Portal to avoid any unnecessary duplication of page crawling. In Portal, the most effective way of doing this is to add some logic to the theme that emits the following meta data if the current page is the site map:

```xml
<meta name="robots" content="noindex,follow"/>
```

To do this, create a custom theme if the page currently uses the Portal default. Then edit the theme’s Default.jsp and add the following statement:

```xml
<portal-logic:if selection="ibm.portal.Travel Site Map">
  <meta name="robots" content="noindex,follow">
</portal-logic:if>
```

Similar statements should be added for other pages included in the site map:

```xml
<portal-logic:if selection="ibm.portal.Vacations Home Page">
  <meta name="robots" content="index,nofollow">
</portal-logic:if>
```

Alternatively, for a more dynamic solution the robots information could be stored to each page as Portal meta data, in a similar way to how the site map page was made invisible previously in this case study. Then a different Portal tag can be added to the Default.jsp to extract this information as HTML meta data. As a starting point, this statement would render all meta data:

```xml
<portal-logic:pageMetaData varname="pageMetaData">
  <table>
    <tr><th>Name</th><th>Value</th></tr>
    <c:forEach var="metaItem" items="${pageMetaData}"
      <tr><td>${metaItem.key}</td><td>${metaItem.value}</td></tr>
    </c:forEach>
  </table>
</portal-logic:pageMetaData>
```

Reconfiguring normalized URLs to include request parameters
As discussed previously, if the Portal uses WCM components that rely on a rendering request parameter, it is necessary to redefine the default behavior of Portal when it normalizes URLs to ensure the parameters are not removed. By default, Portal ships two XSL transformation files which illustrate the available options for URL normalization. By default, Portal uses the UrlNormalization_MIN.xsl file that means the minimum amount of state information is left in the URL. The example configuration files are packaged in a jar file and its location varies between Portal versions. For v6.0 see this IBM technote:


For v6.1 the location of the jar file is:

```xml
<WP_ROOT>basewp.engine.implsharedapp
```

The UrlNormalization_MIN.xsl file should be extracted from the jar and renamed. For example to UrlNormalization_CUSTOM.xsl. The new file should be placed in the following directory:

```xml
<WP_ROOT>sharedappcomibmwpsstateoutputmediators
```

The UrlNormalization_CUSTOM.xsl should be modified to add the sections highlighted below:
Finally, we need to configure the Portal State Manager Service to reference the new transformation file. Open the WebSphere Administration Console for the Portal installation and navigate to:

Resource environment providers > WP StateManagerService > Custom properties

Add or amend property "com.ibm.wps.state.outputmediators.OutputMediatorFactory.normalization_xsl_file" and set the value to UrlNormalization_CUSTOM.xsl. Do not use the full path to the file or URI format, simply enter the filename. Restart the Portal.

Testing the new setting is possible using browser tools to customize the user agent string, for example 'User Agent Pick' is suitable for Internet Explorer 8:

http://www.enhanceie.com/ietoys/uapick.asp

The user agent string for Google is "Googlebot/2.1 (+http://www.google.com/bot.html)". By setting the user
agent string to that of an Internet search crawler, you will be able to confirm that Portal URLs are normalized to a greater or lesser extent according to the transformations specified in your XSL customization file. Also note that if you change the user agent string in this way, you will not be able to log in as Portal will not permit a crawler to visit pages that require authentication.

**Searching and crawling portal and other sites with portal search**

Many Portals use a combination of external search engines and Portal search to help their users find the information they are looking for. Recall that Internet search services will only include the pages of the Portal that do not require authentication. Using Portal search enables users to find content via standard search portlets or the search box in theme. The Portal search results can include the secured content that Internet search will not display, and additional search results from other internal or external systems that may compliment the original search.

**Understanding portal search**

Portal search consists of a variety of portlets to administer search activities and display search results. In addition, there is a core search service that includes a variety of crawlers including Web site, Portal site, WCM and seedlist (for Quickr and other content types). During crawler processing, document filters are used to interpret more than 250 document formats. A categorizer organizes the results based on rules, and a summarizer generates a synopsis to be displayed in search results. The crawlers can be scheduled periodically and their output is a search collection, also known as an index file. See the following figure.

Using an appropriate crawler is important. You may wonder why it is not appropriate to simply crawl for all content using a Web crawler. The reason is Web crawlers do not cater for additional meta data, such as user access rights associated with some content sources. Portal pages, portlets and WCM content is constrained using Portal access controls, for example, LDAP groups if the Portal has been configured to use an external directory. The Portal, WCM and seedlist crawlers work differently than the Wweb site crawler to ensure this additional security information is included in the search collection, thus ensuring that Portal users are not presented with search results they are not permitted to see. Note that the concept of seedlists is discussed in more detail later.

**Portal search architecture**

The Portal search core service can be configured to run on the local Portal, or the workload can be delegated to one or more dedicated servers. Note that when running a local Portal search engine, vertical clustering must not be used. If there were two instances of Portal running on the same hardware, the Portal search engine would run in both nodes and attempt to write to a single index file on the filesystem, resulting
in file deadlocks and an incomplete file.

When using remote search architecture, a Web application is installed to a remote WebSphere Application Server and accessed by one or more Portal nodes. In a clustered Portal environment, remote search must be used otherwise individual nodes might return different search results; see the figure below for an illustration of remote search in a clustered environment:

A choice of two communication protocols is available; EJB or SOAP, which have differences related to security. Portal will always maintain two types of security regardless of EJB or SOAP protocol:

- Collection level security - used to associate collections with sets of authorized users. Only authorized users can search in a collection.
- Document level security - ensures that users have proper authorizations on pages and portlets before search results are presented to them.

These two levels of security are enforced on the Portal server (where the Portal access controls are known). However, when using SOAP over HTTP, an unauthorized user could bypass Portal and access the remote Web service directly on the SOAP port. They could receive an unfiltered list of search results that contain a document summary, although they could not view the actual document or page. This is not a trivial task but it is theoretically possible. With EJB it is possible to completely secure the remote search application from unauthorized access.

In a clustered Portal environment, typically multiple nodes will access a remote search server. It is also possible to configure load balancing between multiple remote search servers to further share workload or for redundancy and failover. It is also possible to have multiple but unrelated remote search servers, perhaps to separate search related administrative tasks between organizational units. For more information on configuring high availability for Portal search, see this white paper:


**Search services collections and scopes**

These three concepts are illustrated in the figure below.
The figure above shows the relationships between search services, collections and scopes.

A search service is an instance of the Portal search engine that performs the crawling and document processing described previously. The manage search portlet is used to configure either local or remote search service(s).

Search collections are the result of crawling activities by the search services. Each search service may be responsible for one or more collections, and each collection is built or added to by a number of content sources. The content sources are the crawlers, for example, WCM or Portal crawlers. The collection is represented by a single file on the filesystem known as the index. For example, on a travel web site there could be a collection for the ‘latest holidays’ that contains a Web site content source scheduled to be re-crawled daily. Another collection for the ‘destinations guides’ could contain a WCM content source that is re-crawled weekly. The search collections are configured via the manage search portlet regardless of the search service performing the crawling. The configuration of search collections can be exported to automatically configure a collection on another search services if required.

Search scopes can be configured to categorize search results. This gives the user an opportunity to filter according to pre-defined categories, for example to only show search results for a particular holiday destination. A search scope can be configured to include only results from a particular search collection, or from several search collections. In addition, query text can be configured to include only results that match a certain string. For example, in a travel Web site the ‘destination guides’ search scope might be configured to include all results from the ‘European destination guides’ and ‘Americas destination guides’ search collections relying on a search scope query string ‘About Your Destination’.

**Search and administration user interfaces**

The search portlets provided by Portal are summarized below and some will be explored in more detail in the case study later in this section.

**Search center**

This portlet is used to display search results from multiple Portal search collections or WebSphere OmniFind (IBM’s enterprise search product). The portlet uses Web 2.0 features such as ‘type ahead’ in the search box.
to provide an up-to-date user experience.

The user can also enter a query in the small search box from the Portal theme, and on submitting the search they will be directed to a page showing the search center portlet.

The results are shown, together with the summary and ranking for each. In addition, a drop down selector of search scopes allows the user to filter the results. For example, by default there is a search scope that only shows WCM results. Search scopes can be customized using the manage search portlet. Recall that a scope can be tied to a single search collection, or span multiple collections. The figure below shows the search center portlet.

![Search Center Portlet](image)

### 3 results found in **All Sources** for **Authoring**.

<table>
<thead>
<tr>
<th>Relevance</th>
<th>Title</th>
<th>Person</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Welcome - Information Portlet</strong></td>
<td></td>
<td>10/19/09 8:19 PM</td>
</tr>
<tr>
<td></td>
<td>Summary: Display an information message contained in a JSP file</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Web Content Management - Web Content Authoring</strong></td>
<td></td>
<td>10/19/09 8:21 PM</td>
</tr>
<tr>
<td></td>
<td>Summary: Display an information message contained in a JSP file</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Getting Started - Information Portlet</strong></td>
<td></td>
<td>10/19/09 8:19 PM</td>
</tr>
<tr>
<td></td>
<td>Summary: Display an information message contained in a JSP file</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The search center portlet offers various points of customization by editing the JSPs, for more information see this developerWorks article:

The following figure illustrates the extent of customizations that is possible.
Search and browse

The search and browse portlet allows users to perform more advanced searches compared with the search center portlet. For example, users can specify multiple search conditions and fields, or browse all available results. This portlet allows searches to be made on only one search collection at a time and it does not benefit from the latest Web 2.0 features as the search center portlet does. It is preinstalled but not deployed in the default Portal installation. Its portlet parameters must be configured before use.
Suggested links
This portlet can be used to configure the display of recommended search results based on keywords entered in the search request. Administrators can manually map key sources of information/documents to search terms, and deliver priority results to users. The results are displayed alongside search results from other search portlets.

This portlet is explored in more detail in the case study of this chapter.

External links
The portlet can be used to display the search results obtained from an internal or external search service that provides results as a feed. The results are displayed alongside search results from other search portlets. The search engine service must provide a public Web-interface and return the search result as either an RSS or Atom feed. Regular HTML results pages cannot be rendered within the external links portlet.

