PORTAL SETUP WIZARD

A portal can be automatically generated without requiring user (150) coding of the portal. An interface can be provided to receive configuration information from a user (150). A template can be modified using the supplied configuration information to build a working portal in accordance with a user’s requirements. A portal definition can be automatically generated as well as accompanying resources including portlet definitions, Java server pages, web application descriptors, and image files.
For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.
PORTAL SETUP WIZARD

COPYRIGHT NOTICE

[0001] A portion of the disclosure of this patent document contains material which is subject to copyright protection. The copyright owner has no objection to the facsimile reproduction by anyone of the patent document or the patent disclosure, as it appears in the Patent and Trademark Office patent file or records, but otherwise reserves all copyright rights whatsoever.

PRIORITY CLAIM

[0002] The present application claims the benefit of the filing date of U.S. Provisional Patent Application No. 60/376,913 [Attorney Docket No. BEAS-01250US0], filed May 1, 2002, incorporated by reference herein.

CROSS-REFERENCE TO RELATED APPLICATIONS

[0003] The following application is cross-referenced and incorporated herein by reference:


FIELD OF THE INVENTION

[0005] The present invention is directed to technology for automatically generating portals.

BACKGROUND

[0006] Portals can provide access to information networks and/or sets of services through the World Wide Web and other computer networks. Portals can
provide a single point of access to data and applications, making them valuable to developers, businesses, and consumers alike. A portal can present a unified and personalized view of enterprise information to employees, customers, and business partners. In many implementations, portal applications can include web applications designed as a portal.

[0007] Portals are capable of presenting multiple web applications within a single web interface. In addition to regular web content that can appear in a portal, portals provide the ability to display portlets (self-contained applications or content) in a single web interface. Portals can also support multiple pages with tag-based navigation for accessing the individualized content and portlets for each page.

[0008] The generation of portals, however, can be an involved and cumbersome process that requires considerable development time and effort. In order to create a working portal, an application developer may have to create: a portal definition such as a file including Extensible Markup Language ("XML"); portlet definition files for any portlets associated with the portal; Java server pages ("JSPs"); web application descriptors; images such as graphics interchange format files ("GIFs"); deployment descriptors, configuration files, the Java archive files ("JAR") that contain the logic and formatting instructions for the portal application; and any other files necessary for the desired portal application. In addition, the developer may have to determine the appropriate directories in which to place each file and then manually move files to appropriate directories.

**BRIEF SUMMARY**

[0009] Systems and methods in accordance with embodiments of the present invention provide for the automatic generation of portals. In one embodiment of the present invention, an interface is provided to facilitate the gathering of information from a user. Using information gathered from a user, embodiments of the present invention facilitate the automatic generation of code and files to create a working portal.
[0010] Other features, aspects, and objects of the invention can be obtained from a review of the specification, the figures, and the claims.

**BRIEF DESCRIPTION OF THE FIGURES**

[0011] Figure 1 is a high level block diagram of various software components of a computer network supporting a configurable electronic business system in accordance with an embodiment of the present invention.

[0012] Figure 2 illustrates a sample screen shot of an application displayed to a client in a portal-style presentation employing multiple portal pages and portlets.

[0013] Figures 3A-3B illustrate a hierarchy of objects which can be used to implement a portlet wizard in accordance with an embodiment of the present invention.

[0014] Figures 4A-4B illustrate a flowchart of a method for creating a portlet in accordance with an embodiment of the present invention.

[0015] Figure 5 is a flowchart of a method for implementing web services in a portlet in accordance with an embodiment of the present invention.

[0016] Figure 6 is a flowchart of a method for automatically generating code for implementing a portlet in accordance with an embodiment of the present invention.

[0017] Figures 7A-7L illustrate sample screen shots of a user interface for creating a portlet in accordance with an embodiment of the present invention.

[0018] Figures 8A-8C illustrate sample portlet code generated in accordance with an embodiment of the present invention using form-based code generation.

[0019] Figure 9 illustrates sample portlet code generated in accordance with an embodiment of the present invention using web service call-based code generation.
Figure 10 illustrates sample portlet code generated in accordance with an embodiment of the present invention using web service interface-based code generation.

Figure 11 illustrates a flowchart of a method using a wizard-style interface to facilitate the automatic generation of portals in accordance with an embodiment of the present invention.

Figures 12A-12H illustrate exemplary screen shots of a user interface for generating a portal in accordance with an embodiment of the present invention.

Figure 13 illustrates a flowchart of a method in accordance with one embodiment of the present invention of automatically generating a portal.

Figures 14A and 14B illustrate sample code for an XML portal definition generated in accordance with an embodiment of the present invention.

Figure 15 illustrates sample code for an XML portlet definition generated in accordance with an embodiment of the present invention.

DETAILED DESCRIPTION

Figure 1 is a high level block diagram of various software components of a computer network supporting a configurable electronic business system in accordance with the present invention.

Business platform 130 is an electronic business platform, such as WEBLOGIC PLATFORM™ 7.0 available from BEA Systems, Inc. of San Jose, California. Business platform 130 is capable of hosting one or more enterprise applications 140 which provide services to client 150 over network 110. Enterprise application 140 can include one or more web applications 145. In one embodiment, business platform 130 is implemented on a web server which implements JAVA™ 2 Platform, Enterprise Edition (J2EE) version 1.3, available from Sun Microsystems, Inc. of Santa Clara, California.
center 120 and provides a wizard-style interface that facilitates the setup of web service 155 to be interfaced with application 135 and displayed to client 150. In various embodiments, wizard 125 facilitates the generation of portals and/or portlets and the integration of the same. In various embodiments, data can be synchronized between control center 120 and business platform 130. As a result, control center 120 can operate as a swing application, capable of configuring multiple business platforms, without requiring control center 120 to be in communication with the business platforms/servers during configuration.

