Protein PocketViewer: A Web-Service Based Interface for Protein Pocket Extraction and Visualization

Xiaoyu Zhang
Department of Computer Science& Information Systems
California State University San Marcos
San Marcos, U.S.A.

Martin Gordon
Department of Computer Science& Information Systems
California State University San Marcos
San Marcos, U.S.A.

***Abstract*—One important problem in bioinformatics is to study pockets or tunnels within the protein structure. These pocket or tunnel regions are significant because they indicate areas of ligand binding or enzymatic reactions, and tunnels are often solvent ion conductance areas. The Protein Pocket Viewer (PPV) is a web interface that allows the user to extract and visualize the protein pockets in a browser, based on the algorithm in** [1]**. The PPV packaged the pocket extraction executable as a web service, and made it accessible to all users with the Internet access and a modern java enabled browser. The PPV employed the Model2design pattern, which led to a loosely coupled implementation that is more robust and easier to maintain. It consists of a client web interface for user inputs and visualization, a middle-layer for controlling the flow, and the backend web services performing the actual CPU-intensive computation. The PPV web client consists of multiple window regions, with each region providing differing views of the protein, pockets and related information. For a more responsive user experience, the PPV web client employs AJAX for asynchronous execution of long running tasks, like protein pocket extraction.**

Keywords—protein structure; pockets; Model 2; AJAX; web service; visualization;

I. INTRODUCTION

Bioinformatics applies computational tools to study problems in molecular biology. Structures are critical for thefunctions of proteins. One important bioinformatics problem is to determine pockets or tunnels within the protein structure. These pocket or tunnel regions are significant because they indicate areas of ligand binding or enzymatic reactions [2], and tunnels are often solvent ion conductance areas [3]. Such computation can be data dense and computationally intensive due to the volume of data processing involved for a single protein and the number of proteins in the database. So the computations are more efficiently performed on powerful computational servers. Visualization tools are also important for bioinformatics research. A visualization tool would help the user to better comprehend and quickly consume data of the computed protein pockets. The visualization should be available for the user on the less powerful client computers, most conveniently in a web browser without installing any special software.

In this paper, we developed a web service [4] based visualization interface to the pocket extraction algorithm described in [1].The algorithm employs a two-step level set marching algorithm (Fig 1). The first step of the level set marching algorithm marches outward from the protein surface to some distance equal to a given threshold. At completion of the outward marching step, an outer surface is obtained with all indentations on original surface filled. The second step of the algorithm marches backward from the outer surface back toward the protein for the same distance. The second marching step cannot infiltratethe protein surface, or reach depressions and tunnels on the surface. The unreachable regions outside the protein surface are considered as pockets. The bounding envelops of the pockets are then extracted using standard level-set methods.

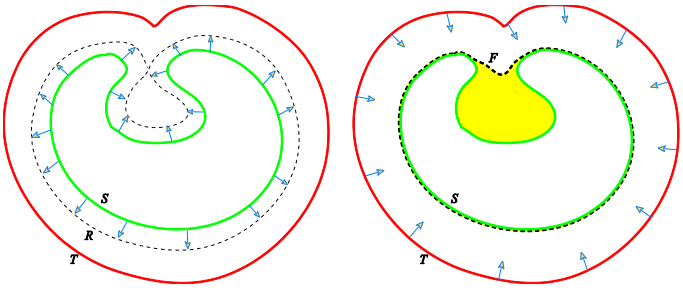

Fig. 1. The two-step level set marching algorithm for pocket extraction. (a) Outward marching from the original surface S to an outer surface T; (b) Backward marching from T to uncover the pocket as the shaded region

The visualization interface, Protein Pocket Viewer (PPV) at *http://ppv.cs.csusm.edu:8080/PPVClient/PPV.jsp,*allows users to display and manipulate data related to protein pockets in ajava-enabled browser.The web interface consists of multiple window regions, with each region providing differing views of protein pocket related data (Fig 2). The display includes both metadata about the protein and associated pockets and the three dimensional rendering of the protein and pockets. The 3D visualization of the pockets is displayed in the central rendering region. The 3D rendering can be maneuveredby the user to view of all surfaces of the protein and pockets. The 3D display can be controlled such that individual pocket can be shown or hidden, and rendered in different styles, such as a filled, dot or mesh surface. The protein sequence informationis displayed as text in the sequence region. The pocket information region displays pocket metadata such as a pocket ID, pocket surface area, and pocket volume for each pocket identified using the two-step level setmarching algorithm. The

web interface also includes a protein information region that provides additional information on the protein containing the selected pockets.