This portlet is explored in more detail in the case study of this chapter.
<table>
<thead>
<tr>
<th>Relevance</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>Gettire Started - Information Portlet</td>
</tr>
<tr>
<td></td>
<td><strong>Summary:</strong> Display an information message contained in a JSP file</td>
</tr>
<tr>
<td>5</td>
<td>Login - IBM Lotus Quickr</td>
</tr>
<tr>
<td></td>
<td><strong>Summary:</strong> The fastest way to share everyday content across connected teams. The goal for Quickr is to provide a fast, easy-to-use platform for users to experiment and prototype with the latest versions of the Quickr product.</td>
</tr>
<tr>
<td>5</td>
<td>Login - IBM Lotus Quickr</td>
</tr>
<tr>
<td></td>
<td><strong>Summary:</strong> The fastest way to share everyday content across connected teams. The goal for Quickr is to provide a fast, easy-to-use platform for users to experiment and prototype with the latest versions of the Quickr product.</td>
</tr>
<tr>
<td>5</td>
<td>IBM Web Content Management - Lotus</td>
</tr>
<tr>
<td></td>
<td><strong>Description:</strong> Web Content Management software from IBM Lotus offers end-to-end Web Content Management capabilities. Web content can be created (using a WYSIWYG rich text editor), managed and published to multiple Web sites.</td>
</tr>
<tr>
<td>5</td>
<td>Welcome - Information Portlet</td>
</tr>
<tr>
<td></td>
<td><strong>Summary:</strong> Display an information message contained in a JSP file</td>
</tr>
<tr>
<td>5</td>
<td>Public Communities</td>
</tr>
</tbody>
</table>
|           | **Summary:** A tag is a keyword that you assign to a community to categorize it and make it easy to find. Tenders for a community can be set to open (anyone can see and join), moderated (anyone can see, but you restrict access to specific members), or restricted (only visible to members). Community of German Lotus Tech Sales, Competitive Team A
Manage search
This portlet allows the configuration and management of search services, collections and scopes.

Manage Search

Choose from the links below to create and manage Search Services, Search Collections, or Search Scopes, and subsequently, the search collections and content sources contained within. Select Search Collections to display the collections that appear in the Search Center.

- **Search Services**
  - Create and modify search service types

- **Search Collections**
  - Create and modify collections of searchable content

- **Search Scopes**
  - Create and modify search scope to deploy to users

Portal Search Toolbox
These portlets can be found on developerWorks and are not included with the Portal product.


They can be used to experiment with providing the most adventurous search scenarios that go beyond the supplied search center. The portlets are supplied with source code that can provide the basis for a spoke search solution. Since this toolbox was published, some of the portlets, such as suggested links and external links, have evolved and benefited from formal testing and are now included with Portal.

WCM search component
This is not a portlet but a WCM component that allows search results to be embedded in WCM content or templates. The component is configured to present results for a Portal search collection. It is a good candidate if the entire Portal is based on WCM content, simply to have all elements of the page constructed from one source.

Portal search APIs
Portal provides the Search and Index API (SIAPI) that can be used to develop custom search portlets to perform search and index operations. This API was used to create the portlets from the Portal Search Toolbox discussed previously. The SIAPI is common to both Portal and WebSphere OmniFind.

In addition to SIAPI, a RESTful interface to Portal search is available that can return search results. A RESTful request could be made over HTTP via a browser and a variety of parameters are available. An example of a simple query is:

http://www.<hostname>:<port>/searchfeed/myportal/search?query=testresults=10

Seedlist framework
As previously mentioned, the Portal, WCM and seedlist content sources work differently with Web site content sources in order to consume additional meta data such as user access rights. When configuring a content source to crawl WCM or Portal content, a seedlist is automatically generated by Portal. The seedlist differs from a sitemap but is complementary. Recall that when the Portal is being crawled by an Internet search engine, the sitemap portlet is often used to generate a site map. The sitemap produces an XML document in a standard format for consumption by most Internet web crawlers. A seedlist is like an extension to a site map and is based on the ATOM syndication format [RFC4287]. The need to develop a single format emerged from the following challenges:

- Search engines cannot develop crawlers fast enough to keep pace with the proliferation of new internal content sources and new third-party content systems.
• Standard web crawling is becoming more and more inefficient because web content is created and changed more rapidly today than ever before. The crawler crawls an ever-growing set of documents, while it actually needs only the delta of modified or newly created documents.
• Web crawling can't reach all content, for example, most crawlers can't follow links that are manipulated by JavaScript code.
• Content meta-data is growing rapidly. It needs to be indexed in a generic and consistent way among all types of content.

The seedlist format is rich enough to describe content and address the challenges described above. This format can be crawled by Portal search and OmniFind, although not yet by Internet search engines. IBM provides a white paper and source code to facilitate the creation of seedlists for any manner of content repositories. For example, if an organization has a document management system that is not supported by Portal, the seedlist framework could be used, together with the API of the document management system to create an ATOM feed in the seedlist format. Additional content sources can then be added to Portal to crawl this new seedlist feed, and add the documents to the search collection.

For more details see: http://www.ibm.com/developerworks/websphere/zones/portal/proddoc/dw-w-seedlist/

Crawlability of portal with WCM content
As with external search, there are additional factors that need to be considered if the Portal contains WCM content. The WCM content should of course be indexed using Portal's WCM crawler. However, note that this will index links to the content and not to the Portal page that is would normally render the content item. For example, a WCM content item representing some news would be displayed in isolation, not on the Portal 'News' page with other related portlets. This is because until recently, there was no link between the content and the Portal page it should be displayed upon.

In the past, this problem was resolved by using rules based mechanisms to adjust the URL to the WCM search results so they included the correct Portal page. This could be achieved by reformatting the URLs output by a WCM navigator component and using that as the crawler start point, or alternatively making a custom search portlet to reformat the search result URLs before presenting to the user. In both cases, this was often driven by fixed rules and was not a very dynamic solution if new site areas and Portal pages were added to the site.

This situation is much improved since the arrival of the Lotus Web Content Management Rendering Portlet in the portlet catalog:


Now using this rendering portlet with Portal v6.1.0.1 and above, when a search result is clicked from the search result portlet, the Portal will automatically determine the page responsible for rendering that content item, select the page, and set the page context to the content item. Thus, the user will see the selected content item on the page on which that item would normally be seen, including all the other portlets that should be on that page.

Categorization and taxonomy
Each search collection can optionally have one categorizer to organize documents into a taxonomy. The default collections in Portal 6.1.5 do not use a categorizer. If a categorizer is used, the taxonomy of search results only becomes apparent when using the search and browse portlet which enables browsing by category.

Portal Search provides two types of categorizers:

• Predefined static categorizer - This works by organizing documents into a predefined static set of over 2,300 categories for subject areas across twelve business disciplines. Note that this categorizer is disabled by default and cannot be selected when making a new search collection. If it is required, follow the instructions of this technote to enable it:
• User-defined rule-based categorizer - This allows categorization of documents by user-defined rules. The manage search portlet can be used to manage the taxonomy of rule-based categories. User defined rules can be based on URLs or search text. For example, there could be a rule to specify that all content with a URL including "/FAQ/*" should be categorized in the "Frequently Asked Questions" category.

Case study crawling the portal and displaying search results

After installing Portal 6.1.5, by default there is one search service (local) that contains two search collections: "Portal Content" and "WebContentCollection". The "Portal Content" contains a content source (or Portal crawler) configured to add the Portal pages to the collection, although by default it has not collected any pages. The "WebContentCollection" is a placeholder for content sources (or web crawlers) that gather web pages. For example, an organization may have some information in static HTML on a web server that is not integrated with the Portal.

To populate the "Portal Content" collection with some documents (portal pages), initiate its crawl activities. Click the Search Administration link from the Search Welcome page to reach the manage search portlet:

The '+' symbol allows individual documents to be added at any time without requiring a complete re-crawl, however in this case it would be better to trigger the complete crawl. Select the "Portal Content" collection and then press the "play" arrow to begin the crawl, the refresh button can be used to update the status:
The time to complete of course depends on the amount of content in the Portal. For a fresh install of Portal 6.1.5 there are only a few pages so the crawl completes in just a few minutes. Now visit the search center page, or use the search box in the theme to make a search for "web content management". The results are displayed in order of relevance:
Suggested links
To configure the suggested links portlet, use the manage search portlet to add a new search collection called "Suggested links":

<table>
<thead>
<tr>
<th>Reference</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Web Content Management - Web Content Authoring</td>
</tr>
<tr>
<td></td>
<td>Welcome - Information Portlet&lt;br&gt;&lt;b&gt;Summary:&lt;/b&gt; Display an information message contained in a JSP file</td>
</tr>
<tr>
<td></td>
<td>Getting Started - Information Portlet&lt;br&gt;&lt;b&gt;Summary:&lt;/b&gt; Display an information message contained in a JSP file</td>
</tr>
<tr>
<td></td>
<td>Welcome - Information Portlet&lt;br&gt;&lt;b&gt;Summary:&lt;/b&gt; Display an information message contained in a JSP file</td>
</tr>
<tr>
<td></td>
<td>Welcome - Information Portlet&lt;br&gt;&lt;b&gt;Summary:&lt;/b&gt; Display an information message contained in a JSP file</td>
</tr>
<tr>
<td></td>
<td>Welcome - Information Portlet&lt;br&gt;&lt;b&gt;Summary:&lt;/b&gt; Display an information message contained in a JSP file</td>
</tr>
<tr>
<td></td>
<td>Wires - Portlet Wiring Tool&lt;br&gt;&lt;b&gt;Summary:&lt;/b&gt; Allows user to edit connections between portlets</td>
</tr>
<tr>
<td></td>
<td>Organize Favorites - Organize Favorites&lt;br&gt;&lt;b&gt;Summary:&lt;/b&gt; Allows a user to add/delete/edit favorites</td>
</tr>
<tr>
<td></td>
<td>Welcome - Information Portlet&lt;br&gt;&lt;b&gt;Summary:&lt;/b&gt; Display an information message contained in a JSP file</td>
</tr>
</tbody>
</table>

Number of results per page: 10  20  30  50
The meta-data from the Web page is used to populate the information about the page in the search collection, although it can be edited:
Next, the suggested links portlet should be configured to use the new search collection:

The keyword "Web Content Management" is already populated in the "Suggested Links" collection so this search term should now trigger a result in the suggested link portlet:
The final task of this case study is to configure the external search results portlet. This requires a search service that can provide results in RSS or ATOM syndication format, that is, the same feeds that newsreader clients subscribe to. For this case study, we will use the feeds service from Yahoo to search for news stories relating to our search term. From the Yahoo web site, it is possible to build custom RSS feeds, for example a feed of all news stories about IBM looks like this:


The URL can then be modified to insert a place holder for the search query entered in the search portlet:


This needs to be configured as a parameter of the external search results portlet. Access the portlet's parameters via Administration->Portlet Management->Portlets and search for "External Search Results". Set the "searchEngineUrl" parameter to the URL above. Note you must press "OK" after editing the individual parameter and again on the manage portlet:
The external search is actually made as an Ajax request from the browser, and this requires the use of an Ajax proxy. Browsers are not permitted to make requests to servers outside of the domain that served the original page. Instead, the portlet will make a request to Portal which in turn makes the request to Yahoo, assuming it is configured as an allowed destination in Portal’s Ajax proxy configuration.