[0029] Web service 155 is a web service as understood by those skilled in the art. Web service 155 can be implemented as a software component that can be invoked by one or more operations across the Internet. For example, web service 155 can receive a stock quote request from an enterprise application such as application 135 and return the requested information back to application 135. Thus, application 135 can utilize the functionality of web service 155 remotely over network 110 without having to duplicate the same services locally within application 135.

[0030] Enterprise applications can communicate with web services using one or more of an emerging group of standards that govern their description and interaction. The Simple Object Access Protocol (SOAP), which uses a combination of XML-based data structuring and the Hyper Text Transfer Protocol (HTTP) to define a standardized method for invoking methods in objects that are distributed in diverse operating environments across the Internet. The Universal Description, Discovery, and Integration (UDDI) project provides a structure for XML-based online business registries where businesses register and advertise their web services. The project creates a platform-independent, open framework for describing services, discovering businesses, and integrating business services using the Internet. The Web Services Description Language (WSDL) is an XML language that is used to describe a web service and to specify how to communicate with the web service.
Referring to Figure 1, client 150 is a software component that requests services provided by application 135. Web service content and/or other content provided by application 135 can be displayed to client 150 in a portal-style display. Such a display can employ multiple portal pages and portlets. Figure 2 illustrates a sample screen shot of an application displayed to a client in a portal-style presentation. Several portlets are included in the portal page of Figure 2, including a LOGIN portlet, a MY TASK LIST portlet, and a BOOKMARK portlet.

Portals can provide access to information networks and/or sets of services through the World Wide Web or other computer networks. These networks can range from broad interconnections of computing systems such as the Internet to localized area networks including a few computers located in close geographic proximity such as a home or office. Portal applications can include web applications designed as a portal.

Portlets can be implemented as java server pages (JSP’s) with XML-based metadata that fit into a portal. Portlets can utilize various types of display code to display highly focused information directed to a specific user or user group, having a portal as its container. Portlets are comprised of portlet components which include portlet attributes (i.e. whether the portlet is editable, floatable, minimizable, maximizable, helpable, mandatory, has defaults minimized, or whether login is required) and portlet layout elements or components (i.e. banner, header, content, and footer sections). In one embodiment, a portlet is defined by a file that contains a portlet's XML-based metadata, which is created and edited by control center 120. Portlets can also be associated with portlet resource files including stub JSPs (one for each portlet layout element) and image files created and saved to a local file system by wizard 125 of control center 120. In one embodiment, portlet resource files do not include the portlet definition itself or a file implementing webflow, as further described herein.

Figures 3A-B illustrate a set of objects that can be used to implement a wizard 125 which can facilitate the setup of web services to be
interfaced with applications and to automatically generate portlets in accordance with the present invention.

[0035] In accordance with one embodiment of the present invention, wizard 125 automates the creation of a portlet and reduces the number of steps otherwise needed for portlet creation. In the simplest case, a developer can create a new portlet without providing any input and simply accept default values. Wizard 125 includes the ability to create a portlet that consumes a web service 155. This allows developers to discover a web service and generate a portlet JSP implementation that can interact with the web service. In one embodiment, the generated code provides a default HTML form for entering input parameters, and a default HTML output of the results of the service invocation. In another embodiment, when project data from control center 120 is synchronized with business platform 130, the JSP files that constitute a portlet are synchronized as well. In another embodiment, the information entered into wizard 125 is stored as variables in a wizard framework that persist until the portlet created with wizard 125 is saved. Wizard 125 can be implemented using software compliant with J2EE.

[0036] In accordance with another embodiment of the present invention, wizard 125 automates the creation of a portal and reduces the number of steps otherwise needed for portal creation. Wizard 125 provides the ability to create a working portal resulting in a client-side file structure including all portal related J2EE and control center files required on the client-side. Wizard 125 can extract J2EE files including JSP’s, html files, images, etc. from a user selected template and save them in a user selected location; modify various XML files as needed; and create and save the necessary portal, portlet, and webflow definition files into the control center “project” directory structure. Wizard 125 can be implemented using software compliant with J2EE.

[0037] Referring to Figure 1, control center 120, business platform 130, web service 155, and client 150 can communicate over network 110. Moreover, communication between the components of Figure 1 can be performed using any
of the applicable data formats and/or protocols known in the art including, but not limited to: TCP/IP, XML, and/or SOAP. In the embodiment depicted in Figure 1, software components 120, 130, 155, and 150 are illustrated as communicating over network 110. In alternate embodiments, some or all of the components can communicate with each other directly (not shown), rather than over network 110.

**Portlet Generation**

[0038] Figures 4A-4B illustrate a flowchart of a method for creating a portlet in accordance with the present invention. In step 410, a user (such as a business platform administrator) decides to create a new portlet. In step 415, a new portlet dialog screen allows the user to specify whether or not the portlet should be created using a wizard-style interface. If the wizard-style interface is not selected, then the method proceeds to step 465 where a dialog-based editor is opened for further interaction by the user. If the wizard-style interface is selected, then wizard 125 is launched and the method proceeds to step 420. Figure 7A illustrates a sample screen shot of a new portlet dialog screen. In the sample screen shot of Figure 7A, the wizard-style interface is selected by default.

[0039] In step 420, wizard 125 displays a portlet name screen to the user. Figure 7B illustrates a sample screen shot of a portlet name screen. The user can input the name and description of the portlet to be created by wizard 125. A drop-down menu is also provided, permitting the user to select the portal in which the portlet will be implemented. In one embodiment, the values displayed in the drop-down menu are obtained by performing a search for existing portal definitions. A default value can also be automatically provided for the user. In another embodiment, wizard 125 validates the selection of portals in the drop-down menu based on whether portals are currently open or are identified as read-only. After the portlet name screen is completed, the method proceeds to step 425. Alternatively, the user can be required to enter a unique and valid name in the
portlet name text field and select a portal with which to associate the portlet before proceeding to step 425.