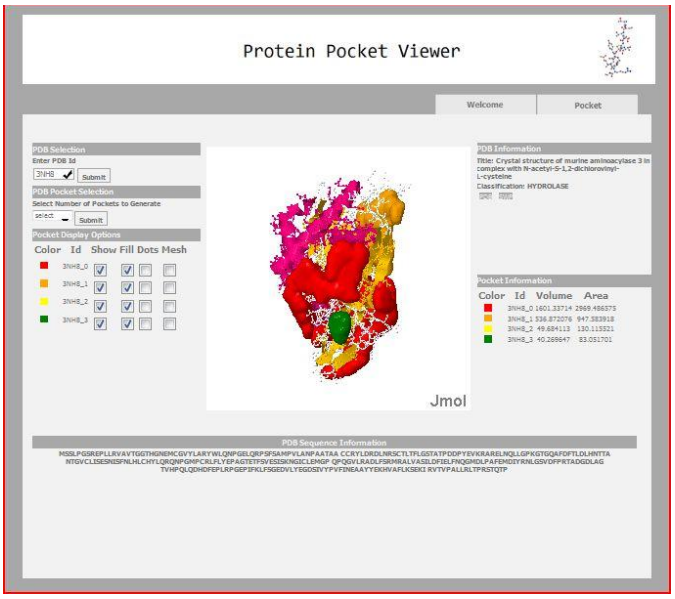

Fig. 2. Protein Pocket Viewer interface.

The PPV allows the user to visualize the protein pockets and their relationship to the protein, and is accessible to all users with the Internet access and a modern java enabled browser. The PPV implemented the web client, a web service wrapper for invoking pocket generation, Jmol[5]viewer integration, and cache management for protein and pocket files. It has some unique features, such as

- On-the-fly generation and visualization of pockets given a valid Protein Data Bank (PDB)[6]ID or a well-formed PDB file using the two-step level setmarching algorithm.
- Caching management capability where expensive pocket generation results are cached for the later requests.
- Asynchronous transactions via AJAX [7]relieving the user from waiting for page refresh during long running tasks, such as pocket generation and PDB metadata retrieval.

In the rest of the paper, section 2 discusses background concepts used by the PPV and related work. Section 3 presents the design of the PPV and implementation details. We then conclude and discuss some future directions in section 4.

II. Background And Related Work

A. Background

The Protein Pocket Viewer employs the implementation for protein pocket extraction algorithm described in [1], which was implemented as a C++ program running on a computational server.

The Protein Data Bank (PDB) is the single worldwide repository of information about the 3D structures of large biological molecules[6], managed by the Research Collaboratory for Structural Bioinformatics (RCSB). Every protein in the PDB has a unique ID and its structure is available in PDB file format. The pocket extraction program requires a valid PDB ID or a well-formed PDB file as input.

A web service [4]is used to initiate the pocket extraction implementation. This web service provides the abstraction of the pocket extraction program and enables a bridge between the client Java classes to the pocket extraction service implemented in C++. Protein metadata, PDB file, and pocket files transfer between the server and the client via the web service interface.A web service is an implementation of the SOA design approach employing XML [8], XSD [8], and SOAP [9] standards based technologies. The Service Oriented Architecture (SOA) is a design consideration where its solution is distributed, loosely coupled, standards based, reusable, and stateless.The Web Service Description Language (WSDL)[10] provides a way for a web service provider to describe the public interface of the web service.

The 3D rendering of the proteins and their respective pockets in a browser was performed with Jmol[5], which is an open source Java viewer for 3D chemical structures. Jmol includes features for element display selection, scheme selection, element color assignment, surface display selection, and measurements. Element display selection allows the user to select the elements within the chemical structure to display. Scheme selection allows the user to select schemes such as CPK space-fill, ball and stick, sticks, wireframe, and cartoon for the chemical structure. Surface display selection allows the user to select how the surface is displayed.

The PPV utilizes Java server pages [11], cascading stylesheets [12], and JavaScript XHTML DOM scripting [13] for designing and displaying the website content. JavaScript [14], AJAX [7], and Java servlets [11] provide the flow and navigation capabilities within the viewer.