The external search portlet is packaged in the searchCenter.war file, so the Ajax proxy configuration must be amended here to permit access to Yahoo. Locate the following file:

<Portal Install Path>wp_profileinstalledAppsluxorPA_Search_Center.earsearchCenter.warWEB-INFproxy-config.xml

Add an additional element, in this case to enable access to feeds at any host:

<proxy:policy url="*">
    <proxy:actions>
        <proxy:method>GET</proxy:method>
        <proxy:method>HEAD</proxy:method>
    </proxy:actions>
    <proxy:mime-types>
        <proxy:mime-type>text/xml*</proxy:mime-type>
        <proxy:mime-type>application/xml*</proxy:mime-type>
        <proxy:mime-type>application/atom+xml*</proxy:mime-type>
        <proxy:mime-type>application/rss+xml*</proxy:mime-type>
    </proxy:mime-types>
</proxy:policy>
After editing this file the web module needs to be restarted. This can be done using the WebSphere
Application Server administration console, or by restarting the entire Portal.

Now when a search is made via the search center, external news results from Yahoo appear alongside:

More than 10 results found in All Sources for ibm

<table>
<thead>
<tr>
<th>Relevance</th>
<th>Title</th>
<th>Person</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Welcome - Information Portlet</td>
<td>N</td>
<td>11/12/12</td>
</tr>
<tr>
<td></td>
<td>Summary: Display an information message contained in a JSP file</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Welcome - Information Portlet</td>
<td>N</td>
<td>11/12/12</td>
</tr>
<tr>
<td></td>
<td>Summary: Display an information message contained in a JSP file</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Welcome - Information Portlet</td>
<td>N</td>
<td>11/12/12</td>
</tr>
<tr>
<td></td>
<td>Summary: Display an information message contained in a JSP file</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Welcome - Information Portlet</td>
<td>N</td>
<td>11/12/12</td>
</tr>
<tr>
<td></td>
<td>Summary: Display an information message contained in a JSP file</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Welcome - Information Portlet</td>
<td>N</td>
<td>11/12/12</td>
</tr>
<tr>
<td></td>
<td>Summary: Display an information message contained in a JSP file</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Welcome - Information Portlet</td>
<td>N</td>
<td>11/12/12</td>
</tr>
<tr>
<td></td>
<td>Summary: Display an information message contained in a JSP file</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Personalized List - Personalized List</td>
<td>N</td>
<td>11/12/12</td>
</tr>
<tr>
<td></td>
<td>Summary: Displays the results of a WebSphere Portal Personalization rule with custom formatting.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Search Center - Search Center</td>
<td>N</td>
<td>11/12/12</td>
</tr>
<tr>
<td></td>
<td>Summary: This portlet allows performing a scoped search and viewing Portal search results</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Welcome - Information Portlet</td>
<td>N</td>
<td>11/12/12</td>
</tr>
<tr>
<td></td>
<td>Summary: Display an information message contained in a JSP file</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Number of results per page: 10  20  30  50
Site analytics and optimization

Portal Analytics is the process of collecting, processing, and reporting portal usage data.

Often, enterprises using a portal for their external web site faced with questions such as:

- "What do our users really do with the portal?"
- "Will our portal be able to deal with evolving user needs?"
- "What are most users doing on the Portal?"
- "How should I plan for the growth of my Portal?"
- "What's the most popular search?"
- "Where are my users coming from?"
- "What are my customers downloading?"
- and so on

Site Analytics help you understand how your site is being used, effectiveness of advertising campaigns, page or portlet popularity, measuring load such as page views per second and unique logins per hour, modeling real user behavior for capacity and performance tests, and more.

Site Analytics enables customers to provide a more personalized experience to their clients by deploying a more dynamic web solution. Site Analytics helps in identifying ways to better target content. For example, if a particular page in the portal site is being accessed more or users are directly going to a page often, should that page be the home page or that content be moved on to the home page. It increases revenue with better targeting and can decrease cost with automatic tuning. Site Analytics also increases customer satisfaction and reduces testing costs with better designs. Site Analytics allows you to measure the success of a portal, predict the demand to a portal in the future and pro-actively plan for adapting to the community's needs, rather than being hit without warning.

So, customers are willing to buy products that are specialized and tested in Web site analysis. Integration with site analyzer tools are performed by creating reports based on the portal site analyzer logs or manually imbedding tags into portlets and themes. Hence, customers are expecting portal to provide the necessary data and offer seamless integration with each site analytics product.

Site Analytics gathers data on the following aspects of a portal, that can then be fed to an Analytic tool to analyze how a site is being used:

- Users requesting pages, incl. contained Portlets
- Session activities (login, logout, timed out, login failed)
- Page management (creating, updating, deleting a page)
- User Management actions (creating, updating, deleting users and groups)

Analytics are measured in one of two ways:

1. Server-side Analytics: It’s a fine-grained resource usage reporting method. It can be used offline and is ideal for historical analysis.
2. Active Site Analytics: It’s also called Client-side or "Web bugs". It’s a client side script-based and real-time reporting method, which collects cached content as well.

In general, active site analytics are recommended since they:

- have less impact on server resources,
- are easier to reconcile in a cluster since there is one log vs many,
- are easier to put in the instrumentation.
The subsequent sections will discuss the two analytic methods in detail.

**Server side site analytics**

Server-side Site Analytics, also known as logfile analysis, is a log file based approach. The site analysis infrastructure provided by the portal accommodates most scenarios and all the necessary data for site analytics. A typical server-side site analytics architecture is shown in the diagram below. User interactions and metadata are written to server logfiles (sa.log), hosted on the Portal Server. Logfiles can be recorded and stored for later offline processing. Logfile recording is activated via a simple configuration change in WebSphere Portal. A list of all the server-side loggers available is described in the table below.

![Server side site analytics architecture](image)

**Server side site analytics architecture**

Information that can be derived from the analytics log includes, but is not limited to:

- Browsers of users visiting the portal
- Different operating systems as reported by the browser
- Different logins corresponding to authenticated users
- Robots and programmatic clients
- Pages that were requested, but not found
- Search engines, key phrases and keywords
- The referring page a user came from
- Entry and exit URLs
- Virtual portal specific and WCM viewed usage statistics, since WebSphere Portal 6.1.0.1 is released

Site Analytic tools offered by third party vendors can be used to analyze the collected data and generate reports. Analytics engines are not part of the WebSphere Portal product offering. Earlier versions of Portal shipped with Tivoli Site Analyzer, which is now discontinued. Customers, who are still running WebSphere Portal version 5.1, can continue to use Tivoli Site Analyzer.

**Server side loggers**

WebSphere Portal provides fine grained usage information, but it will be up to the analytics engines to derive the information from the data in the logs. The recorded URLs are not real, clickable URLs, but they still provide the relevant information to find out which pages the users of the portal looked at. With the logfile
based approach, only hits on the Portal Server are recorded, so cached page hits are not recorded. Portlets can report specialized business events by calling analytics log API, since WebSphere Portal version 6.1.0.1.

<table>
<thead>
<tr>
<th>Logger</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>SiteAnalyzerSessionLogger</td>
<td>Logs session events like login or logout</td>
</tr>
<tr>
<td>SiteAnalyzerUserManagementLogger</td>
<td>User and group management events like creating or deleting users and groups</td>
</tr>
<tr>
<td>SiteAnalyzerPageLogger</td>
<td>Page render events</td>
</tr>
<tr>
<td>SiteAnalyzerPortletLogger</td>
<td>Portlet render events</td>
</tr>
<tr>
<td>SiteAnalyzerJSRPortletLogger</td>
<td>Custom business events in standard Portlets (JSR 168, JSR 286)</td>
</tr>
<tr>
<td>SiteAnalyzerPortletActionLogger</td>
<td>Actions occurred in a Portlet</td>
</tr>
<tr>
<td>SiteAnalyzerApplicationActionLogger</td>
<td>Actions occurred in a Portlet application</td>
</tr>
<tr>
<td>SiteAnalyzerErrorLogger</td>
<td>Logs any errors</td>
</tr>
</tbody>
</table>

**Analytics and reporting with WCM**

The JSR 286 Web Content Viewer portlet implements detailed site analysis logging, which enables you to gather usage statistics of each instance of the WCM rendering portlet, since WebSphere Portal version 6.1.0.1 is released. It also offers syndication pending and deployment views as well as additional date fields.

The configuration process to enable WebSphere Portal Site Analyzer requires a simple change to a properties file. The logger that needs to be activated is SiteAnalyzerJSRPortletLogger logger. This will cause log information to be added to the site analytics log (sa.log) files. The site analytics log can then be formatted using a third party analytics tool in order to analyze the collected data. When the generated reports are shared with content authors, it will give them an opportunity to enhance their content to better target the content audience.

**Active site analytics**

Active site or Client-side analytics, also known as Page Tagging Analytics, is JavaScript injection based. JavaScript is embedded in the theme or skin of a page, so the JavaScript is also called a “web bug”. The web bug helps to gather data from portal perspective, where as 3rd party vendors or partners will provide the reporting function and analysis. IBM also provides a few sample JavaScript scripts out of the box, which can be modified to suit the needs or the partner supplied web bugs can be embedded in a portal page. A typical Active Site Analytics Architecture diagram is shown in the picture below.
The following diagram depicts a general scenario of how the data is collected in Active Site Analytics. When a client (browser) requests a portal page, portal returns page markup, including metadata markup. Client aggregates the metadata using JavaScript snippet or web bug embedded in theme/skin of the page. Client then sends aggregate data to the analytics server, which is typically installed on infrastructure, separate from portal server’s, e.g. via HTTP protocol. Analytics Server’s web server instance records the access log.

