Integration of AJAX Technologies into the MaterialsGrid Portal for Physical Properties Queries

Xiaoyu Yang[1], Thomas V. Mortimer-Jones[2], Dan J. Wilson[3], Martin T. Dove[1], Lisa Blanshard[2]

1. *Department of Earth Sciences, University of Cambridge, Downing Street, Cambridge CB2 3EQ*
2. *Science and Technology Facilities Council, Daresbury Laboratory , Warrington, Cheshire WA4 4AD*
3. *Department of Crystallography, J. W. Goethe University, 60438, Frankfurt, Germany*

Abstract

The query of material's properties via a grid enabled Web portal plays a critical role in the MaterialsGrid project within eScience. However, developing a normal standard portlet to query material properties is by no means a perfect solution. Portal page refresh is an expensive action as one portlet refresh can result in the refreshing of other portlets. In order to address this issue in material properties queries, the AJAX technique has been adopted. In this paper, we proposed a development model of using AJAX in a JSR-168 portlet, and discussed the integration of AJAX into the MaterialsGrid portal for chemical formula search. By using AJAX, users can handle many queries and only apply the updates to the display of table, without requiring a full page to refresh. This can improve query performance and bring a more user-friendly interface.

1 Introduction

MaterialsGrid [1] is UK government funded project that aims to create a pilot dynamic database of materials properties based on quantum mechanical simulations run within grid computing environments within eScience. As eScience promotes developing a grid-enabled Web portal to interface with complex underlying grid tools and services through a standard Web browser, a grid portal will be developed as one of the core components in MaterialsGrid. Queries of the physical properties of inorganic materials via grid portal and the display of the returned information in a user-friendly way play a critical role in this project.

Portals can be classified into portlet-based and non-portlet based [2], and it has been decided to develop a portlet-based portal for MaterialsGrid. In order to query properties on a material of interest, a chemical formula search facility has been formulated, which can then be used to develop a query portlet that can be plugged into the MaterialsGrid portal. The novelty of this search facility is that it allows users to enter a partial chemical formula rather than the formula in a rigid format. For example, it can facilitate the query by entering a formula containing wildcards (e.g. Ti*), splitting formula into components and the number of atoms (e.g. AlOOH $\rightarrow$ Al x 1, O x 2, H x 1), or entering a formula which contains bracket notation (e.g. Ca (OH)2 $\rightarrow$ Ca x 1, O x 2, H x 2).

However, as a single portal page may contain more than one portlet, which supply content to the portal page, portal page refresh is an expensive action as one portlet refresh can result in other portlets refreshing at the same time. This means developing a normal standard material properties query portlet cannot be a perfect solution as properties query each time brings overhead to the MaterialsGrid portal.

In order to tackle this problem, the AJAX (Asynchronous JavaScript and XML) technique has been used in developing the query portlet. Portlets and Ajax can be a perfect fit, as they are both focused on using a Web browser as the vehicle for presenting a user interface to the user [3]. AJAX is a Web development technique for creating interactive Web applications, and is one of the major enabling techniques for Web 2.0 [4]. AJAX can improve the user experience and make Web pages feel more responsive by exchanging small amounts of data with the server behind the scenes, so that the entire Web page does not have to be reloaded each time the user makes a request. This is meant to increase the Web page's interactivity, speed, and usability.

There are some Java/J2EE based AJAX toolkits/libraries available to facilitate the AJAX application development. However, documentation and tutorials provided by these toolkits/libraries mostly target J2EE Servlets [5]. Due to differences between Servlets and portlets, many of these AJAX toolkits may not be explicitly appropriate for portlets, or applying them to portlets may require additional configuration or learning. There is a concern that similar codes which are based on these toolkits or libraries work fine with servlets but may not work properly with portlets. Moreover, these toolkits usually hide the detail of AJAX working mechanism, which can make it a bit hard for new AJAX developers to start. In order to facilitate the integration of AJAX into the portal without relying on any AJAX toolkits / libraries to avoid any unpredicted hassles, a development model for the use of AJAX in JSR-168 [6, 7] portlets has been proposed with reference to related work. The development model proposed is then used in formulating an AJAX-based material properties query portlet which can be plugged into the MaterialsGrid portal.

The paper structure is as follows: Section 2 introduces the chemical formula search facility. Section 3 proposes a development model for the use of AJAX in JSR-168 portlets. Section 4 details the integration of AJAX into the MaterialsGrid portal for properties query using the development model proposed. In Section 5, the MaterialsGrid portal prototype is discussed.