The Protein Pocket Viewer employs the Model 2 design pattern[15], which is a variant of the Model-View-Controller (MVC) design pattern. Model2 extends the Model-View-Controller design pattern for use with web applications. Model2 can employ Java server pages and cascading stylesheets to implement the view of the pages within the website. Java server pages (JSP) enable dynamic content within HTML pages. Cascading stylesheets are used to describe the look and feel of the content to display. The Java server pages and servlets provide the controller responsibility of the implementation. Behind the scenes, some Java classes provide the functionality that enable retrieval of the protein, the pockets, and the associated metadata.

There are many advantages to using the Model2 design patterns[15]. The main advantage of the design pattern is to separate the different levels of concerns within the implementation. Keeping the levels of concerns separate helps facilitate cleaner, and more loosely coupled implementation, which lowers the dependencies between different components. It allows the developer to break the implementation into smaller more manageable components. Each component can be tested separately to ensure its correctness. This makes the implementation more maintainable.

Asynchronous JavaScript and XML (AJAX) [7]provides the ability to initiate functionality asynchronously. AJAX frees up the user from busy waiting when the request is being processed at the server end. In an interactive program like a web interface, the user wants to continue interacting with the interface while waiting for a long-time processing or data retrieval. For example, a graphical user interface that performs a long running calculation while allowing the user to navigate through related charts, data, and documentation will give the user the option of continuing with other tasks instead of having to just sit and wait.

B. Related Work

Computed Atlas of Surface Topography of proteins (CASTp)[16] is an online tool that locates and measures pockets on 3D protein structures. The CASTpuses a program named CAST [17]to locate and quantify pockets. The CAST employs computational geometry of complex shapes, based on alpha shape and discrete flow theory, for pocket calculation. More recently, the CASTp[18] provides annotations derived from the Protein Data Bank (PDB), Swiss-Prot, and Online Mendelian Inheritance in Man (OMIM).

The PPV and the CASTp are similar in that both display proteins and their respective pockets in Jmol, and both display protein and pocket metadata information. However, there are significant differences between the PPV and the CASTp. First, the PPV uses a different pocket extraction algorithm from the theCASTp. In many cases, the CASTp requires the user to upload PDB structured files in order to generate and visualize pockets while PPV provides on-the-fly generation and visualization of pockets given a valid PDB Id or a well-formed PDB file. The PPV also employs caching management to avoid expensive computation for pocket extraction when possible.The PPV uses AJAX for long running tasks, such as pocket generation and PDB metadata retrieval, relieving the user from needlessly waiting for page refresh.

III. DESIGN AND IMPLEMENTATION

The PPV project employs the Model 2 design pattern to ensure the responsibility of a particular component does not overreach into other components. We first describethe high level architecture and data flows of the PPV.Subsequent subsection takes a closer inspection of the view, controller, and model components.

A. PPV Architecture

The PPV consists of a client web interface for user inputs and visualization, a middle-layer for controlling the flow, and the backend web services performing the actual computationsuch as retrieving PDB files and pocket extraction,as shown in **Error! Reference source not found.** The web interface is implemented primarily with Java Server Pages and a cascading style sheet; the middle-layer control is implemented using JavaScript and Java servlets running on the Tomcat web server[19]; the backend web services and business logic are implemented with Java and C++ classes.

In the case of the PPV,the business logic is the preparation, extraction, and retrieval of protein PDB files and their associated pockets into PMESH [20] file format, and metadata describing those files. The PPV architecture uses a web serviceto enable the Java web client to initiate protein pocket generation implemented in C++. The reasons for choosing this architecture for the protein pocket viewer are threefold. First, using Java technologies for the web client allows the developer to capitalize on the capabilities that are robust and freely available. Second, existing program to extract the protein pockets was implemented in C++. A web service allows us to bridge the capability between the Java client and the C++ server in a clean, standards based manner. Third, packaging the pocket extraction as a web service also abstracts it as a single purpose, loosely coupled component that can be tested separately and used by others. The WSDL provides all the information necessary to create a web service client to consume the pocket extraction web service, without knowing the implementation details of the pocket extraction algorithm.