What gets recorded depends on the specific 3rd party client-side analytics solution. Typically recorded events are page display, listing of items on the page, download actions and links leaving the site. The injected JavaScript determines which data will be recorded and it requires proper planning. Some tools cover multiple requests per single page such as page, portlet and content. Injection into themes and skins allows reporting of page and Portlet views. Portlet business events are collected by adding JavaScript to the Portlet markup. Any rendering of the corresponding page in the user’s browser can be recorded, including cached pages. 3rd party Analytics software systems gather and process this data locally and generate reports.
Supported-tags or micro-format are used to annotate the page markup with the page name, portlet names and ids and themes / skins will be instrumented according to the micro-format for Active Site analytics logging in WebSphere Portal.

### Microformat/ Supported Tags

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Tagged as</th>
<th>Injected in</th>
</tr>
</thead>
<tbody>
<tr>
<td>Page Title</td>
<td>Title of the page in portal default language.</td>
<td>asa.page.title</td>
<td>Theme</td>
</tr>
<tr>
<td>Portlet Window Title</td>
<td>Title of the portlet as delivered to the client</td>
<td>asa.portlet.title</td>
<td>Skin</td>
</tr>
<tr>
<td>Portlet Window ID</td>
<td>Unique identifier of the portlet</td>
<td>asa.portlet.id</td>
<td>Skin</td>
</tr>
<tr>
<td>WCM Content Querystring</td>
<td>Unique identifier of the WCM content item that is displayed in a portlet</td>
<td>asa.portlet.screen.id</td>
<td>WCM Rendering Portlet</td>
</tr>
<tr>
<td>WCM Content Title</td>
<td>May not be the same as the portlet window’s title</td>
<td>asa.portlet.screen.title</td>
<td>WCM Rendering Portlet</td>
</tr>
</tbody>
</table>

Aggregator script supplied by the business partners will be injected into the portal page’s markup. The aggregator will iterate over all instances of the micro-format, then extract the relevant information and submit it to the external analytics server. Aggregators are managed as part of the theme, including staging to production.

### Instrumenting a theme for active site analytics

This section described the steps involved in instrumenting a theme for Active Site Analytics. Decide what data should be recorded and mark the data in the theme or skin JSPs:

```html
<span class="asa.portlet.id" style="display:none;">%
<%= myPortletID %>
</span>
```

Include aggregator with the page:

```html
<portal-theme-ext:themeExtension
id="com.ibm.portal.theme.plugin.ActiveSiteAnalyticsItems">
<portal-theme-ext:themeExtensionLoop>
<portal-theme-ext:themeExtensionItemText />
</portal-theme-ext:themeExtensionLoop>
</portal-theme-ext:themeExtension>
```

Adapt the aggregator, if required. Note that, dojo provides simple ways to access DOM data.

### Differences between server side and active site analytics

Here is a comparison of Server-side versus Active Site Analytics. It's important to note that, both approaches can be used with WebSphere Portal, either exclusively or in combination.

### Server side
User interactions and meta-data is written to server log files, hosted on the Portal Server.

- Log files can be recorded and stored for later offline processing.
- Analytics engines read these log files for collecting analytics data.
- WebSphere Portal provides fine grained usage information.
- Only hits on the Portal Server are recorded and for example, cached page hits are not recorded.
- Log file recording is activated via configuration in WebSphere Portal.
- No special coding needed for base reporting (pages, portlets, etc.)
- Portlets can report specialized business events by calling analytics log API, since portal version 6.1.0.1.

Active site

- JavaScript within the Portal page (e.g. theme / skin) notifies analytics server about user activity.
- Analytics data is collected online within the Analytics system, which gathers and processes this data outside the portal server and the browser.
- Up until WebSphere Portal v6.1, appropriate data and JavaScript needed to be injected manually into Portal. With v6.1.5, IBM provides the integration out of the box.
- The injected JavaScript determines which data will be recorded.
- Injection into themes and skins allows reporting of page and portlet views.
- Portlet business events are collected by adding JavaScript to the Portlet markup.
- Any rendering of the corresponding page in the user’s browser can be recorded, including the cached pages.

User experience optimization initiative

Today, IBM customers are demanding better integration between WebSphere® Portal Server and Web analytic tools. They use site metrics to capture and measure user activity primarily to understand end user needs, behaviours and site usability, so that portals can be better designed and targeted. In response to our customers’ requests, IBM created the IBM User Experience Optimization Initiative (UEOI) that teams up IBM with leading Web analytic software companies (Coremetrics, WebTrends, and Omniture) to seamlessly integrate their products with the WebSphere Portal family product line, OmniFind®, Quickr®, and IBM Lotus® Web Content Management. This is the first step in implementing our vision to provide an improved and more personalized experience to our clients by allowing them to deploy dynamic and best-of-breed website analytic solutions.

Three market-leading partners have signed strategic partnership agreements with IBM for this initiative -- Coremetrics, Omniture and WebTrends. Coremetrics is an IBM Partner and has an alliance with WebSphere Commerce. Omniture is a fast growing web analytic leader and a leading provider of online business optimization software. It has integrations with IBM WebSphere Commerce and WebSphere Portal. WebTrends is a well established analytics product among WebSphere Portal customers.

Future outlook and summary

WebSphere Portal will offer enhanced integration with Web Analytic vendor products, when version 7 is released in 2010. Future releases will also aim to create an administrative portlet, which will enable the user to choose the web analytic product he/she is using and add portlets with dashboard and content / grouping segmentation reports to provide portal and portlet views.

The long-term vision is to create a self-optimizing portal that automatically responds to the demand on the site and optimizes for better results.

Additional information and resources

Site Analytics properties on a portal server can be seen in file:  
<profile>/PortalServer/config/SiteAnalyzerLogService.properties

WebSphere Portal 6.1.5 InfoCenter - Analyzing portal usage data
Using portal analytics with open-source reporting tools (developerWorks)

Adapting AWStats for IBM WebSphere Portal 6.0.x and virtual portals

WebSphere Portal – IBM Site

WebSphere Portal Business Solutions Catalog:
http://catalog.lotus.com/wps/portal/portal

Websphere Portal Developer’s Zone

Product Documentation and WebSphere Portal Wiki
http://www-10.lotus.com/ldd/portalwiki.nsf

Education

WebSphere Portal DemoNet
http://docs.dfw.ibm.com/wp6/?DDSPageRequest=/
Mobile device support

This chapter provides an overview of IBM Mobile Portal Accelerator v6.1.0.2 and details about delivering Lotus Web Content Management (LWCM) content to mobile devices.

Overview

Driven by major growth trends and demands to constantly access information on-the-go (email, business applications, social networking applications, and downloads such as ring tones, photos, videos or games), the mobile applications market is poised for rapid growth. Integration between Content Management Systems, Mobile Portals, and the sophisticated features of new mobile devices is helping enterprises to maximize on both their B2C and B2B opportunities.

Mobile portals have a smaller footprint of displayable content with navigation through menu links or icons. The ability to create dynamic content in a timely manner and render it on to a variety of mobile devices without being concerned about a mobile device’s specific physical characteristics (screen size) is essential in providing a rich mobile user experience. Lotus Web Content Management (LWCM) and IBM Mobile Portal Accelerator (MPA) are now deeply integrated via authoring and presentation templates to automatically wrap Web content in XDIME to provide seamless content publishing to virtually all mobile devices used today.

Mobile Portal Accelerator

IBM Mobile Portal Accelerator (formerly known as IBM WebSphere® Everyplace® Mobile Portal Enable) is an advanced content adaptation product that provides seamless, cost-effective ways to deliver web content via portlets to mobile devices.

The content is automatically formatted and customized to display on over 6000 mobile devices worldwide via its Multi-channel Server. Instead of the PC-based browser HTML markup of choice, Mobile Portal Accelerator uses an XML vocabulary called XML Device Independent Markup Extension (XDIME/XDIME 2). XDIME 2 describes content, but does not explain how to display content. It is the device policies from the device repository that determine how the content should be displayed.

- Mobile Portal Accelerator includes the following out-of-the-box XDIME 2 mobile portlet samples to jumpstart mobile portal solutions:

  - **Mobile Portal RSS** is an XDIME/XDIME 2 mobile-enabled version of the IBM Syndicated Feeds Portlet and supports RSS2 and ATOM feeds
  - **Mobile Portal Stock portlet** is an XDIME/XDIME 2 mobile-enabled portlet that displays stock quote and performance data for selected companies, symbols and investments, with configuration and edit options for users to modify selected stocks and quote sources
  - **Mobile Struts Mail Reader portlet** sample
  - **Mobile Image Conversion Server (ICS) portlet** sample
  - **Change of Address portlet** sample for mobile devices
  - **Calendar portlet** sample, for managing appointments, meeting events and appointments
Mobile Portal Accelerator also provides seamless integration with Lotus WCM content authoring services. LWCM has extended its capability to present content not only using HTML markup, but also using XDIME markup, that enables content for display on mobile devices. The presentation templates allow a single copy of the authored content to be presented as both HTML and XDIME simultaneously in portlets. In LWCM, there are two portlets used for rendering content: LWCM local rendering portlet and an LWCM remote rendering portlet.

Sample Portlet is a mobile-enabled portlet and artifacts that showcase over 50 XDIME 2 sample widgets that can be used to deliver an enhanced, more interactive experience to client capable mobile devices. Examples of widgets include styling, transition, AJAX table, auto complete, popup, ticker tape, and more.
According to your environment setup and platform, verify that the following described here:
http://publib.boulder.ibm.com/infocenter/mpadoc/v6r1m0/index.jsp?topic=/com.ibm.websphere.wemp.doc/designing/update_wcm_portlets_c.html have been implemented in either the LWCM local rendering front-end portlet or the LWCM remote rendering front-end portlet to support XDIME seamlessly out-of-the-box.

New features in Mobile Portal Accelerator
Mobile Portal Accelerator V6.1.0.2 has following new features:

Key New Features

- Support for rendering of Lotus Web Content Management (WCM) content to XDIME for multi-channel rendering to mobile devices
- New operating system and database support
- New Multi-Channel Server Client Framework creation of XDIME applications, and sample widgets with rich client-side user interfaces that work on sophisticated mobile devices. The Mobile Portal Client Framework Sample portlet provides a rich set of XDIME 2 widgets for use in developing applications. The following widgets are available: AJAX widgets, Non-AJAX widgets, Effects widgets, Building Block widgets.