[0040] In step 425, wizard 125 displays a portlet components screen to the user, allowing the user to specify the components to be displayed in the new portlet. For example, in one embodiment, portlets can include a titlebar, banner, header, footer, help page, and content. Figure 7C illustrates a sample screen shot of a portlet components screen. In the sample screen shot, the titlebar and content components have been automatically selected. In one embodiment, if header or footer components are selected, then alternate header or footer JSP files are created in addition to default header or footer JSP files.

[0041] In step 430, wizard 125 displays a portlet content types screen to the user, allowing the user to specify whether the portlet will be implemented as a basic portlet, webflow-enabled portlet, or web service-enabled portlet. Figure 7D illustrates a sample screen shot of a portlet content types screen. Basic portlets provide content, but do not implement logic to switch to alternate page views and do not provide web service content. In one embodiment, the content of basic portlets is static.

[0042] In contrast, webflow-enabled portlets can implement logic to switch to alternate page views, but do not provide web service content. For webflow-enabled portlets, separate content files are linked to each other and a webflow file is generated which describes the logic linking the pages together. In one embodiment, webflow-enabled portlets created by wizard 125 permit portlets to switch between exactly two page views.

[0043] Web service-enabled portlets can display web service content. In various embodiments, such portlets can permit clients to input data to be sent to web services as operations. If the user selects web service content, then the method proceeds to step 435 where the steps of Figure 5 can be performed. Otherwise, the method proceeds to step 440.
In step 440, wizard 125 displays a resource files location screen, allowing the user to specify where generated portlet files can be stored, such as a path on the user’s local computer which is running control center 120. Figure 7E illustrates a sample screen shot of a resource files location screen. The user can accept a default path, enter a path, or browse for a path. In one embodiment, wizard 125 provides a default path based on an educated guess of the root directory for application 135.

In step 445, wizard 125 displays a summary screen, allowing the user to view a list of portlet resource files (including their locations) to be generated in response to the data provided to wizard 125 in the preceding steps. In various embodiments, the list can be copied to a “clipboard” and/or printed. Figure 7F illustrates a sample screen shot of a summary screen.

When the user clicks the “NEXT” button on the Summary screen, wizard 125 closes and files for the new portlet can be created and/or modified (step 450). In one embodiment, template JSPs and a template portlet definition are modified with configuration information supplied to wizard 125 by the user. In other embodiments, the steps of Figure 4 can be performed to generate and/or modify the files for the new portlet. Appropriate client-side files for the portlet are generated and/or modified and stored in the location specified in step 440. A portlet definition which can contain XML based meta-data can be generated in accordance with user supplied configuration information. The portlet definition can specify the various resource files for the new portlet, including, for example: titlebar.jsp, banner.jsp, header.jsp, altheader.jsp, content.jsp, content2.jsp (if, for example, the two page with webflow option was previously selected); footer.jsp; altfooter.jsp; and help.jsp. Other resource files including images and webflow files can also be generated and/or modified in this step. In one embodiment, an XML definition of the portal containing the generated portlet is modified to reflect the new portlet. The method then proceeds to step 455.
In step 455, a next steps screen is displayed that confirms the successful creation of a portlet and provides the user with options for tasks they are likely to perform next. These can include, but are not limited to: editing the definition of the previously selected portal, editing the definition of the generated portlet, or reading documentation for related tasks. Figure 7G illustrates a sample screen shot of a next steps screen.

If the user decides to edit the portlet definition, a portlet editor is launched (step 465), allowing an XML-based portlet definition to be edited. In one embodiment, portions of the portlet definition relating to web services are not edited with the portlet editor. If the user chooses to edit the portal definition, a portal editor is launched (step 470), allowing an XML-based portal definition to be edited. For example, the portal definition can be edited to identify which portlets are assigned to the portal page. In various embodiments, the portlet and/or portal editors can be swing-based editors, wherein the user edits the portal and portlet definitions (i.e. edit portal attributes) by interacting with dialog boxes. If the user chooses to read documentation for related tasks, portlet documentation is opened (step 460).

Figure 5 is a flowchart of a method for implementing web services in a portlet in accordance with the present invention. In one embodiment, the method of Figure 5 can be performed at step 435 of Figure 4A. In step 510, a server location screen is displayed, allowing the user to identify a file system path for the location of business platform 130. This file system path can identify the location of a java archive file ("JAR") which can be used to generate server-side proxies, as further described herein. The file system path can be validated before the method proceeds to the next step. Figure 7H illustrates a sample screen shot of a server location screen.

In step 515, server-side proxies can be generated to allow application 135 to communicate with web service 155. The proxies, such as proxy java code, can serialize and deserialize SOAP messages exchanged between a web
service and the generated portlet. The proxies can be generated by proxy generation facilities of business platform 130 using a WSDL file (often referred to as “a WSDL”) for web service 155. It is contemplated that various proxy generation methods can be used, as known and understood in the art. In one embodiment, a proxy generator such as Apache™ Axis, available from The Apache Software Foundation™ of Forest Hill, Maryland, can be used (Apache™ and The Apache Software Foundation™ are trademarks of The Apache Software Foundation™). In other embodiments, proxy generation can be performed by a proxy generator included within business platform 130. In an alternate embodiment, proxy generation step 515 can be skipped in favor of allowing application 135 to communicate with web service 155 using XML and XSLT transforms.

In step 520, a generated code types screen is displayed which allows the user to select from several methods of generating portlet JSP files. In various embodiments, form-based generation, web service call-based generation, and web service interface methods are available. Figure 71 illustrates a sample screen shot of a generated code types screen.