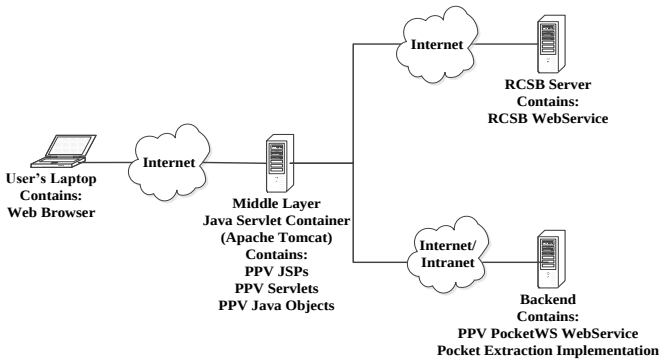

Fig. 3. Protein Pocket Viewer Web Service Architecture Diagram

The user's web browser connects to the PPV web client in the middle-layer. The PPV web client resides within an Apache Tomcat web server, containing the code for PPV JSPs, Servlets and Java objects. The PPV web client consumes the web services for protein pocket extraction, named "PocketWS", implemented in C++ and hosted on the backend computational server together with the existing pocket extraction program. The "PocketWS" web service generates the protein pocketsin the PMESH file format for displaying in the user's browser. Besides the "PocketWS" web service,the PPV web client can also consume other web services, e.g. the RCSB PdbWS(www.rcsb.org/pdb/services/pdbws) hosted on the remote RCSB server. The PdbWS web service is used by the PPV to validate PDB ID entry supplied by the user.

B. PPV DataFlow

Fig 4 shows the steps and dataflow how the PPV performs the function of extracting protein pockets and displaying them for the user.

Step 1: The user enters the PDB ID, selects the number of pockets to generate and clicks the submit button on the PPV interface, which calls the CheckPDBId servlet.

Step 2: The CheckPDBId servlet checks for the validity of the PDB ID by calling the RCSB pdbWS web service.

Step 3:TheCheckPDBId returns the validation result to the PPV interface, which displays a visual indicator if the given ID is invalid.

Step 4a:If the ID is valid, the GetPDB servlet iscalled asynchronously via AJAX.

Step 4b: The GetPocket servlet is called asynchronously via AJAX with the validated PDB Id and the number of pockets to be generated. Note that step 4a and 4b run currently in two separate threads.

Step 5: The GetPDB servlet retrieves the PDB XML file and its metadata from the RCSB PDB database, and returns it to be displayed in the PPV interface.

Step 6:The GetPocket servlet calls the "PocketWS"web service, which extracts the pocketsas PMESH files for the given protein. The PocketWS returns an array of PocketData, each of which contains pocket metadata such as volume and surface area, and the location of corresponding PMESH file. The generated pocket PMESH files are then transferred to the client and displayed in the Jmolviewer.

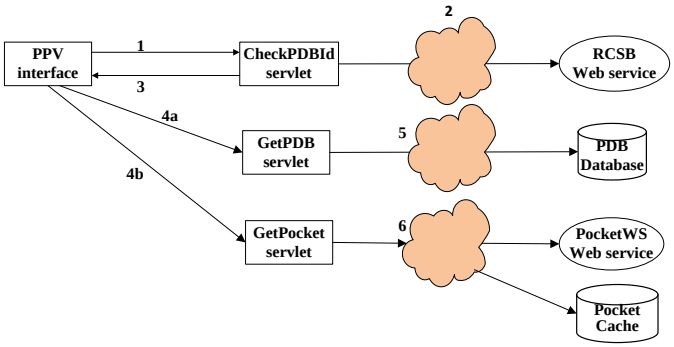

Fig. 4. The dataflow for pocket extraction and display.

C. Implementaion

The PPV was implemented using the Model 2 design pattern. The Model 2 design consists of model, view, and controller components.

1) View Related Components:

The PPV web interface was implemented using Java Server Pages and cascading style sheets, which define the layout and look-and-feel of the main interface.

Fig 5 shows the major components in the PPV web interface. A user can entera PDB Id and select the number of pockets in order to extract and visualize the protein pockets. If the number of pockets is not selected, then no pockets will be generated, but only the PDB file will be retrieved and rendered. The PDB information region displays metadata including PDB title, classification, andhyperlink to additional information on the PDB websites. Alsothe PDB sequence region displays the amino acid sequence of the protein.