Support for Multichannel Server
MCS version 4.3.5 includes the following enhancements:
• XDIME 2 support: Users have the option of using XDIME2 as their primary content authoring language for MCS.

• Enhanced theme processing: MCS version 4.3.5 streamlines the processing of themes by sending only the style information that is needed by a given page and by using shorter style names in the XHTML output.

• Device classification: You can map component, layout, and theme policies to devices according to device groupings and device hierarchies that you have defined.

• User interface for theme variants: A new user interface shows a summary of the properties defined on a theme variant and provides a mechanism for navigating to a property.

Updated runtime and development code

• The IBM Mobile Portal Accelerator version 6.1 runtime features are based on WebSphere Portal version 6.1.0.3.

• The Mobile Portal Toolkit is based on Rational Application Developer version 7.5.1 and Rational Software Architect version 7.5.1. Developers can use the standard portlet API (encompassing JSR 168 and 286). The IBM portlet API is no longer recommended for use.

Installation enhancements

• The installation process makes use of enhancements in WebSphere Portal version 6.1.0.3, including use of the ConfigEngine utility and updated directory structure.

• A health check utility is provided so that, when the installation is complete, you can verify that all required files are in their proper locations.

XDIME 2 sample portlets

The following new XDIME 2 portlets are provided for out-of-the-box capability and as samples for demonstration purposes:

• The Mobile Wireless Portal (MWP) Stock portlet gives users the ability to view stock quotes and financial information for a customizable list of stock ticker symbols.

• The Syndicated Feed portlet (which replaces the Mobile RSS portlet) extends the existing portlet for WebSphere Portal to XDIME 2, giving users the ability to read Web feeds, such as RSS, on their mobile devices.

Customer experiences

For one example of a company utilized Mobile Port Accelerator to make a difference, watch the flash video available at: http://www.ibm.com/ibm/ideasfromibm/us/bharti_airtel/20080714/index.shtml

High level architecture

The following figure and the sections below provide an overview of the architecture of a mobile device solution using WebSphere Portal.
**Multi Channel Server**

The Multi-Channel Server is the runtime component that transforms XML-based Device Independent Markup Extensions (XDIME) markup into native markup languages for individual devices. MCS uses the built-in MCS Policy Repository to manage a large number of devices such as PDAs, cell phones, smartphones, Web TVs, and other devices.

The MCS Policy Repository is not a single database or a single file; rather, it is a set of policy files managed by MCS. These MCS policy files define the presentation characteristics (layout, component and theme, and so on) of a device. There are a number of policies defined in MCS. Device, layout, theme, and component policies are the most commonly used.

- **Device policy** - The device policy is stored in a compressed XML file, extension *.mdpr, containing specific attributes of devices supported by MCS. The device policy repository can be updated as new devices emerge by subscribing to the Mobile Portal Accelerator Device Update Subscription service.
- **Layout policy** - The layout policy specifies physical positions of elements on a page. MCS manages the policies and maps resources to devices. MCS policies describe specific layout rules used to render pages for requesting devices. The layout policy file has the extension *.mlyt. Create different layout policies to map different mobile devices to specific layout policies.
- **Theme policy** - The theme policy is similar to the theme concept in WebSphere Portal. With it, you can manage the overall look and feel on devices. It has the file extension *.mthm.
- **Component policies** - Use the component policies to address complex content type such as images, rollover images, audio, chart, and dynamic visual elements. The file extensions of component policies vary depending on the component types. MCS transforms the device-independent (XDIME markup) content into device-specific native markups (WML, cHTML, and so on.) by utilizing the powerful combination of XDIME and various policies defined in MCS Policy Repository.

**Mobile Portal Accelerator extensions**

MPA include the XDIME aggregator and several portlets. For example, the Manage Mobile Pages portlet and the Extended Properties portlet are provided to facilitate management and deployment of mobile portlets.
Manage Mobile Pages portlet
The Manage Mobile Pages portlet is designed to be used by portal administrators and marketing personnel to create mobile style navigation. The navigation tree of mobile pages is stored in the navigation model of WebSphere Portal, and consists of nodes that represent pages, URLs, labels, or portlets.

Extended Properties portlet
The Extended Properties portlet, accessed through the Manage Mobile Pages portlet, provides a set of pages to configure the display attributes and the rules of nodes in the Mobile Portal Accelerator navigation hierarchy. Each node (page, portlet, and URL) in the MPA has associated extended properties or attributes that determine how and when it should be displayed. Extended properties or attributes include icons, device capabilities, and cache control.

XDIME Aggregator
XDIME aggregator extends the existing portal aggregation support to XDIME/XDIME 2. It is associated with the markup type XDIME and the MIME type text/xml. WebSphere Portal uses the user agent of the requesting device to look up a supported client definition. If the client definition for the device specifies XDIME markup, then the XDIME aggregator is invoked to combine the markup from all the portlets together with headers and footers to generate the page. At runtime, the XDIME aggregator uses the navigation tree created by the Manage Mobile Pages portlet to select and aggregate the output (pages, portlets, and URLs) in XDIME.

During this process, the aggregator evaluates the meta-data attributes for each navigation node. For instance, the XDIME aggregator queries the attributes to display appropriate icons or alternate text. MCS converts the aggregated XDIME markup generated by XDIME aggregator into device-dependent markup such as cHTML or WML, based on the requesting device, component, theme, and layout policies referenced in the XDIME markup.

Mobile Portal Toolkit
To help developers create XDIME/XDIME 2-enabled mobile portlets, the Mobile Portal Accelerator software includes the Mobile Portal Toolkit, an Eclipse plug-in. The toolkit can be used with IBM® Rational® Application Developer 7.5.4 or IBM Rational Software Architect 7.5.1 to create XDIME/XDIME 2 portlets and Multi-Channel Server policies for mobile portlets.

The toolkit includes:

- Portlet Creation Wizards. These wizards include support for creating Mobile XDIME Standard and Mobile Struts portlets that support standards-based JSR168/286 APIs. Developers can use the Portlet Creation Wizards to create configure-and-edit mode JavaServer™ Pages (JSP) for XML-based, XDIME (XHTML with Device Independent Markup Extensions) portlets. (XDIME is a flexible, device-independent authoring language that is used in many mobile networks to collect information about a user’s request; then to select related pages or portlets; and finally, to aggregate the output for display. With XDIME aggregation, content can be created once but delivered to a wide variety of mobile devices that are used to connect to the Web.)
- Device Repository of mobile-device policies, which define visual presentation elements (such as page layouts and style sheets) and device-attribute information, and device categorization for over 6000 mobile devices.
- Policy editors. Rational-software-based policy editors are used to create policies, which are referenced in the XDIME markup and used by the Multi-Channel Server runtime to make the best decision for emitting device-dependent markup to a target device.
- Test Environment. The test environment enables portlet developers to deploy and test XDIME portlets.
**Runtime component interaction**

The following figure and the text below describes the run-time interaction of the various components.

The numbers in the figure correspond to key steps in a typical MPA scenario:

1. Write the XDIME Portlet or add XDIME JSPs to existing portlet

2. The resulting navigation hierarchy defines nodes (pages, labels, URLs, and portlets) and extended attributes that specify required device capabilities and type for each node. Navigation nodes and attributes are stored in the Portal Model.

3. When the Portal receives a request from a mobile device, the appropriate markup is determined by comparing the User-Agent string to configured Portal clients. MPA clients are configured with XDIME support, so the portal passes the request to the XDIME Portal Filter. The filter then invokes the XDIME Aggregator to process the request.

4. The XDIME Aggregator queries the Portal model to determine navigation and portlet availability based on the user and the extended attributes of each node. If the request is for a portlet node, a PortletRequest object, containing request-specific data, is passed to the portlet.

5. The Portlet Container invokes the portlet with the PortletRequest. XDIME portlets render their content in XDIME and return the content to the aggregator via a PortletResponse object.

6. Aggregated XDIME markup for the requested navigation or portlet is returned to the Portal Filter, which passes it to the Multi-Channel Server. MCS transforms the XDIME content to device-specific markup by
matching it with policies in the repository of Mobile Device Profiles.

7. The Portal Filter inserts the device-specific content in the ServletResponse object for delivery to the mobile device.

**Installation hints and tips**

IBM Mobile Portal Accelerator v6.1.0.2 requires a separate installation after WebSphere Portal v6.1.0.3 Feature Pack 6.1.5 has been installed and configured. Download the MPA v6.1.0.2 code fixes from Fix Central: [http://www-933.ibm.com/support/fixcentral/](http://www-933.ibm.com/support/fixcentral/)

### Create a DB2 MCS database

1) Verify that you have a working WebSphere Portal server (v6.1.0.3 Feature Pack 6.1.5) with any applicable fixpacks needed and that uses one of the following databases with its applicable fixpack:

- DB2 Universal Database Enterprise Server Edition version 8.1 (with fix pack 16) or version 9.1 (with fix pack 2 or 4) or DB2 Universal Database Workgroup Server Edition version 9.5 (with fix pack 1) (Note: my local DB2 installation was based on DB2 Workgroup Server 9.5 FP1)
- Oracle Database Enterprise Edition Release 10.1.0.5 patch 4, Release 10.2.0.3, or Release 11g
- Microsoft SQL Server 2005 Enterprise Edition SP2

2) Install Java SDK 1.5.0 service release 8 or higher if it is not already installed.
3) Unzip the file wp.mobile.zip to a temporary directory to create Multi-Channel Server (MCS) and Mobile Portal Accelerator installation images on your WebSphere Portal application server. (Note: The temporary directory is specified when you run the ConfigEngine assembly command below). Next, install MCS and Mobile Portal Accelerator as a ConfigEngine assembly using the following command (where tmp is used as the name of the directory to which you unzipped wp.mobile.zip)

```
ConfigEngine -install -assemblyludd sdd.xml -assemblyRoot c:tmpwp.mobile -installLocation c:IBMWebSphere -offeringName PortalServer
```

4) Prior to creating a database for a DB2 MCS, verify that you have set the DB2 Variable DB2_CREATE_DB_ON_PATH=YES as shown in the following figure:

5) a) Create database using the following commands in a db2 command window:

```
db2 create db mcs on path_name using codeset UTF-8 territory us
```

b) Update the mcs database configuration using the following command

```
db2 update db cfg for mcs using logfilsiz 10000 logprimary 10 logsecond 10
```

c) Connect to the mcs database

```
db2 connect to mcs user db2instance_userID using db2instance_password
```

d) Copy the file wp_root/wp.mobile/wp.mobile.mcs/repository/db2/create_vm_tables.sql from WebSphere Portal server to a directory on your database server. Then, make that directory your working directory

e) Create the WebSphere Portal tables in your mcs database using the following command.

```
db2 -tf create_vm_tables.sql
```

6) Log in to your WebSphere Portal Admin console and add the JDBC drivers to the Java Virtual Machine (JVM) classpath of the WebSphere Portal server. Go to the following on the Portal Admin Console: Servers -> Application Servers -> WebSphere_Portal -> Server Infrastructure -> Java & Process Management -> Process Definition -> Java Virtual Machine: In the Classpath field, add the following JDBC drivers if using DB2 9.5 Fixpack1

- db_root\sqllib\java\db2jcc.jar
- db_root\sqllib\java\db2jcc_license_cu.jar

Save the configuration changes

7) Run the following command to ensure that the WebSphere environment is set up correctly

```
wp_profile_root\bin\setupCmdLine
```
8) Add the JDBC drivers to your current CLASSPATH environment variable.