Form-based generation creates portlet JSPs from portlet JSP templates. The resulting JSPs provide a portlet capable of calling web service operations based on parameters entered into wizard 120. If form-based generation is selected, wizard 125 can display a description and example of the output of form generation. A portlet having a form field for the user to pass data to a web service can be created.

Web service call-based generation creates portlet JSPs from portlet JSP templates. However, the resulting JSPs provide a portlet capable of calling web service operations based on parameters inserted into the JSP code by a user at a later time. For example, JSPs can be created with parameters having "_REPLACE_ME_" dummy values that a user subsequently replaces with actual values. In various embodiments, the replacement values of these parameters can also be retrieved from a client request, session, and/or client profile. If web service
call generation is selected, wizard 125 can display a description and example of the output of the web service call generation below a drop-down menu.

[0054] Web service interface-based generation creates portlet JSPs which implement any number of web services with any number of web service operations. In various embodiments, JSPs created by this method can be further modified by developers. For example, a developer may want to chain two web services by passing the output of a first service to a second service. Using interface-based generation, the developer can modify the necessary JSPs to establish the chain.

[0055] In step 525, a web services selection screen is displayed which allows the user to select available web services and operations for use in the generated portlet. In certain embodiments, if form-based generation or call-based generation was selected in step 520, only one web service and one operation may be selected. Alternatively, if interface-based generation was selected, then the user can select multiple web services, and need not select any operations (all operations for the Web services can be automatically included). In one embodiment, the lists of web services and operations shown by wizard 125 can be created from a directory of WSDL files. Figure 7J illustrates a sample screen shot of a web services selection screen.

[0056] If a user desires to edit the list of web services (step 530), an add/remove web services screen can be displayed where the user can add or delete web services. Figure 7K illustrates a sample screen shot of a web services selection screen. In various embodiments, the user can locate WSDL files by entering a WSDL URL, browse for WSDL files in a file system, and/or access a UDDI directory to retrieve a list of web services. The WSDL file for existing web services can also be updated. When a user attempts to add a web service, the corresponding WSDL file can be validated to determine whether proxy code can be generated based on the WSDL information. In yet another embodiment, the WSDL for added web services can be stored locally at control center 120, thus allowing the control
center 120 to configure portlets without retrieving and/or validating a WSDL file multiple times.

[0057] A preview of code to implement the portlet can be generated (step 540) and displayed to the user with a code preview screen (step 545). In one embodiment, the user can edit the previewed code and substitute actual values for any variables shown in the code. Additionally and/or alternatively, the user can perform full text editing of the previewed code, employ code completion methods as known in the art, and view the code in a syntax-colored format. In various embodiments, the code displayed in step 545 can, but need not, be validated. Figure 7L illustrates a sample code preview screen.

[0058] Figure 6 is a flowchart of a method for automatically generating code for implementing a portlet in accordance with the present invention. In various embodiments, the method of Figure 6 can be performed at step 450 of Figure 4B. In one embodiment, the type of code generation previously selected at step 520 can be determined at step 610. If form-based generation is selected, the method proceeds to step 615 where a portlet template is accessed. The template can be modified in accordance with the web services information previously determined by wizard 125 (step 620). A portlet generated using form-based generation can provide a form entry field for the user to enter simple variable types in accordance with the specified parameters of a web service which can then be passed to the service. Figures 8A, 8B, and 8C illustrate sample portlet code generated in accordance with form-based code generation of the present invention.

[0059] If web service call-based generation is selected, the method proceeds to step 625 where a portlet template is accessed (step 625) and modified in accordance with the web services information previously determined by wizard 125 (step 630). In step 635, dummy values are inserted for parameters to be passed during web service operations. These parameters can later be replaced by the user with actual values if desired (step 640). Figure 9 illustrates sample portlet code
generated in accordance with web service call-based code generation of the present invention.

[0060] If web service interface-based generation is selected, the method proceeds to step 645. A portlet template can be accessed (step 645) and modified by the user to implement any number of web services with any number of web service operations (step 650). Figure 10 illustrates sample portlet code generated in accordance with web service interface-based code generation of the present invention.

**Portal Generation**

[0061] Figure 11 illustrates a method of user interaction with a wizard-style interface that facilitates the automatic generation of portals in accordance with one embodiment of the present invention. In step 1110, a user (such as a business platform administrator) opens an existing enterprise application in control center 120. In step 1115, the user selects a new portal web application option from, for example, a menu selection in the control center. In step 1120, a new portal dialog screen allows the user to select whether or not the new portal should be created using a wizard-style interface. If the wizard-style interface option is not selected, the method proceeds to step 1125 where a dialog-based portal editor is opened for further interaction by the user. Using the dialog-based editor, the user can create a new portal by specifying existing J2EE resources. If the wizard-style interface is selected, wizard 125 is launched and the method proceeds to step 1130. Figure 12A illustrates a sample screen shot of a new portal dialog screen. In the exemplary screen shot of Figure 12A, the wizard-style interface option is selected by default.

[0062] In step 1130, wizard 125 displays a portal name screen to the user. Figure 12B illustrates an exemplary screen shot of a portal name screen. The user can enter a name and description for the portal to be created by wizard 125. The portal name screen in Figure 12B includes fields for portal name, web application, and portal description (optional).
[0063] In step 1135, the user decides whether or not to create a new web application. A “NEW” button is provided within the portal name screen of Figure 12B for the user to select if the user wishes to create a new web application to associate with the portal. If the user selects the “NEW” button to create a new portal web application, the method moves to step 1140 where the user can enter a name for the new portal web application. Figure 12C illustrates an exemplary screen shot of a new portal web application screen. After entering a name for the new portal web application, the method moves to step 1150.

[0064] If the user does not select the “NEW” button to create a new web application, the method moves to step 1145 where a pre-existing web application can be selected. The web application field of the portal name screen in Figure 12B allows the user to select a pre-existing web application with which to associate the portal. A browse feature or drop-down menu can be provided to allow the user to browse local directories for web applications. A list of web applications can be found, for example, locally in the current control center project directory.