2 Chemical Formula Search

As one of the requirements, chemical formula search should allow users to enter chemical formula with a fair degree of flexibility rather than just to enter a chemical formula in a rigid format. In many cases, the user may not know the exact chemical composition of the materials that they are interested in. Furthermore, scientists have their practice in writing some chemical formulas which do not follow this rigid format. For example, people like to write as “Ca(OH)2” rather than as “CaO2H2”, write as “FeOOH” rather than as “FeO2H”, and write as “CaAl2Si2O7(OH)2•(H20)”[1] (a mineral Lawsonite) rather that as “CaAl2Si2O10H4”.

However, this flexibility in chemical formula search is not very common. For example, search the ICSD (Inorganic Crystal Structure Database), which is the world’s most extensive database on inorganic crystal structures [8], is not very flexible. In order to accommodate these needs in MaterialsGrid, a chemical formula search facility has been developed. Major flexibilities of the search facility formulated are introduced as follows.

[1] Mineralogists do differentiate between element O on its own, O as part of the OH, and O as part of H2O.

The user can enter either a whole or partial chemical formula, which is used to query the database in order to return information on materials that may be of interest. The elements in the search string are totalled and used to search the database for matching structures. For instance a user interested in the properties of Quartz would enter "SiO2" into the search box and would receive back data about materials that contain one silicon atom and two oxygen atoms, plus any other elements, in its asymmetric unit cell. Different ways of writing the chemical formula are allowed - for example, the different ways of writing the formula of hydrated iron oxide, “FeOOH” and “FeO2H”, result in the same search being performed. Bracket notation can be used to group together repeating groups of chemical structure. For example searching for calcium hydroxide, "Ca(OH)2", will return materials containing one calcium, two oxygen and two hydrogen atoms. If the user does not know the exact chemical composition of the materials that they are interested in, they can use '*' as a wildcard to specify that they are interested in materials with any number of that element. A user interested in titanium containing materials would enter “Ti*” into the search box and would receive back data about all of the materials contained in the database that contain at least one titanium atom.

The analysis and design of this search facility is briefly described as follows. When the calculated properties of a material are entered into the database the contents of the asymmetric unit cell needs to be stored. Having a column for every element in the periodic table to store, how many occurrences of that element are contained in the materials asymmetric unit cell was discounted. It was decided to have another table to hold this information on the frequency of elements within the material. This table contains a row for each element in a material, which holds the element symbol, the frequency and a reference to the material in which it belongs.

When the user enters a search, a query is constructed that creates multiple joins between the main material information table and the table containing the element frequencies - one for each of the specified elements. This query will return only those materials that contain the specified elements. If the user has specified the number of elements, rather than using a wildcard, a clause is added to the query, which restricts the results to records that have the right frequency of elements. Once this query has been executed it is returned to the portal to be displayed.

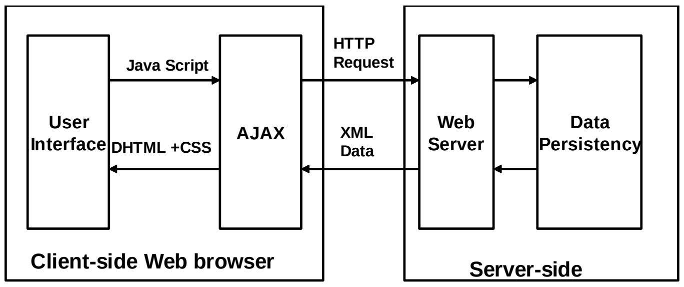


Figure 1. AJAX interaction mechanism

3 Development Model for the Use of AJAX in a Portlet

In order to facilitate the development of an AJAX-based material properties query portlet based on the chemical formula search facility, a development model for the use of AJAX in JSR-168 portlets has been proposed.

3.1 AJAX Core Mechanism

AJAX is a combination of techniques such as JavaScript, DOM, XML, and HTML/DHTML, etc., which allows a Web browser to update parts of a Web page asynchronously by communicating with a Web server using JavaScript through an *XMLHttpRequest* component.

Currently, most of Web applications adopt Browser/Server (B/S) architecture. In B/S mode, only a Web-browser is needed at the client side to view the process result from server, which is known as the so-called thin client mode. The Web server sends an HTML page which contains the data and HTML structure/frame to the client-side Web browser over the network (i.e. the internet). However, the major hidden problem with this approach is that each time the HTML structure /frame is also transferred to the client side, which overloads the networks. These HTML structure / frame can actually be generated locally within the Web browser. Moreover, each time a new HTTP request must be made to reload the whole page to view the different datasets.