9) Make sure that the correct values are specified for the following parameters in the file wp_profile_root/ConfigEngine/properties/wkplc.properties: (Note: These are values that were entered or existed when data was transferred from the default Derby database to DB2)

- WasHome
- WpsInstallLocation
- NodeName
- ServerName
- PortalAdminId
- PortalAdminPwd
- WpsHostPort
- WpsContextRoot

10) Change your working directory to wp_profile_root/ConfigEngine and run the following command (Note: Ensure the command specified below is on one line)

```bash
```

**Note:**
-DMcsRepositoryType and -DMcsProject have required default values:

  - `hostname`: the fully qualified host name of the database server. (for example itsoportal.sc.ibm.com)
  - `db_name`: the name of the Multi-Channel Server database (ex. mcs)
  - `db2instance`: the user ID of the DB2 instance user (ex. db2admin)
  - `db2instance_password`: the DB2 password for MCS database (ex. db2admin)
  - `mobile-portal`: the project name

After running the command above, the return code should display Return value 0 in the DOS command prompt.

11) Using a text editor, open the ImportPolicies batch file in the wp_root/wp.mobile/wp.mobile.mcs/repository/bin directory. Add the path of the JDBC drivers to the end of the CLASSPATH= line:
12) Run the following command to import the default device policies:

```
ConfigEngine init mcs-import-devices -DMcsProject=mobile-portal
```

After running the command above, the return code should display Return value 0 in the DOS command prompt (Note: It may take up to 2 hours before the import is complete)

13) Restart the WebSphere Portal application server.

**Installing MPA extensions**

If you are installing with v6.1.0.3 Feature Pack 6.1.5 of Portal, the following action may be necessary to complete installation of the MPA extensions. Check to see if wp.config.jar exists in \IBM\WebSphere\wp_profile\ConfigEngine\lib. If it does not, copy wp.config.jar from \IBM\WebSphere\PortalServer\installer\wp.config\bin to \IBM\WebSphere\wp_profile\ConfigEngine\lib. At the time of our installation, v6.1.0.3 Feature Pack 6.1.5 of Portal had not been released, so this was a workaround to resolve a SqlProcessor class not found error that was occurring after running the command to install and configure all of the MPA extensions

1) Use the existing DOS command window and verify the values still exists for step 8 (setting up the classpath to point to the JDBC drivers) and step 9 (verifying existing values in the wkplc.properties file) above in "Creating a DB2 MCS database".

2) Run the following command to install and configure the MPA extensions with sample XDIME portlets

```
ConfigEngine mwp-config-all
```

After running the command above, the return code should display Return value 0 in the DOS command prompt (Note: It may take up to 1 hour before install for mwp extensions is complete)
3) Run the following command to import XDIME aggregator policies and generic portlet policies into the MCS policy repository

```
ConfigEngine mwp-import-policies
```

After running the command above, the return code should display Return value 0 in the DOS command prompt.

4) Import Mobile Welcome portlet policies using the following command

```
ConfigEngine mwp-import-portlet-policies
```

After running the command above, the return code should display Return value 0 in the DOS command prompt.

5) Restart WebSphere_Portal application server

**Note:** In WebSphere Portal v6.1.0.3 FP 6.1.5 w/MPA 6.1.0.2, the Manage Mobile Pages, Extended Properties, and Preload Notice pages display an empty portlet with title of page. A bug report has been opened to resolve the problem.

For detailed steps about installation and running a health diagnostics check of your installation, see: [http://publib.boulder.ibm.com/infocenter/mpadoc/v6r1m0/index.jsp](http://publib.boulder.ibm.com/infocenter/mpadoc/v6r1m0/index.jsp)

The optional Image Conversion Server (ICS) component has not been installed.

### Installing Mobile Portal Toolkit

1) After Rational Application Developer v7.5.4 is installed, double click on the MPA Toolkit installation image executable that was unzipped previously in step 3 above.

Follow the prompts to install the toolkit with a portal test environment in RAD

**Note**, that the Lotus Web Content Management structure is created by default during your WebSphere Portal installation.

### Validating installation of mobile portlets

1) Log in to the WP Administration console

2) Select **Applications->Enterprise Applications**
Tools for Mobile Portal Accelerator
The toolkit includes support for mobile application development: markup/layout editors, templates for rapid site development and enforcement of style guidelines, mobile portlet samples, APIs, and documentation. In addition, when the toolkit is installed, sample XDIME/XDIME 2 portlets are added to the Samples Gallery for use within the Rational development environment. Mobile Portal Toolkit 6.1.0.2 provides support for the following Rational development platforms:

- Rational Application Developer 7.5.4
- Rational Software Architect 7.5.2

In your Rational development environment, select Help Contents from the Help menu. Then refer to the section titled Mobile Portal Toolkit Information.

Note: This help information is available only after you install the Mobile Portal Toolkit.

- Multi-channel Server

Multi-channel Server enables development of policies in the Mobile Portal Toolkit workbench for layouts, devices, themes, components and assets, Newly developed policies can be imported into your database repository.

Tools for authoring content in LWCM

- Inline editor
**Design considerations for your mobile portal**

As a content designer for LWCM content, presentation templates are used to display the look and feel of the site. Updates to existing presentation templates with HTML-type content elements are necessary (for example, img, Rich Text, and so on) in order to display on mobile devices. For example, instead of using the "Rich Text" element type, that creates HTML, use the "Text" element type. If updates are not made to use non-HTML elements, the following generic error message may display on your emulator/simulator: Error 500: MCSAA0051X: Caught exception. Additional details about the error may be found in the wp_profile_root/logs/WebSphere_Portal/SystemOut.log. However, to support the display of content on both PC-based and mobile devices in parallel, the default HTML markup content presentation template is required as well as an additional presentation template using the XDIME markup language. (See sample scenario below for details) Configurations to support the additional XDIME presentation template in your portlet are done in the "Edit Shared Settings" view of the portlet. See details in the section below on "Configure portlet to display LWCM content on mobile device".

As a UI designer/developer for mobile portals, the dynamic device-specific content presentation is done using the XDIME markup language with Java Server Pages and the MCS policies for layout, component, and theme. After a mobile portlet project creation in the Mobile Portal Toolkit, edit JSPs under the WebContent folder for xdime. The html folder and view JSPs are created by default. See the table below for a subset of common XDIME markup elements and their equivalent HTML tags used in LWCM templates.

<table>
<thead>
<tr>
<th>Common XDIME/XDIME 2 Elements</th>
<th>Equivalent HTML Elements</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>contentType= &quot;x-application/vnd.xdime+xml&quot;</td>
<td>contentType=&quot;text/html&quot;</td>
<td>content type</td>
</tr>
<tr>
<td>&lt;xfform&gt;</td>
<td>&lt;form&gt;</td>
<td>form used in page</td>
</tr>
<tr>
<td>&lt;xftextinput type=&quot;text&quot;&gt;</td>
<td>&lt;input type =&quot;text&quot;&gt;</td>
<td>text input field</td>
</tr>
<tr>
<td>canvas&gt;&lt;/canvas&gt;</td>
<td>&lt;/html&gt;</td>
<td>beginning/ending of page</td>
</tr>
<tr>
<td>&lt;layoutName=&quot;name.mlyt&quot;&gt;</td>
<td>&lt;table border=&quot;0&quot; width=&quot;100%&quot;&gt; &lt;tbody&gt;</td>
<td>arrange content on page</td>
</tr>
<tr>
<td>&lt;pane=&quot;name&quot;&gt;</td>
<td>&lt;tr&gt;</td>
<td>destinations for content output</td>
</tr>
<tr>
<td>&lt;img url=&quot;&quot; /&gt;</td>
<td>&lt;img src=&quot;&quot; /&gt;</td>
<td>display image</td>
</tr>
</tbody>
</table>


- For details about the differences between the first implementation of XDIME (called XDIME 1) and XDIME 2, refer to the Volantis Web site at [www.volantis.com](http://www.volantis.com).


After the creation of your mobile portlet, to deliver the content to a range of device types, create a layout policy with the MCS workbench. The layout policy is given a name and associated in the XDIME markup through the canvas element type. A sample_layout.mlyt file is created by default under the WebContent/mcs-policies directory in the Mobile Portal Toolkit. Delete this sample layout file and right click.
on mcs-policies and select **New>Other>MCS>Layout** to create a new custom layout for your portlet in the mcs-policies directory. When clicking the file to view the sample layout, you are prompted to specify the location of the standard device repository file, devices.mpdr. It is found under your installation directory for Mobile Portal Toolkit\DeviceRepository. A default layout policy created with the sample Change of Address portlet displays below.

The top of the layout design contains two rows and one column with a name of title assigned to the Pane. The title Pane will contain the information submitted in the form beneath. The bottom portion of the layout design for the sample Change of Address portlet contains a form with ten rows and two columns with T1-T10 assigned to the name of the panes for display of content. See the layout of the sample Change of Address portlet in the WebSphere Portal user interface.
**MPA user interface**

Access the administrative user interface of Mobile Portal Accelerator through the Manage Mobile Pages Administration portlet. This portlet provides a navigation tree for urls, labels, and pages. Properties (such as, markup supported) are displayed after a label or page is selected. The Manage Mobile Pages portlet provides access to the Preload Notice portlet and the Extended Properties portlet.