[0065] After entering a new portal web application name in step 1140 or selecting a web application in step 1145, the method moves to step 1150 where wizard 125 displays a portal template screen. The portal template screen allows the user to select a portal template on which to base the portal. Figure 12D illustrates an exemplary screen shot of a portal template screen. A list of templates with icons and descriptions for the templates can be presented. While three portal templates are shown in Figure 12D, any number of templates can be presented to the user. Additionally, multiple types of portal templates can be presented, including for example, a template for a single portal with one portlet or a template for a portal with multiple portlets. Users can also create their own templates (outside of wizard 125) which can then be presented in the portal template selection screen. The portal wizard can search known directories for portal templates and present these to the user in the portal template selection screen. In one embodiment, the templates are
stored in a sub-directory of the control center installation directory or within a sub-directory of a home directory of business platform 130.

[0066] A template can be a JAR file or a set of directories that contain control center portal information and J2EE portal information. A template can include a portal definition, portlet definitions, an Ant (a java-based build tool available from The Apache Software Foundation™ of Forest Hill, Maryland) build file or other build file, JSP's, images such as GIF files, and other files for portal generation. The following contents can appear inside an exemplary template JAR file (e.g. “myPortalTemplate.jar) or set of directories: a template.xml file ; a template.properties file; a templateIcon.gif file; any additional resources needed by the template; and directories containing any resources needed by the template including JSPs, images, libs, and other directories.

[0067] After selecting a portal template, the method moves to step 1155 where wizard 125 displays a resource files location screen to the user, allowing the user to select a location in which to store resource files. Figure 12E illustrates an exemplary screen shot of a resource files location screen. In this step, the user selects a directory or location in which to place the J2EE resource files. In one embodiment, the portal resource files location defaults to a file system path corresponding to the directory for the enterprise application from the control center project file and the web application selected in step 1145 or 1140. If no default file system path is present, or if the user wishes to place the resource files into an alternate location, the user can enter a file system path or select “BROWSE” to select a file system path. The user can select to have the selected web application exist in a different location from the enterprise application by selecting a path different from that of the enterprise application.

[0068] After selecting a resource files location, the user selects “CREATE” to begin generation of the resource files. In one embodiment, the resource files are generated using Ant. Other build tools can also be used, as known and understood in the art.
In step 1160, wizard 125 displays a summary screen, allowing the user to view a list of portal resource files (including their locations) generated in response to the information provided to wizard 125 in the preceding steps. In one embodiment, the list can be copied to a “clipboard.” Figure 12F illustrates an exemplary screen shot of a summary screen.

In step 1165, wizard 125 displays a next steps screen, allowing the user to select deployment options for the web application created by the wizard in the preceding steps. The user can select between a hot deploy option and a normal deploy option. Additionally, hyperlinks are provided to documentation regarding hot deployment and normal deployment, as well as to instructions on how to manually deploy the portal web application created in the preceding steps, should the user choose the normal deployment option. A link can be provided to information regarding adaptation of the portal to a user’s needs. The user is also presented with the option of viewing the portal definition after selecting “DEPLOY.”

During deployment, configuration information required for proper use of the web application is supplied to the application server. If normal deployment is selected by a user, the user manual deploys the web application. The user can follow instructions for manual deployment by clicking the provided hyperlink to view the instructions. In a manual deployment, the server is stopped in order to provide configuration information for the web application to the server. After the configuration information has been provided, the server can be restarted. If hot deployment is selected, the web application generated by wizard 125 is deployed while the server is running and without manual deployment by the user. If not already connected to a server, wizard 125 can prompt the user to connect to a server. A default server can be determined and presented to the user for selection using the server specified in the control center project.

In step 1170, wizard 125 displays a hot deployment status screen if the user selected the hot deployment option in step 1165. The hot deployment
status screen can display the status of the deployment. Figure 12H illustrates an exemplary screen shot of a hot deployment status screen.

[0073] Figure 13 illustrates a flowchart of a method in accordance with one embodiment of the present invention of automatically generating a portal step 1310, a portal wizard is launched. The portal wizard can be launched in response to a user selection as shown in step 1120 of Figure 11 and the screen shot of Figure 12A.

[0074] In step 1320, appropriate user screens are displayed and portal configuration information is received from user inputs. In one embodiment, the appropriate user screens are displayed and the portal configuration information is received as illustrated in steps 1130 to 1155 of Figure 11 and the screen shots of Figures 12B to 12E.

[0075] In step 1330, a build file is prepared. In one embodiment, the build file can include XML. In another embodiment, the build file is a template Ant XML build file modified according to configuration information input by the user into the wizard. As previously discussed, each template includes an Ant XML build file or other build file. The build file can contain configuration information as XML based meta-data that a builder program can access in order to copy, modify, create, and/or move files in order to generate a portal. The build file (e.g. template.xml) can contain information that specifies: what files to copy in the build process; where the J2EE resource files are to placed; where the set of control center files is to be placed; a template version number and type (e.g. domain template, portal template, etc.); template name; description (for display in wizard 125); the replacement, filtration, and/or substitution of certain strings, variables, or information within certain template files with other strings, variables, or information (e.g., substitute the user entered display name for the stock display name in “web.xml”); optional summary text to display at the end of the wizard; as well as other instructions or information to be used in the portal generation. The information entered by the user and gathered by the wizard can be used to modify the Ant XML build file included within a template so that Ant can build and compile the appropriate project.
In step 1340, a portal is built. In one embodiment, the portal is built by copying and configuring files from a template with information input into the wizard by the user. As part of the build process, the builder (e.g., Ant) can copy over and/or modify the template files including portlet XML files, a portal XML file, JSP’s, images, and other resource files as identified by the configuration information. The copying and/or modification can be carried out in accordance with the information included within the build file. For example, the modified Ant build file can specify that the default portal name be replaced with the name entered by the user in step 1130 during the generation process.