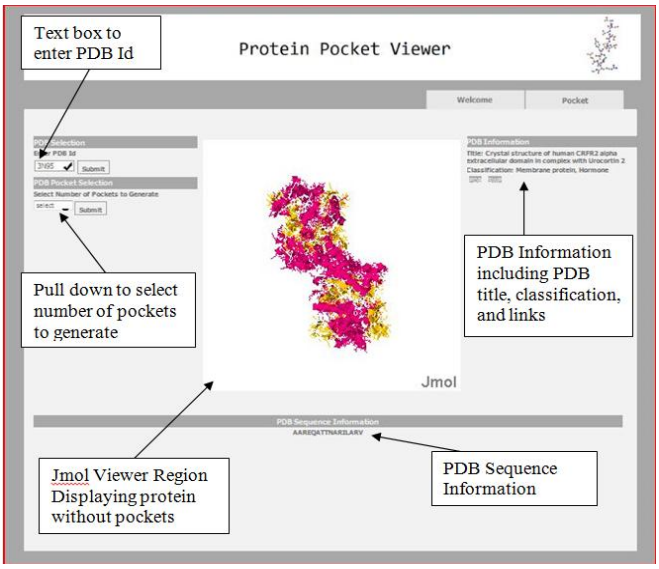

Fig. 5. Major components of the PPV web interface.

Fig 6 displays the protein pockets in the Jmol viewer and additional informationassociated with the pockets. Thecolor-coded pocket display options allow the user chooses to show or hide a pocket, or change its display as a filled surface, a dot surface or a mesh. The color-coded pocket information region displays metadata (volume and surface area) of the pockets. If the protein has fewer pockets than the number selected by the user, the maximum number of available pockets will be generated and displayed.

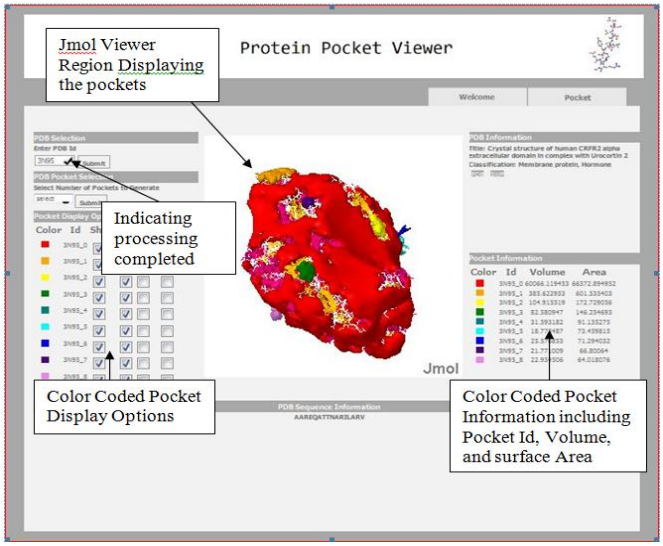

Fig. 6. Displaying multiple protein pockets in the PPV window.

Extracting pockets for a large protein can be CPU-intensive. Since the call to the server is handled asynchronously, the user can interact with currently display while the request is being processed. An in-progress icon is placed within the PDB ID input box as a visual cue, indicatinga request being processed. Atthe successful completion of the request, a checkmark icon replaces the in-progress icon. If there is an error in the processing, e.g. an invalid PDB ID, a red 'X' icon would indicate the failure of the request. Such a

visual indicator is very useful because of the asynchronous nature of request processing.

2) Controller Related Components:

The controller related components were implemented using JavaScript and Java servlet files.

There are four JavaScript files used in the current implementation. The JavaScript functions use AJAX for asynchronous execution. This asynchronous execution allows the user to interact with other components in the web application while the asynchronous calls execute. The JavaScript functions asynchronously call the appropriate servlets for services such as validating PDB ID, retrieving proteins, and extracting pockets and their metadata.

The Java servlets help to provide the controller aspect of the model-view-controller design pattern. The current PPV implementation has three Java servlet files, as shown in the middle layer of Fig 4 The servlets call appropriate methods provided by the business logic classes.

3) Model Related Components:

The PPV implemented a web service "PocketWS" for pocket extraction. The interface of the web service was defined in the "PocketWS" WSDL file, and implemented as a C++ web service class. The C++ implementation is a thin web service layer that calls upon the existing executable described in [1]with the appropriate arguments. The executable may be updated or replaced, without affecting the rest components of the PPV. Since extracting pocket is CPU intensive, the web service maintains a disk-resident cache of previously generated pockets. Atthe completion of the execution, it returns the result in XML format.

Besides the pocket extraction web service, the PPV also makes use of external web services, e.g. the RCSB web service "pdbWS" for PDB ID validation, and downloads PDB XML files using HTTP from PDB database. These business logics in the model related components wereimplementeda number of Java classes.