---

**Change Of Address Form**

Current Address is:
Name: Mrs. IBM User
Home:, Business:
Fac., Mobile: 999-999-9999
Title: IT Specialist
Company: IBM
Street: 1177 Beltline Rd
City, Coppell, State/Province: TX
Zip/Postal Code: 75019, Country: US E-Mail: user@ibm.com
www.ibm.com

<table>
<thead>
<tr>
<th>Title</th>
<th>Mr.</th>
<th>Business</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>First Name</td>
<td>IBM</td>
<td>Home</td>
<td></td>
</tr>
<tr>
<td>Middle Name</td>
<td></td>
<td>Fax</td>
<td></td>
</tr>
<tr>
<td>Last Name</td>
<td>User</td>
<td>Mobile</td>
<td>999-999-9999</td>
</tr>
<tr>
<td>Job Title</td>
<td>IT Specialist</td>
<td>E-Mail</td>
<td><a href="mailto:user@ibm.com">user@ibm.com</a></td>
</tr>
<tr>
<td>Company</td>
<td>IBM</td>
<td>Web Addr</td>
<td><a href="http://www.ibm.com">www.ibm.com</a></td>
</tr>
<tr>
<td>Street</td>
<td>1177 Beltline Rd</td>
<td>City</td>
<td>Coppell</td>
</tr>
<tr>
<td>City</td>
<td></td>
<td>State/Province</td>
<td>TX</td>
</tr>
<tr>
<td>Zip/Postal Code</td>
<td>75019</td>
<td>Country</td>
<td>US</td>
</tr>
</tbody>
</table>

Submit
Preload notice portlet
This portlet is used to display custom, informational, promotional advertisements or warnings about content that is to be displayed after a link is selected and before the target portlet content displays. When configuring the Preload notice portlet, you are able to specify when the notices should appear and the maximum number of times it should display.
Extended Properties portlet

The Extended Properties portlet enables the configuration of specific device-type properties that mobile devices must have in order to display pages. It may restrict image capability, color capability, and audio output capability.
Extended Properties

Page: MWPChangeOfAddressSample

Device requirements
There are no device requirements defined for this node.

Edit device requirements

Icon Resource

Icon URL

Extended Properties

Edit device requirements for Page: MWPChangeOfAddressSample

☐ Limit use by general device characteristics such as manufacturer or device capabilities
☐ Limit use to specific device model

Next  Cancel
Developing mobile portlets

This section will provide an overview of how to use the Mobile Portal Toolkit to create XDIME portlets and how to use the associated MCS tools to design the user interface.

Mobile Portal Toolkit
1. Create a new XDIME portlet *(File->New->Portlet Project)*

   ![Portlet Project Creation](image)

   - **Project name:**
   - **Contents:**
     - Like default
   - **Target Runtimes:**
     - WebSphere Portal 6.1
   - **EAR Membership:**
     - Add project to an EAR
   - **EAR Project Name:**
     - EAR_1
   - **Portlet Settings:**
     - Portlet API:
       - JSR 168 Portlet
       - Standard portlet API according to the JSR 168 (SR 168)
   - **Create a portlet:**
     - Portlet name:
     - Portlet type:
       - Mobile Portlet
       - Mobile XHR160 portlet

2. View project hierarchy and files (for example, jsps for xdime) created under the WebContent folder

3. Create a MCS layout policy (right click on mcs-policies folder *(New > Other > MCS > Layout)*)
4. Design the layout of your portlet and specify device dependent policies
5. View a sample layout policy for the Change of Address portlet.
6. Edit the XDIME JSP to reference the layout elements defined above (See the sample Change of Address View JSP below).
7. Test and Deploy your portlet.
**MPA client framework**
The Client Framework is a platform that you can use to create XDIME 2 applications with a rich client-side user interface, for a wide range of devices. The client side of Client Framework is targeted for the following:

- Opera 8.6 and higher
- Nokia OSS browser for S60 DP3 phones
- Access NetFront browser v3.4
- Firefox
- Safari 2.0
- Microsoft Internet Explorer 6.0 and up

The Client Framework consists of a server side and a client side. On the server side, a combination of XDIME 2 markup and policy values to access the widget functions is used. In addition, you can include JavaScript in your markup to invoke functions in the client library. On the client side, a widget library consists of JavaScript that implements functions. This portlet is a showcase sample that is usable with a limited set of mobile devices.

To Setup the Client Framework sample portlet:
1. Ensure that the device for simulation supports AJAX to display AJAX widget functionality *(Note: Non-Ajax widgets are available for display as well)*
2. If the sample Client Framework portlet is not installed by default, the WAR file can be found in (wp_root/wp.mobile/wp.mobile.mwp/installableApps/Client-framework.war) along with the archive file (projects.zip). Copy the unzipped archive folder to wp_profile_root/installedApps/node_name/wps.ear/wps.war/
3. Restart the server and add portlet to a page.

**MPA widgets**
The Mobile Portal Client Framework sample portlet showcases XDIME 2 widgets separated into four categories: AJAX, non-AJAX, Effects, and Building Block. There are over 50 widgets to provide a richer client experience. See screen capture of Ticker Tape Text widget (sideways scrolling display of text) below.
Client framework

- Carousel
- Dynamic menu
- Fold Expander
- Folding menu
- Highlight Navigation
- Popup
- Progress Bar
- Validator
- Validator with popup
- Picker
- Wizard
- Wizard with popup
- Image Camera
- Image Picker
- Timer

The Text

Text begins to scroll
Sample scenario
An existing River Bend LWCM library was imported into WebSphere Portal. Currently, the website only displays content on a PC-based browser which supports html. To provide an end-to-end scenario, we will initially walkthrough steps of how to create content in LWCM for display on a PC based browser, focusing on content creation for the Home page. For mobile devices, we focus on the content created for the three middle portlets. Next, we will describe steps to update the existing authoring template to support non-HTML
type field elements. Then, we create a new presentation template that will use XDIME markup to allow content to be delivered to mobile devices seamlessly. Content will then be able to display simultaneously on a HTML PC-based browser and via mobile devices.

Delivering mobile content

In order to deliver mobile content to mobile devices, you must have a working portal and Lotus Web Content Management server. The LWCM local rendering viewer portlet should be updated to support XDIME (See link in What Is Mobile Portal Accelerator section for steps). Also, within the LWCM server, you must have created authoring and presentation templates, workflow, site and site areas. See the following: http://www.ibm.com/developerworks/websphere/library/techarticles/0703_smit/0703_smit.html.
Integrating MPA with Lotus Web Content Management
Summary of steps
1. Create a library to contain all the assets
2. Assign access roles to library and item types
3. Configure WCM authoring portlet to select available libraries to display to user
4. Create a workflow to manage the content approval process
5. Create the site structure to organize your content
6. Create an authoring template to define your content
7. Create a presentation template to format the display of the content
8. Create your content
9. Configure the content and template mapping relationships
10. Configure HTML supported only JSR286 Web Content Viewer portlet to display content on PC-based browser
11. Update existing authoring template to support non HTML type elements
12. Create new XDIME presentation template using default layout
13. Configure XDIME supported Web Content Viewer portlet to display content on mobile devices
14. Download mobile device emulators/simulators
15. Login to mobile device to view content

Create a new WCM library
1. Login as administrator
2. Click Administration to open the administration portlet
3. Go to Portal Content > Web Content Libraries
4. Click Create new library

Assign role to library
1. Login in as administrator
2. Click Administration to open the administration portlet
4. Click on the RiverBend library to set permissions for a role.

Assign role to item types

Library Resources RiverBend

Set the permissions on the item types for this web content library.
Configure WCM authoring portlet to view library
1. Login as administrator
2. Click Applications
3. Click Content tab. Select menu drop down arrow on Content tab and select Web Content Management
4. Click Configure link (right side of page)
5. Select library (for example, RiverBend) and click Add
Create workflow
Before creating a workflow, setup the workflow actions and the workflow stages

Workflow Actions
1. Click New and select Workflow Actions > Publish Action
2. Enter values for a Publish workflow Action: Name, Display title, Description, and so on.
3. Create additional workflow actions that may be needed for your workflow

**Workflow stages**
1. Click **New** and select **Workflow stages**
2. Enter values for Workflow stage: Name, Display title, Description, Workflow security and so on.
3. Select Workflow Actions created in Step 2 to execute when entering or exiting this stage of the workflow (for example, when entering the Publish workflow stage)

4. Enter additional stages for your workflow that may be needed
1. Click **New** and select **Workflow**
2. Enter values for Workflow: Name, Display title, Description, and so on.
3. Select Workflow stages created in Step 5 to execute in your workflow (that is, Approval workflow, and so on)
Follow these steps to set up the container for the Web site layout

1. Click **New** and select **Site**
2. Enter values for Site: Name, Display title, Description, and so on.
Create site area
Follow these steps to create the page layout for the Web site.
1. Click New and select Site Area
2. Select location for Site Area ("Home" page) (First child is the default) in your Site (for example, River Bend)
New Site Area

Select a parent site or site area.

Library: RiverBend

Placement of new item:

- First child
- Last child
- Before specified child
- After specified child

OK  Cancel

3. Next, enter values for a Site Area (for example, Home): Name, Display title, Description, and so on.
4. Repeat steps 2 and 3 to create additional areas of your site (for example, News, Company, Beverages, Food, and so on) and then select Site Areas > By Site (for example, River Bend) to verify the site structure
Create an authoring template
Follow these steps to specify form content and elements:

1. Click New and select Authoring Templates
2. Enter values for: Name, Display title, Description, and so on.
3. Click link to create Default Content Settings
4. Click Manage Elements to begin creating elements
5. Select Element Type, and enter Name, Display title, Description, and so on.
6. Click OK after elements have been Added
7. Click link to Default Content Settings (to preview previously added entry fields)
8. Set the Workflow
9. Click Save
10. Create additional Authoring Templates required for content
Create HTML presentation template for home page
1. Click **New** and select **Presentation Template**
2. Enter values for: Name, Display title, Description, and so on.
3. Enter HTML/LWCM markup elements to format content

- `<Element context="current" type="content" key="Summary"/>
  <br/>
  <Element context="current" type="content" key="Body"/>
  <br/>`
- Content in Summary (Text field) and Body (Rich Text field) fields will be formatted for display
Create content
1. Click **New** and select **Content**
2. Select **Authoring Template** for content
3. Enter values for: Name, Display title, Description, and so on.
4. Enter content in a selected fields (Summary, Body)
5. Repeat steps as necessary
Identification

* Name: Welcome
  
  Display title: Welcome
  
  Description: Welcome to the River bend Tea and Coffee company.