Figures 14A and 14B illustrate sample code for an XML portal definition generated as part of the creation of the portal resource files. Figure 15 illustrates sample code for an XML portlet definition generated as part of the creation of the portal resource files. In one embodiment, the portlet definition is identical to the XML portlet definition included in the template.

The foregoing description of preferred embodiments of the present invention has been provided for the purposes of illustration and description. It is not intended to be exhaustive or to limit the invention to the precise forms disclosed. Many modifications and variations will be apparent to one of ordinary skill in the relevant arts. For example, steps performed in the embodiments of the invention disclosed can be performed in alternate orders, certain steps can be omitted, and additional steps can be added. The embodiments were chosen and described in order to best explain the principles of the invention and its practical application, thereby enabling others skilled in the art to understand the invention for various embodiments and with various modifications that are suited to the particular use contemplated. It is intended that the scope of the invention be defined by the claims and their equivalence.

One embodiment may be implemented using a conventional general purpose or a specialized digital computer or microprocessor(s) programmed according to the teachings of the present disclosure, as will be apparent to those
skilled in the computer art. Appropriate software coding can readily be prepared by skilled programmers based on the teachings of the present disclosure, as will be apparent to those skilled in the software art. The invention may also be implemented by the preparation of integrated circuits or by interconnecting an appropriate network of conventional component circuits, as will be readily apparent to those skilled in the art.

[0080] One embodiment includes a computer program product which is a storage medium (media) having instructions stored thereon/in which can be used to program a computer to perform any of the features presented herein. The storage medium can include, but is not limited to, any type of disk including floppy disks, optical discs, DVD, CD-ROMs, microdrive, and magneto-optical disks, ROMs, RAMs, EPROMs, EEPROMs, DRAMs, VRAMs, flash memory devices, magnetic or optical cards, nanosystems (including molecular memory ICs), or any type of media or device suitable for storing instructions and/or data.

[0081] Stored on any one of the computer readable medium (media), the present invention includes software for controlling both the hardware of the general purpose/specialized computer or microprocessor, and for enabling the computer or microprocessor to interact with a human user or other mechanism utilizing the results of the present invention. Such software may include, but is not limited to, device drivers, operating systems, execution environments/containers, and user applications.
CLAIMS

What is claimed is:

1. A method for automatically generating a user-configured portal, comprising:
   receiving portal configuration information from a user through a
   graphical user interface;
   preparing a build file, said build file prepared in accordance with
   said configuration information; and
   generating said portal in accordance with said build file.

2. The method of Claim 1, further comprising:
   deploying said portal.

3. The method of Claim 1, wherein the step of presenting a graphical user
   interface comprises presenting a wizard-style interface.

4. The method of Claim 1, wherein the step of generating said portal includes
   modifying a portal template in accordance with said build file.

5. The method of Claim 4, wherein modifying a portal template includes
   modifying a portal definition.

6. The method of Claim 5, wherein modifying code for a portal definition
   includes modifying a template portal file including XML based meta-data.

7. The method of Claim 1, wherein the step of preparing a build file includes
   modifying a template build file in accordance with said configuration information.

8. The method of Claim 1, wherein the step of receiving portal configuration
   information includes receiving at least one of a portal name, a web application with
which to associate said portal, a portal description, a portal template on which to
base said portal, and a resource files location for said portal.

9. The method of Claim 1, wherein the step of preparing a build file includes
preparing an Ant XML build file.

10. A system for generating a user-configured portal, comprising:
    a business platform;
    an enterprise application running on said business platform; and
    a control center in communication with said enterprise application, said
    control center capable of generating a portal for a web application of said enterprise
    application using configuration information received from a user through a
    graphical user interface.

11. The system of Claim 10, wherein said web application is part of said
    enterprise application.

12. The system of Claim 11, wherein said enterprise application is a J2EE
    application.

13. Computer software, capable of performing the steps:
    receiving portal configuration information from a user through a graphical
    user interface;
    preparing a build file, said build file prepared in accordance with said
    configuration information; and
    generating said portal in accordance with said build file.
14. Computer software, comprising:
   means for receiving portal configuration information from a user through
   a graphical user interface;
   means for preparing a build file, said build file prepared in accordance with
   said configuration information; and
   means for generating said portal in accordance with said build file.

15. A computer program product comprising:
   a computer usable medium having computer readable program code
   embodied therein for automatically generating a user-configured portal, the
   computer readable medium program code having:
   computer readable program code for receiving portal configuration
   information from a user through a graphical user interface;
   computer readable program code for preparing a build file, said build file
   prepared in accordance with said configuration information; and
   computer readable program code for generating said portal in accordance
   with said build file.

16. A computer program product for automatically generating a user-configured
    portal, comprising:
    computer code for receiving portal configuration information from a user
    through a graphical user interface;
    computer code for preparing a build file, said build file prepared in
    accordance with said configuration information; and
    computer code for generating said portal in accordance with said build file.

17. A computer-readable medium, comprising:
    means for receiving portal configuration information from a user through
    a graphical user interface;
means for preparing a build file, said build file prepared in accordance with said configuration information; and
means for generating said portal in accordance with said build file.

18. A method for receiving portal configuration information used to automatically generate a user-configured portal without requiring user coding of the portal, comprising:
presenting at least one screen to a user requesting selection of a name for said portal, a web application with which to associate said portal, a portal template on which to base said portal, and a resource files location for said portal;
receiving at least one of said name for said portal, said web application with which to associate said portal, said portal template on which to base said portal, and said resource files location for said portal from said user in response to said presenting said at least one screen; and
presenting a summary screen to said user, said summary screen including a list of files created during said generation of said portal.