By using AJAX, only the data or the content is transferred over the networks where the data/content is marshalled in XML format. The HTML structure and frame are created locally within the Web browser using JavaScript. As each time only the data is delivered rather than the whole Web page, users do not have to reload the whole page to get a different dataset. An AJAX mechanism is illustrated in Figure 1.

AJAX is made up of the following major technologies/components [9]:

1) *JavaScript* - JavaScript is the essential ingredient in AJAX allowing the building of client-side functionality. JavaScript is used to write functions that are embedded or included in HTML pages and interact with the Document Object Model (DOM) of the page to perform tasks.
2) *DOM* (Document Object Model) – DOM is a platform and language independent standard object model for representing HTML or XML and related formats. DOM is used heavily in JavaScript functions to manipulate parts of the HTML page.
3) *XMLHttpRequest* - The *XMLHttpRequest* object enables JavaScript to access the server asynchronously, so that the user can continue working whilst functionality is performed in the background. Accessing the server simply means making a simple HTTP request for a file or script located on the server. HTTP requests are easy to make and usually do not cause any firewall-related problems.
4) *Server-side component* - A server-side technology is required to handle the requests that come from the JavaScript client. This may involve retrieving data from database, handling the data retrieved and marshalling them in XML or text as a response to JavaScript. Apart from marshalling data using XML/text, another data marshalling mechanism is to use JavaScript Object Notation (JSON)[10].

3.2 Servlets and Portlets

Knowing the similarities and differences between portlets and servlets can help in understanding the integration of AJAX into portal.

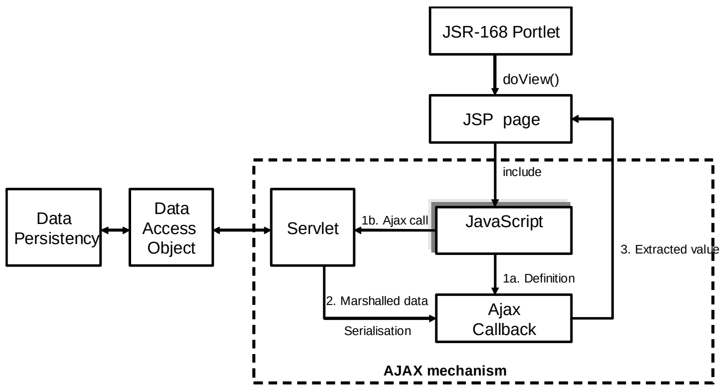


Figure 2. Development model of using AJAX in portal application

Servlets and portlets are Java components that run at server side to handle client requests to generate dynamic content [11]. They both use request and response objects, and their lifecycles are managed by containers. The packaging and deployment of both servlets and portlets are essentially the same. The portal needs to be run inside a servlet container such as IBM WebSphere Application Server (WPS) or Apache Tomcat [11].

Servlets and portlets also have got notable differences. With servlets, there is one request per response, i.e. a browser asks for a URL and the servlet container processes the HTTP request. The servlet normally calls its doGet() or doPost() method to generate a response, which is sent back to the client's Web browser. With portals, there can be more than one portlet supplying content for each Web page. Even though the end user submits a form for only one portlet at a time, the other portlets may still need to refresh their content to reflect any changes [11]. Servlets can provide complete Web pages, whereas portlets only provide fragments, which are aggregated by the portal to form a complete Web page. Portlets are not allowed to generate HTML code that contains tags such as base, body, frame, frameset, head, html, or title. The user cannot access a portlet directly using a URL in the way that a servlet is accessed. Instead, the URL points to the page containing the aggregated content on one page. All these differences have impact on the methods of building AJAX-based J2EE applications.

3.3 A Development Model

There are a number of open-source AJAX toolkits available such as DWR (Direct Web Remoting) [12], Dojo [13], Google Web Toolkit [14], Microsoft Atlas [15], Open Rico [16], Yahoo AJAX Library [17], and Zimbra's Kabuki AJAX Toolkit [18]. These tools offer a number of extremely useful user interface widgets and background tools for simplifying the process of building an AJAX application. However, in terms of a J2EE environment, most documentation and tutorials of these toolkits focus on building AJAX applications using servlets rather than portlets. Therefore, using these toolkits in portlets may cause some unexpected hassles as the differences between Servlets and portlets may result in similar codes which are based on these toolkits or libraries working fine with Servlets but not working properly with portlets. In order to facilitate the use of AJAX in a portlet application without depending on any AJAX toolkits / libraries to avoid any unpredicted hassles, we proposed a development model for the use of AJAX in JSR-168 portlets with reference to some related work [19, 20, and 3]. This model gives a clear view of how AJAX works, which is not always explicitly clear in AJAX toolkits, and makes it easy for new AJAX developers to start. The model indicates that in order to incorporate AJAX into the portlet the action request which should be handled by the *processAction()* method in JSR-168 portlet is now handled by a servlet. The development model is illustrated as follows:

1) Create a JSP page which is rendered through the *doView()* method of *a* JSR-168 portlet.
2) Write AJAX code in JavaScript, and include this JavaScript in the JSP page. The procedures of writing AJAX code within JavaScript is summarised as follows:
    a) Initialise the *XMLHttpRequest*
    b) Define a call-back function, which processes the server-side response
    c) Make AJAX call. This actually makes an HTTP request to the server by calling the *open()* and *send()* methods of the *XMLHttpRequest* object.

3) Develop a server-side component (i.e. a Servlet) to handle the request from the AJAX call. Usually this involves the operation of retrieving data from a back-end database. The retrieved data is marshalled in XML and sent back to the defined AJAX call-back function.
4) Parse the XML within the call-back function using DOM and DHTML technologies.

4 Integration of AJAX Technologies into the Material Properties Query

The integration of AJAX into the MaterialsGrid portal adopts the development model proposed. Some of the challenges presented are discussed.

4.1 Server-side Data Process

Many of the materials properties (e.g. atomic population, stress) involve complex data types (e.g. vector, matrix) and are stored in the form of XML. Oracle[2] has a dedicated XML data type called `XMLTYPE`, which is a system-defined opaque type for handling XML data [21]. It is made up of a CLOB to store the original XML data and a number of member functions to make the data available to SQL. One of the main challenges on the server side is how to query the XML data stored in Oracle and marshal them with other data types into XML which will be sent back to the AJAX call-back function.

In order to tackle these problems, we have developed a software component which can retrieve the XML-type data from Oracle working with a Data Access Object (see Section 4.2) and marshal the data obtained into XML by employing *Oracle XML DB* [22] and *JDOM [23]*.

4.2 Data Access Object and Data Persistency

One of the major requirements of MaterialsGrid is that it should provide a barter system to allow customers to offer data or computing resources in exchange for system access credit. This means that the MaterialsGrid may connect to different data persistence offered by customers such as relational database, XML dictionary, and flat files. In order to decouple object persistence and data access logic from any particular persistence mechanism, a J2EE pattern Data Access Object (DAO) [24] has been employed in the integration of AJAX into the MaterialsGrid. It is a layer between the servlet and specific data persistency (see Figure 2). By using the DAO layer, the change of data persistent storage has minimal rippling impact on the whole system. The Servlet only interacts with the DAO to retrieve the data of material properties regardless of the specific data source.

4.3 Client-side JavaScript Functions

The AJAX call-back function is the main client-side JavaScript function which is usually responsible for parsing the XML sent back from the servlet, extracting the data and present the data extracted in an HTML table. The DOM technology is heavily used within JavaScript to manipulate the HTML and display the data in the table. The key operations in the call-back function are discussed as follows:

1)Display “ Retrieving data …” message

After the HTTP request is sent out, AJAX keeps checking the state of the request changes. The *XMLHttpRequest* defines the ready state as one of five levels: 0-uninitialised, 1-data loading, 2-data loaded, 3-interactive, and 4-completed. The data loading may take some time, and it is always good practice to display such a message as “Retrieving / loading data …” to the end user while the browser is waiting for the response [19]. Without the message, the end user may think that nothing is happening.