IV. Conclusion

The protein pocket viewer provides a web-based interface for protein pocket extraction and visualization based on the algorithm in [1]. It was implemented using web services and followed the Model 2 design pattern, consisting of a client web interface for user inputs and visualization, a middle-layer for controlling the flow, and the backend web services performing the actual CPU-intensive computation. Packaging the pocket extraction as a web service makes it as a single purpose, loosely coupled component that can be updated or replaced for a different algorithm easily. The PPV differs from other systems with features such as on-the-fly generation and visualization of pockets, asynchronous transactions via AJAX, and caching.

We found that it is effective to use web services to make existing programs available to users over the Internet with use of a modern browser.Because Java provides feature rich and widely supported graphical user interface, it was chosen to implement the web interface of the PPV.The existing executable of pocket extraction was written in C++. The web service solution bridged the cross domain and cross language solution of the existing C++ pocket executable with the Java web interface.

Future directions for the PPV may include the integration of related protein analysis capabilities, like a protein structure simulation, within the same architecture. A web application that provides multiple utilities sharing a common interface would make it more convenient for the user and ensure the compatibility of the analysis.

References

[1] X. Zhang and C. Bajaj, "Extraction, quantification and visualization of protein pockets.," Computational systems bioinformatics / Life Sciences Society. Computational Systems Bioinformatics Conference, vol. 6, pp. 275–86, Jan. 2007.

[2] J. D. Berman, "Structural properties of acetylcholinesterase from eel electric tissue and bovine erythrocyte membranes.," Biochemistry, vol. 12, no. 9, pp. 1710–5, May 1973.

[3] N. Unwin, "Refined structure of the nicotinic acetylcholine receptor at 4A resolution.," Journal of molecular biology, vol. 346, no. 4, pp. 967–89, Mar. 2005.

[4] W3C Working Group, "Web Services Architecture." [Online]. Available: http://www.w3.org/TR/ws-arch/.

[5] "Jmol: an open-source Java viewer for chemical structures in 3D." [Online]. Available: http://jmol.sourceforge.net/.

[6] H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov, and P. E. Bourne, "The Protein Data Bank.," Nucleic acids research, vol. 28, no. 1, pp. 235–42, Jan. 2000.

[7] J. J. Garrett, "Ajax: A New Approach to Web Applications - Adaptive Path." [Online]. Available: http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications.

[8] W3C, "Extensible Markup Language (XML) 1.0 (Fifth Edition)." [Online]. Available: http://www.w3.org/TR/REC-xml/.

[9] W3C, "SOAP Version 1.2 Part 1: Messaging Framework (Second Edition)." [Online]. Available: http://www.w3.org/TR/soap12-part1/#intro.

[10] W3C, "Web Services Description Language (WSDL) Version 2.0 Part 0: Primer." [Online]. Available: http://www.w3.org/TR/wsdl20-primer/.

[11] B. Basham, K. Sierra, and B. Bates, Head First Servlets and JSP. O'Reilly Media, 2004, p. 888.

[12] J. Zeldman, Designing With Web Standards. New Riders, 2003, p. 436.

[13] R. Harris, Murach's JavaScript and DOM Scripting. Mike Murach & Associates, Incorporated, 2009, p. 764.

[14] R. York, Beginning JavaScript and CSS Development with JQuery. Wiley, 2009, p. 529.

[15] G. Seshadri, "Understanding JavaServer Pages Model 2 architecture - Exploring the MVC design pattern," JavaWorld.com, 1999. [Online]. Available: http://www.javaworld.com/javaworld/jw-12-1999/jw-12-ssj-jspmvc.html.

[16] T. A. . Binkowski, "Castp: computed atlas of surface topography of proteins," Nucleic Acids Research, vol. 31, no. 13, pp. 3352–3355, Jul. 2003.

[17] J. Liang, H. Edelsbrunner, and C. Woodward, "Anatomy of protein pockets and cavities: measurement of binding site geometry and implications for ligand design.," Protein science : a publication of the Protein Society, vol. 7, no. 9, pp. 1884–97, Sep. 1998.

[18] J. Dundas, Z. Ouyang, J. Tseng, A. Binkowski, Y. Turpaz, and J. Liang, "CASTp: computed atlas of surface topography of proteins with structural and topographical mapping of functionally annotated residues.," Nucleic acids research, vol. 34, no. Web Server issue, pp. W116–8, Jul. 2006.

[19] "Apache Tomcat." [Online]. Available: http://tomcat.apache.org/.

[20] "Jmol Documentation." [Online]. Available: http://jmol.sourceforge.net/docs/JmolUserGuide/.