19. A method for automatically generating a user-configured portal without requiring user coding of the portal, comprising:
presenting a graphical user interface for configuring a portal, said graphical user interface including at least one of:
a portal name screen, said portal name screen including at least one of a portal name field, a field for a web application to associate with said portal, and a portal description field;
a portal template screen, said portal template screen including at least one portal template on which to base said portal;
a resource files location screen, said resource files location screen including a field for a resource files location for said portal;
receiving portal configuration information from a user through said interface, said portal configuration information including at least one of a portal name, a web application to associate with said portal, a portal description, a portal template on which to base said portal, and a resource files location;

preparing a build file, said build file prepared in accordance with said configuration information;

generating said portal in accordance with said build file; and

presenting a summary screen, said summary screen including a list of files created during said generation of said portal.
Figure 1
Figure 3A
Figure 3B
Create new portlet

Use wizard? (N) → To Step 465

Use wizard? (Y) → Specify portlet name, description, and portal

Specify portlet components

Specify content → Basic/Two Pages

Specify location of portlet resource files

Perform steps for implementing web services

Figure 4A
From Step 440

Summary screen

Automatically generate portlet

Select next step

Review docs

From Step 215

Open portlet documentation

Launch portlet editor

Edit portlet

Launch portal editor

Edit portal

Figure 4B
510 Specify location of business platform server

515 Generate server-side proxies

520 Specify type of code generation

525 Select web service and operations

530 Edit web services list? Y → Add/Remove web services

N → 540 Generate preview of code

545 Display code

Figure 5
Figure 6
Figure 7A

Figure 7B
Portlet Wizard [MyPortlet]

1. Portlet Name
2. Portlet Components
   3. Content Types
   4. Resource Files Location
   5. Summary
   6. Next Steps

Portlet Components

Indicate which components to include in the portlet:

- Titlebar (required)
- Banner
- Header
- Content (required)
- Footer
- Help Page

Figure 7C

Portlet Wizard [MyPortlet]

1. Portlet Name
2. Portlet Components
3. Content Types
4. Resource Files Location
5. Summary
6. Next Steps

Content Types

Choose a content type for the portlet:

- Basic (no Workflow)
- Two Pages (with Workflow)
- Web Services

Figure 7D
Figure 7G

Figure 7H
Figure 71

Generated Code Types

Select the output option for the code generated to call the Web service.

Output Options:
- Form

Description:
Blah blah blah blah blah blah blah blah blah blah blah blah

Example:

```xml
<portletform event="Fm" FormAppFlowConstants.PORTLET_REFIRE
<table border="0" align="center">
  <tr>
    <td width="100%" align="center">
      Operation name: <B>BoboBakeFish BakeFish</B>
    </td>
  </tr>
  <tr>
    <td>
      if (!target.equals("target_submit_BakeFish")
      && !target.equals("target_do_BakeFish")
      && !target.equals("target_form_BakeFish")
      && sourceData.length > 0)
    </td>
  </tr>
</table>
```

Figure 7J

Web Services

Choose a Web service and an operation to include in the portlet.

<table>
<thead>
<tr>
<th>Web Services</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web Service 1</td>
<td>Operation 1</td>
</tr>
<tr>
<td>Web Service 2</td>
<td>Operation 2</td>
</tr>
<tr>
<td>Web Service 3</td>
<td>Operation 3</td>
</tr>
</tbody>
</table>

Operation Details:
- Parameters: parm1, parm2, parm3
- Description: Blah blah blah blah blah blah blah blah blah
Figure 7K

Figure 7L
Figure 8A
if ( target != null
   && target.equals("target_submit_hello")
   && req_name.trim().length() > 0 )
{
    try
    {
      <\%

      <tr><td align="right"><b><%=cnvrtSC("name")%></b></td><td><%=cnvrtSC(String.valueOf(req_name))%></td></tr>

      <td align="right"><b><i18n:getMessage
      messageName='result'
bundleName='=getBundleName((javax.servlet.jsp.JspPage)pageContext.getPage())%'></b><%=cnvrtSC(String.valueOf(p_http__workshop_7001__samples_HelloWorld_jws_WSDL_wdsl_HelloWorldSoap.hello((new
String(req_name)).toString())))%></td>
<% } catch ( Exception ignore )
{
%

      <i18n:getMessage messageName='functionality.unavailable'
bundleName='=getBundleName((javax.servlet.jsp.JspPage)pageContext.getPage())%'></b><%=cnvrtSC(ignore.toString())%><br>
<% }
%

%>

Figure 8B
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>name</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>submit</strong></td>
<td></td>
</tr>
</tbody>
</table>

Figure 8C
<%@ include file="HelloWorld_include.inc" %>
<%@ taglib uri="portlet.tld" prefix="portlet" %>
<%@ taglib uri="i18n.tld" prefix="i18n" %>
<%@ page import="com.bea.portal.appflow.PortalAppflowConstants"%>
<%@ include file="/framework/resourceURL.inc"%>

<% http__workshop_7001_samples_HelloWorld_jws_WSDL_wdsl.HelloWorld
p_http__workshop_7001_samples_HelloWorld_jws_WSDL_wdsl_HelloWorld =
new
http__workshop_7001_samples_HelloWorld_jws_WSDL_wdsl.HelloWorld();
http__workshop_7001_samples_HelloWorld_jws_WSDL_wdsl.HelloWorldSoap
p_http__workshop_7001_samples_HelloWorld_jws_WSDL_wdsl_HelloWorldSoap
 =
p_http__workshop_7001_samples_HelloWorld_jws_WSDL_wdsl_HelloWorld.get
HelloWorldSoap();
%>

<% java.lang.String name = __REPLACE_ME__;
%>
<%="cnvrtSC(String.valueOf(p_http__workshop_7001_samples_HelloWorld_jws_
WSDL_wdsl_HelloWorldSoap.hello(name)))"%>

