A Web-Based Interface for the NeuroSys Database Project

Submitted to the
Department of Computer Science
Montana State University

in Partial Fulfillment of
the Requirements for the Degree of
Master of Science
by
Stuart W. Howard

Supervised by
Dr. John Paxton

April 8, 2004

Table of Contents

Figures

Abstract

This paper describes and documents the implementation of a web-based interface for an existing database application. In response to the demand for managing and storing large amounts scientific data, programmers at the Center for Computational Biology at Montana State University have developed client-side software that allows a user to access, edit, and query a remote database. To use this application the client computer must have a current version of Java installed, the application must be downloaded, and browser settings may need to be adjusted. While many of the end users are computer savvy, the setup process challenges other end users. The goal of this project was to develop a server-side, web-based interface for the application that allows users to browse, query, and edit a database by simply visiting a website and logging in.

Principles of good design for user interfaces and interactive systems are reviewed and the resulting system is compared with three other database applications using these design principles as criteria. Results of the comparisons are summarized and recommendations for future work on the project are made.

Acknowledgements

I would like to thank Dr. Gwen Jacobs, Sandy Pittendrigh, and Gary Orser at the Center for Computational Biology at Montana State University for the opportunity to work on this project. Sandy, Gary, and the system administrator of the center, Ben Livingood, were always willing to offer technical help whenever I needed it. Many thanks to Dr. John Paxton for his excellent guidance and feedback as I wrote the paper. Lastly, I wish to thank my wife, Kim, for her encouragement and support as I worked on the project and paper.

1. Introduction

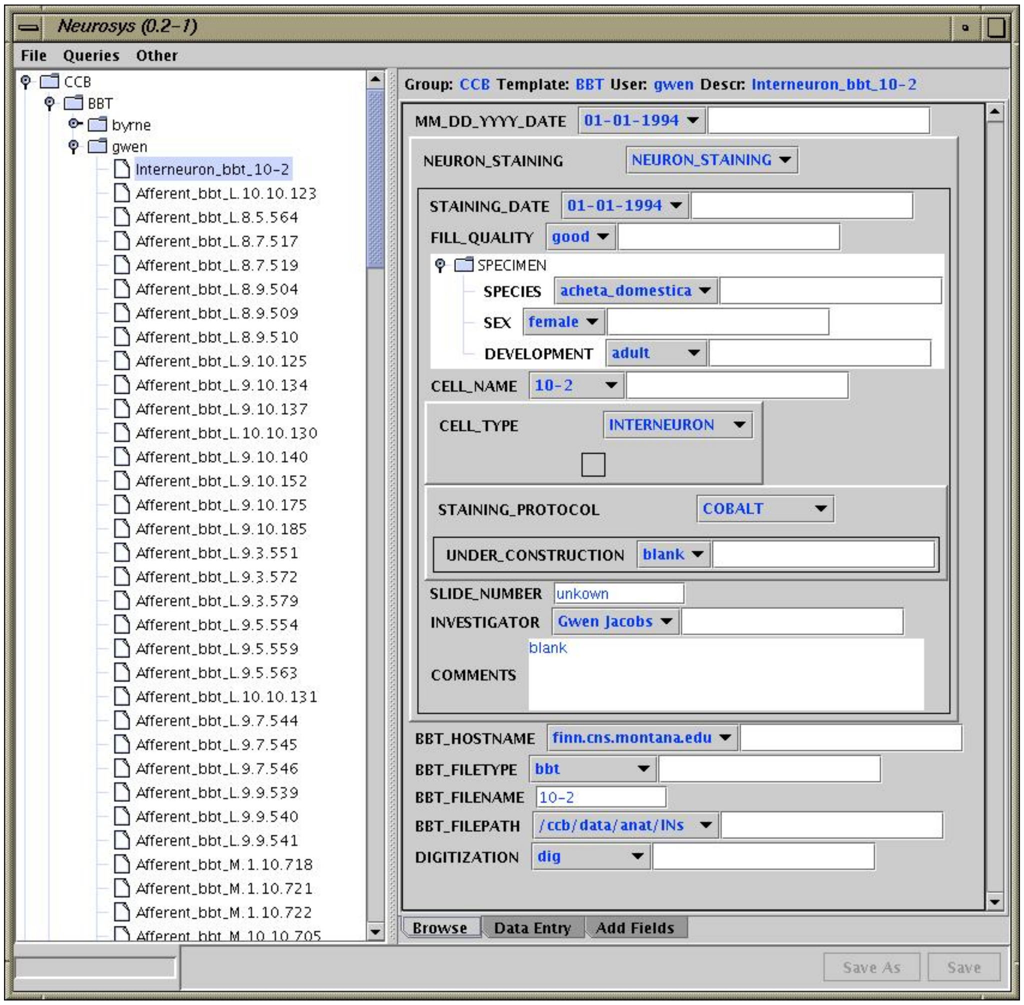
Scientific research, especially in the field of neuroscience, generates large amounts of often complex data. Such laboratory data are often stored in lab notebooks or spreadsheets [23]. Sharing of this data among peers is currently accomplished via published papers, which are not always accessible and are not machine readable. Published papers may present results on only averaged data or a subset of the data [5]. There is a need for database systems that allow such data to be organized, stored, reused, and easily shared among peers [23]. In response to these problems, there are many database projects underway.