2)Read XML to the table

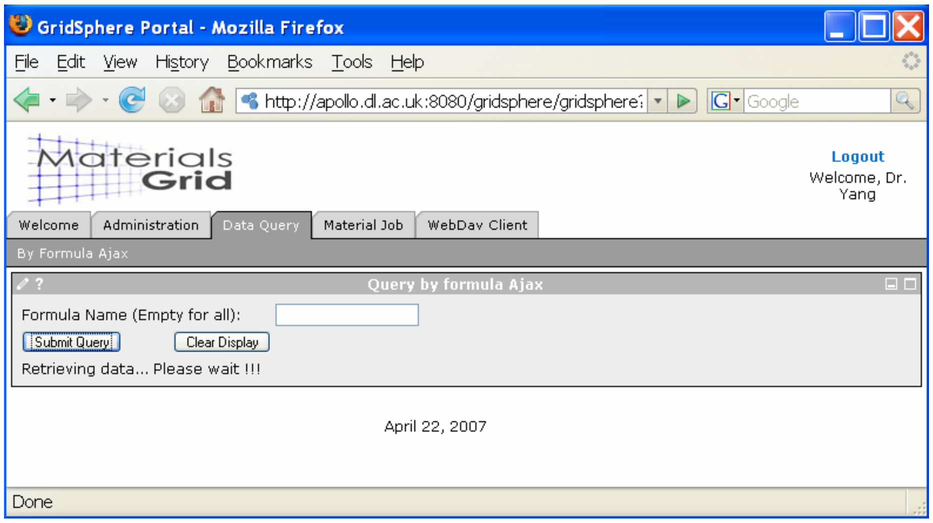
This operation is responsible for the parsing of the XML, extracting data and presenting data in an HTML table. DOM technology and DHTML are used to dynamically update the HTML selected contents.

3)Clear display data

This operation is not often used in B/S mode where the whole page is refreshed. In AJAX, the whole Web page is not refreshed and only the particular part is updated. This “clear data display” sub-function is invoked to clear the data display area before the next data is displayed.

In addition to the call-back function, another major client-side JavaScript function is required which handles wrapping the form data in XML as part of the HTTP request and sending it to the server.

[2]Oracle is adopted as the back-end database in MaterialsGrid.



Formula Name (Empty for all):

Retrieving data... Please wait !!!

April 22, 2007

Figure 3. MaterialsGrid portal prototype screenshot 1 - material properties query in AJAX

5 MaterialsGrid Portal Prototype

The MaterialsGrid portal prototype has been developed, and major functionalities include: (i) the query of materials properties, (ii) the visualisation of the cell structure, and (iii) the submission of simulation job(s) to a GridSam resource manager [25][3].

A JSR-168 AJAX-based material properties query portlet has been primarily developed utilising the development model proposed. Two JSR-168 compliant portlet containers (i.e. GridSphere and IBM WebSphere Portal) have been used to test this portlet [2]. It has been demonstrated that this portlet can successfully work in both GridSphere [26] and IBM WebSphere Portal [27] frameworks. The properties query portlet supports chemical formula search by entering a partial chemical formula, and makes the query in an asynchronous mode. A screenshot showing the retrieval of properties for all available materials within the database is illustrated in Figure 3.

The Jmol [28] Java Applet has been employed in the MaterialsGrid portal for unit cell visualisation. The structure visualisation involves several operations: (i) the user provides cell parameters and atomic coordinates, (ii) the portal exports them in CML format [29], and (iii) the CML document is loaded into the Jmol. A screenshot of visualising the structure of unit cell of diaspore is shown in Figure 4.

[3] A workflow will soon be developed in MaterialsGrid.

6 Discussions

The integration of AJAX into the MaterialsGrid portal for properties query adopts the development model proposed. The main advantages of using this model include: (i) the development does not rely on any third-party AJAX toolkits or libraries. There are no third party toolkits /libraries downloading/installation/configuration/learning involved, which could be a time-consuming job, especially in the case that many of these toolkits target servlets rather than portlets, or their usage specification is poorly documented. By referencing the development model, new AJAX developers can start using AJAX in portlet straightaway. (ii) the model presents a detailed view of an AJAX working mechanism, and (iii) the model gives a straightforward workflow showing how to integrate AJAX into a portal.

However, using this model for AJAX-based portlet development requires a lot of client-side JavaScript coding, especially using DOM to interact with DHTML, which can be quite tedious and may cause JavaScript cross-browser problems. For example, the material properties query portlet currently does not support IE6 (Internet Explorer) (it supports IE7, Firefox, and Safari). Hence for experienced AJAX developers, the integration of AJAX into the portal can partially or wholly use some third-party AJAX toolkits or libraries. For example, it has been shown that DWR (Direct Web Remoting) can be used in AJAX-based portlet development although it involves extra configuration and usage learning.