Figure 9
%@ include file="HelloWorld_include.inc" %>
%@ taglib uri="portlet.tld" prefix="portlet" %>
%@ taglib uri="i18n.tld" prefix="i18n" %>
%@ page import="com.bea.portal.appflow.PortalAppflowConstants" %>
%@ include file="/framework/resourceURL.inc" %>

<% http___workshop_7001_samples_HelloWorld_jws_WSDL_wsd1_HelloWorld
    p_http___workshop_7001_samples_HelloWorld_jws_WSDL_wsd1_HelloWorld = new
    http___workshop_7001_samples_HelloWorld_jws_WSDL_wsd1_HelloWorld();

    http___workshop_7001_samples_HelloWorld_jws_WSDL_wsd1_HelloWorldSoap
    p_http___workshop_7001_samples_HelloWorld_jws_WSDL_wsd1_HelloWorldSoap =
    p_http___workshop_7001_samples_HelloWorld_jws_WSDL_wsd1_HelloWorld.getHelloWorldSoap();
%>

<%/
    public abstract java.lang.String
    http___workshop_7001_samples_HelloWorld_jws_WSDL_wsd1_HelloWorldSoap.
    hello(java.lang.String)
    throws java.rmi.RemoteException
    http___workshop_7001_samples_HelloWorld_jws_WSDL_wsd1_HelloWorldSoap.
    hello(org.openuri.www.Hello)
    throws java.rmi.RemoteException
*/%>

Figure 10
Open Project

Create new portal

Use wizard? N→ Edit portal definition using portal editor

Y→ Enter portal name and description

New web application? Y→ Enter web application name

N→ Select web application

Select template

Select resource files location

View list of files created

Select type of deployment

View deployment status (optional)

Figure 11
Figure 12A

Figure 12B
Figure 12C

Figure 12D
Figure 12E
Figure 12F
Figure 12G

Figure 12H
Launch portal wizard

Receive portal configuration information and display appropriate user screens

Prepare build file

Build portal

Figure 13
<portlet-pool>
  <portlet-name>WorldNews</portlet-name>
  <portlet-name>Bookmarks</portlet-name>
  <portlet-name>Dictionary</portlet-name>
  <portlet-name>Quote</portlet-name>
  <portlet-name>WebSearch</portlet-name>
</portlet-pool>

</page>

</pages>

<property-set-pool>
  <property-set-name>Avitek.usr</property-set-name>
  <property-set-name>Investor.usr</property-set-name>
</property-set-pool>

<skin-pool>
  <skin-name>blueinfinity</skin-name>
  <skin-name>brightlight</skin-name>
  <skin-name>coologic</skin-name>
  <skin-name>futurism</skin-name>
  <skin-name>lumiereorange</skin-name>
</skin-pool>

<layout-pool>
  <layout-name>twocolumn</layout-name>
  <layout-name>threecolumn</layout-name>
  <layout-name>fourcolumn</layout-name>
  <layout-name>spanning</layout-name>
  <layout-name>spanningthreecolumn</layout-name>
</layout-pool>

<portal-pool>
  <portlet-name>WorldNews</portlet-name>
  <portlet-name>Portfolio</portlet-name>
  <portlet-name>Bookmarks</portlet-name>
  <portlet-name>Dictionary</portlet-name>
  <portlet-name>Email</portlet-name>
  <portlet-name>Quote</portlet-name>
  <portlet-name>WebSearch</portlet-name>
  <portlet-name>PrimaryCampaign</portlet-name>
  <portlet-name>SecondaryCampaign</portlet-name>
  <portlet-name>QuickLinks</portlet-name>
  <portlet-name>CompanyProfiles</portlet-name>
  <portlet-name>GroupToDo</portlet-name>
  <portlet-name>MyToDo</portlet-name>
  <portlet-name>CustomerService</portlet-name>
  <portlet-name>Newsletters</portlet-name>
  <portlet-name>MyNewsletters</portlet-name>
  <portlet-name>ReviewNewsletters</portlet-name>
  <portlet-name>WhatsHot</portlet-name>
</portal-pool>

Figure 14B
Figure 15
INTERNATIONAL SEARCH REPORT

A. CLASSIFICATION OF SUBJECT MATTER

IPC(7) : G06F 15/16
US CL : 709/ 202, 250

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 709/ 202, 250

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

Please See Continuation Sheet

C. DOCUMENTS CONSIDERED TO BE RELEVANT

<table>
<thead>
<tr>
<th>Category</th>
<th>Citation of document, with indication, where appropriate, of the relevant passages</th>
<th>Relevant to claim No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y,P</td>
<td>US 6,476,828 B1 (BURKETT et al) 05 November 2002 (05.11.2002), column 1, line 21 - column 2, line 12.</td>
<td>1-19</td>
</tr>
<tr>
<td>Y</td>
<td>US 2,260,021 B1 (WONG et al) 10 July 2001 (10.07.2001), column 3, line 17 - column 5, line 43; column 12, line 65 - column 13, line 58.</td>
<td>1-19</td>
</tr>
</tbody>
</table>

Further documents are listed in the continuation of Box C. See patent family annex.

Date of the actual completion of the international search: 30 June 2003 (30.06.2003)

Date of mailing of the international search report: 11 AUG 2003

Name and mailing address of the ISA/US
Mail Stop PCT, Attn: ISA/US
Commissioner for Patents
P.O. Box 1450
Alexandria, Virginia 22313-1450
Facsimile No. (703)305-3230

Authorized officer
Paul H Kang
Telephone No. (703) 305-3900

Form PCT/ISA/210 (second sheet) (July 1998)
Continuation of B. FIELDS SEARCHED Item 3:
East
Search terms: user GUI, portal, java, J2EE, XML