Type: Content

Library: RiverBend

Authors:

Owner:

Profile

Content Properties

Content

Menu
Component: Select a menu to display.

Select Component

Headline or summary

Enter a summary

Welcome to the River bend Tea and Coffee company
Create template mappings for default content
1. Select Site Areas > By Title (where content will be created)
2. Navigate to Site Area of content (for example, Home)
3. Click on Home and select Edit
4. Click Edit Template Mapping
5. Click Add button
6. Select Authoring Template & Presentation Template
Configure portlets to display LWCM content on PC based browser
(Note: The following steps may be omitted if you have existing pages setup from the library import)

1. Log in to WebSphere Portal as administrator.
2. Click the Administration tab to open WebSphere Portal Administration. Click Portlet Management.
3. Click Portlets.
4. In the Search window, type Web and click Search.
5. Locate the Web Content Viewer JSR286 portlet and click Copy Portlets.
6. Enter Name for portlet.
7. Click OK.
8. From the WebSphere Portal page, select Manage Pages.
9. Expand **Content Root > RiverBend > Home**.
10. In the Type of Page section, select **HTML**.
11. Click **OK**.
12. In the left panel under Manage Pages, select **Home**
13. In the Portlets section of the right panel, click **Edit**.
14. In the left panel, click **Add portlets**.
15. Set the search by Title starts with and type Web Content Viewer JSR286 portlet name
16. Select portlet and click **OK**.
17. Click the down arrow next to portlet from layout screen or on page of portlet and select **Edit Shared Settings**

18. In the Content section, click **Edit**.
19. In the Sites: section, select the content and Alternate Presentation Template to display
20. In the Broadcast Links To section, select **Dynamically select a Web Content page**
21. In the Receive Links From section, select the **Other portlets** and this portlet option
22. Click **OK**.
23. Display content in Browser after adding content for additional JSR286 portlets

**Display LWCM content on mobile devices**

**Update existing authoring template to handle non HTML elements**

1. Select existing Authoring Template (for example, General)
2. Click **Edit > Manage Elements** to create an additional Text element to contain the same text as the Body Rich Text element field
3. Select **Element Type**, and enter Name, Display title, Description
4. Position new field, for example, BodyMP, under Rich Text element type
5. Click **OK** after elements have been Added

<table>
<thead>
<tr>
<th>Element type</th>
<th>Name</th>
<th>Display title</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component Reference</td>
<td>Menu</td>
<td>Menu</td>
</tr>
<tr>
<td>Text</td>
<td>Summary</td>
<td>Headline or summary</td>
</tr>
<tr>
<td>Rich Text</td>
<td>Body</td>
<td>Main body</td>
</tr>
<tr>
<td>Text</td>
<td>BodyMP</td>
<td>Body Mobile</td>
</tr>
<tr>
<td>Image</td>
<td>Image</td>
<td>Article image</td>
</tr>
<tr>
<td>File Resource</td>
<td>Mobile Image</td>
<td>Mobile Image</td>
</tr>
<tr>
<td>Link</td>
<td>Link</td>
<td>Link</td>
</tr>
<tr>
<td>Component Reference</td>
<td>Navigator</td>
<td>Navigator</td>
</tr>
</tbody>
</table>

6. Click link to Default Content Settings (to preview previously added element type)
Create new XDIME presentation template
1. Click New and select Presentation Template
2. Enter values for: Name, Display title, Description, and so on.
3. Enter XDIME/LWCM markup to format content
4. Click Save

<canvas layoutName="mcsi:policy('default', '/wp_xhtml_webclip.mlyt')"
type="portlet">
  <pane name="Panel">
    <img url=""/>
  </pane>
  <pane name="Pane2">
    <h3><Element context="current" type="content" key="Summary"></h3>
    <Element context="current" type="content" key="BodyMP"/>
  </pane>
  <pane name="Pane3">
    <img url=""/>
    <img url=""/>
    <img url=""/>
  </pane>
</canvas>

Notes:
- a default layout policy is being used
- the component id is auto-generated by LWCM
- "htmlencode=true" prevents the following SAXPARSER error message from occurring on mobile devices when working with LWCM component element types for images:
  "org.xml.sax.SAXParseException: The reference to entity "CACHEID" must end with the ';'
deiminator."

New XDIME presentation template created

- Home Page PT XDIME
- Home Page Presentation Template

Update content
1. Select Authoring Template for content
2. Enter duplicate content in selected fields (for example, BodyMP)
Our company River Bend has been established since 1971, for 30+ years we've been in the business of making our own special blends of tea and coffee. Today River Bend is a thriving business and community and we welcome you to our site, and hope you enjoy your experience when you next visit one of our stores.

Take a look at our specials menu - we've cooked up some tasty deals and lunch time specials.

Update template mappings for default content
1. Select Site Areas >By Title (where content will be created)
2. Navigate to Site Area of content
3. Click on Home and select Edit
4. Click Edit Template Mapping
5. Click Add button
6. Select Authoring Template & Presentation Template
Configure portlets to display LWCM content on mobile devices

1. Log in to WebSphere Portal as administrator.
2. Click the Administration tab to open WebSphere Portal Administration.
3. Click Portlet Management.
4. Click Portlets.
5. In the Search window, type Web and click Search.
6. Locate the Web Content Viewer portlet and click Copy Portlets.
7. Enter Name for portlet (for example, Mobile Web Content Viewer)
8. Click OK.
9. From the WebSphere Portal page, select Manage Mobile Pages or Manage Pages.
10. Expand Content Root > Home
11. In the left panel under Manage Mobile Pages or Manage Pages, click the New Page icon.
12. Under Title, type Mobile Test Page
13. For the Theme, select Mobile Portal
14. In the Type of Page section, select the XDIME option

Note: If the xdime markup language is checked without support for html, the Mobile Test page will not display in the PC-web based browser. The HTML setting should remain until testing of web site is not necessary.
15. Click OK.

Note: By default, only one root node (page) is handled by a mobile device, so make any previous pages that support XDIME inactive or your page will not display on mobile device.

16. In the left panel under Manage Mobile Pages or Manage Pages, select Mobile Test Page.
17. In the Portlets section of the right panel, click Edit.
18. In the left panel, click Add portlets.
19. Set the search by Title starts with and type Mobile RiverBend Web Content Viewer portlet name (Note: add Mobile Login portlet to layout as well)
20. Select portlet and click OK.
21. Click Done.
22. Click the down arrow next to Mobile Test Page and select Edit Shared Setting.
23. In the Content section, click Edit.
24. In the Sites section, select the content.
25. In the Alternate Presentation Template section, click Edit.
26. Select Home Page PT XDIME.
27. Click OK.
28. In the Broadcast Links To section, select None.
29. In the Select a target portal page section, select Mobile Test page.
30. In the Receive Links From section, select the Other portlets and this portlet option.

31. Click OK.
32. Login via Mobile Login portlet to access website
Welcome to the River Bend Tea and Coffee Company

Our company, River Bend, has been established since 1971. For 30 years, we’ve been in the business of making our own special blends of tea and coffee. Today, River Bend is a thriving business and community, and we welcome you to our site, and hope you enjoy your experience when you visit. Visit one of our stores. Take a look at our special menu – we’ve cooked up some tasty deals and lunch and dinner specials.
Welcome to the River Bend Tea and Coffee company

Our company River Bend has been established since 1971, for 30+ years we've been in the business of making our own special blends of tea and coffee. Today River Bend is a thriving business and community and we welcome you to our site, and hope you enjoy your experience when you next visit one of our stores. Take a look at our specials menu - we've cooked up some tasty deals and lunch time specials.
Download emulator and simulators
1. To test with the User Agent Switcher AddOn Extension Tool of Fire Fox, see the following for details: https://addons.mozilla.org/en-US/firefox/?application=firefox
2. To test with an i-phone emulator: http://www.genuitec.com/mobile/download.html
3. To test with the Black Berry emulator displayed below: https://www.blackberry.com/Downloads/entry.do?code=060AD92489947D410D897474079C1477 (Note: Download the BB Email and MDS Services Simulator Package. Start MDS Services Simulator before running the device simulator. RIM BB will work with Simulator Devices for v4.1 (for example, 7130e, 8700c, and so on) and up. More current simulator devices, v5.0, may not run using the MDS Services Simulator Package)

Image conversion service
The Image Conversion Service (ICS) will be used to convert the images to different formats and pixel depth. It can also reduce the color depth for a smaller color palette. We can use this to scale the image and we can also limit the output file to a particular size that can be handled by the device. The ICS may use the Caching Proxy of the WebSphere Application Server Edge Components as both a forward and reverse proxy. Using this feature it can cache source images fetched from the Web Server by ICS prior to conversion. It can also be installed on a separate WebSphere Application Server than WebSphere Portal and may clustered.

Typical Image Conversion Service flow
In the above service flow, when the Multi-Channel Server (MCS) encounters an image that should be converted by ICS, it includes a device-specific image tag specifying the URL of the ICS servlet. The URL also specifies where to find the image to be transcoded and the transcoding rule to apply when converting the image to a format that is supported by the mobile device.

1. When the mobile device's browser requests an image that should be transcoded, the device sends a request to the ICS servlet.

2. Using the image path and server information in the URL, and information from the ICS configuration file, the ICS servlet retrieves the original image from the image server. ICS uses the transcoding rule and any
transcoding parameters included in the URL to transform the image.

3. The ICS servlet returns the transcoded image to the mobile device.

**Virtual portal support**

Virtual Portal support allows support for multiple brands and mobile virtual network operators. Virtual portals enable you to deploy additional portals quickly on an existing infrastructure. With virtual portals, you can use a single installation of Mobile Portal Accelerator to deploy multiple portals with different URLs, anonymous pages, user groups, and look and feel. Advantages of using virtual portals include reusing existing hardware, and simplified administration of multiple portals.

**MPA device subscription service**

A separately priced, related subscription service—IBM Mobile Portal Accelerator Device Update entitles customers to receive regular updates to the mobile device repository that include new mobile device definitions and updates as they become available from device manufacturers. The Mobile Portal Accelerator and Mobile Portal Accelerator Device Update service helps organizations quickly deliver personalized content, applications and services to mobile employees, customers and subscribers on the wide variety of mobile devices they use - even the latest devices available - without redeveloping their mobile Web applications.