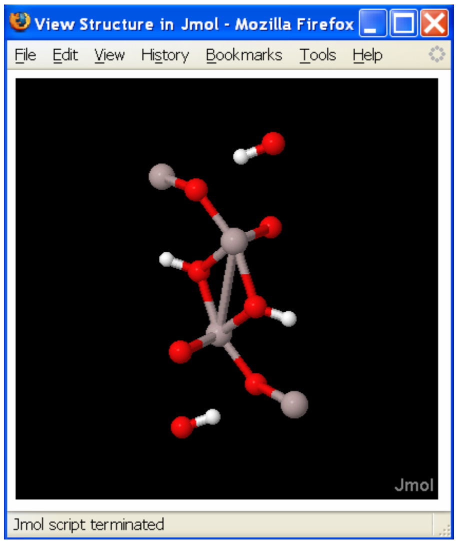

View Structure in Jmol - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Jmol
Jmol script terminated

Figure 4. MaterialsGrid portal prototype screenshot 2 - structure visualisation

7 Conclusions

A chemical formula search facility has been formulated which can support the properties query by entering partial chemical formula. In order to tackle the problem that one portlet refresh can in the meantime result in other portlets to refresh, the AJAX technique has been employed and a development model of using AJAX in JSR-168 portlet has been proposed with reference to the related work. The development model proposed has been applied in developing a JSR-168 portlet for the query of materials properties. It has been demonstrated that, by using AJAX, users can handle many queries and only apply the updates to the display of a table, without requiring a full page refresh. This improves query performance and brings a more user-friendly interface.

Acknowledgement

The authors would like to acknowledge the UK government DTI funded MaterialsGrid project.

References

[1] MaterialsGrid http://www.materialsgrid.org
[2] X. Yang, M., Dove, M., M., Hayes, M., Calleja, L. He, P. Murray-Rust "Survey of Tools and technologies for Grid-enabled Portals". *UK e-Science All Hands on conference, 2006*, UK.
[3] S. Salkosuo "DWR makes interportlet messaging with Ajax easy", *IBM developerWorks*, 14, July, 2006. http://www-128.ibm.com/developerworks/web/library/j-ajaxportlet/index.html
[4] T. O'Reilly "*What is Web2.0*" http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html?page=1
[5] Java Servlets http://java.sun.com/products/servlet/
[6] JSR168 and JSR 286 http://developers.sun.com/portalserver/reference/techart/jsr168/
[7] Alejandro, H. Stefan. (2003, Oct 7). *Java Portlet Specification.* Available: http://jcp.org/aboutJava/communityprocess/final/jsr168/index.html
[8] ICSD http://www.fiz-karlsruhe.de/ecid/Internet/en/DB/icsd/
[9] C. Darie et al, "*AJAX and PHP: Building Responsive Web Applications*", Packt Publishing, Feb. 2006.
[10] P. McCarthy "AJAX for Java Developers: Java Object Serialisation for AJAX", *IBM developerWorks*, 04,October, 2005.
[11] J.Linwood and D.Minter "*Building Portals with the Java Portlet API*", Apress,2004
[12] DWR http://getahead.org/dwr
[13] Dojo http://dojotoolkit.org/

[14] Google Web Toolkit http://code.google.com/webtoolkit/
[15] MircoSoft Atlas http://ajax.asp.net/Default.aspx
[16] Open Rico http://www.openrico.org/
[17] Yahoo Ajax Library http://developer.yahoo.com/yui/
[18] Zimbra's Kabuki AJAX Toolkit http://www.zimbra.com/community/kabuki_ajax_toolkit_download.html
[19] K.Bishop, D.Philips "Building an Ajax portlet for WebSphere Portal", I*BM developerWorks*, 16 August, 2006.
[20] K.Bishop, D.Philips "Using Ajax with WebSphere Portal", I*BM developerWorks*, 28 June, 2006.
[21] Oracle-Base http://www.oracle-base.com/articles/9i/XMLTypeDatatype.php
[22] Oracle XML DB http://www.oracle.com/technology/tech/xml/xmldb/index.html
[23] JDOM http://www.jdom.org/
[24] Data Access Object http://java.sun.com/blueprints/corej2eepatterns/Patterns/DataAccessObject.html
[25] L. He, M. Dove, M. Hayes, M. Calleja, X. Yang, P. Murray-Rust "Developing Lightweight Application Execution Mechanism in Grids" *UK e-Science All Hands on conference, 2006*, UK
[26] GridSphere http://www.gridsphere.org
[27] IBM WebSphere Portal: http://www-306.ibm.com/software/genservers/portal/
[28] Jmol http://jmol.sourceforge.net/
[29] P. Murray-Rust, C. Leach and H. S. Rzepa, Henry S. *Chemical Markup language*, Book of Abstracts, 210th ACS National Meeting, Chicago, IL, August 20-24 (1995), (Pt. 1), COMP-040.