Relational database systems can fill this gap; however the complexity inherent in the design, setup, and maintenance of such a system prohibits many scientists from this approach [23]. A related constraint that many laboratories face is the lack of resources to hire professional programmers and a database administrator to manage such a system [9]. There is a need for a system that allows researchers with minimal time and IT skills to record their data in an organized and network-accessible fashion without hiring professional programmers [23].

1.1 Project Background

In 2002 the Human Brain Project [11], a federal interagency initiative that supports research and development of information management systems for neuroscience, launched four programs to further the study of neuroinformatics. The Center for Computational Biology (CCB) at Montana State University received a grant from one of these programs and began the NeuroSys Database Project [22]. Aware of the obstacles of using relational database technology, the NeuroSys developers chose to store scientific data in the hierarchical format of XML. Data in this format can easily be stored to and retrieved from an XML database server in the form of a simple text file.

The CCB currently has an operational prototype software application [22] that allows users to access and query a remote XML database server with a GUI interface (Figure 1). However, to use this application the client computer must have Java Version 1.4 [18] installed, the application must be downloaded, and browser settings may need to be adjusted. While many of the end users are computer savvy, the setup process challenges other end users. The goal of this project is to develop a server-side, web-based interface for the NeuroSys project that allows users to browse, query, and edit a database by simply visiting a website and logging in. Therefore, the topic of this paper relates to the development of a web-based interface for the NeuroSys Database Project.



CCB
BBT
byrne
gwen
Interneuron_bbt_10-2
Afferent_bbt_L.10.10.123
Afferent_bbt_L.8.5.564
Afferent_bbt_L.8.7.517
Afferent_bbt_L.8.7.519
Afferent_bbt_L.8.9.504
Afferent_bbt_L.8.9.509
Afferent_bbt_L.8.9.510
Afferent_bbt_L.9.10.125
Afferent_bbt_L.9.10.134
Afferent_bbt_L.9.10.137
Afferent_bbt_L.10.10.130
Afferent_bbt_L.9.10.140
Afferent_bbt_L.9.10.152
Afferent_bbt_L.9.10.175
Afferent_bbt_L.9.10.185
Afferent_bbt_L.9.3.551
Afferent_bbt_L.9.3.572
Afferent_bbt_L.9.3.579
Afferent_bbt_L.9.5.554
Afferent_bbt_L.9.5.559
Afferent_bbt_L.9.5.563
Afferent_bbt_L.10.10.131
Afferent_bbt_L.9.7.544
Afferent_bbt_L.9.7.545
Afferent_bbt_L.9.7.546
Afferent_bbt_L.9.9.539
Afferent_bbt_L.9.9.540
Afferent_bbt_L.9.9.541
Afferent_bbt_M.1.10.718
Afferent_bbt_M.1.10.721
Afferent_bbt_M.1.10.722
Afferent_bbt_M.10.10.705

Group: CCB Template: BBT User: gwen Descr: Interneuron_bbt_10-2

MM_DD_YYYY_DATE	01-01-1994
NEURON_STAINING	NEURON_STAINING
STAINING_DATE	01-01-1994
FILL_QUALITY	good
SPECIMEN	
SPECIES	acheta_domestica
SEX	female
DEVELOPMENT	adult
CELL_NAME	10-2
CELL_TYPE	INTERNEURON
STAINING_PROTOCOL	COBALT
UNDER_CONSTRUCTION	blank
SLIDE_NUMBER	unkown
INVESTIGATOR	Gwen Jacobs
COMMENTS	blank
BBT_HOSTNAME	finn.cns.montana.edu
BBT_FILETYPE	bbt
BBT_FILENAME	10-2
BBT_FILEPATH	/ccb/data/anat/INs
DIGITIZATION	dig

Figure 1 - Interface for Client-side Application

1.2 Objective of Project

The specific objective of this project is to implement a web-based interface for the NeuroSys application that allows ‘*Browse*’, ‘*Data Entry*’, and ‘*Query*’ capabilities. The ‘*Browse*’ capability allows users to view records in a database. The ‘*Data Entry*’ capability allows users to edit their own records and also allows users to save and edit copies of records belonging to other users. The ‘*Query*’ capability allows a user to search for records matching some given parameters.

1.3 Outline of Remaining Chapters

Chapter Two discusses literature related to the topic of user interface design and introduces several other scientific database systems that provide a web-based interface. Chapter Three describes the underlying technologies used in the implementation of this project and explores the reasons for choosing them. Chapters Four and Five consist of user manuals for end users and for programmers respectively. Chapter Six compares and contrasts the project to the database systems introduced in Chapter Two. Chapter Seven summarizes and critically reviews the results of this project and offers recommendations for future work on the project.

2. Research and Literature Review

This chapter reviews literature related to the topic of user interface design and introduces several other scientific database systems that offer a web-based interface. The purpose of the chapter is to acquaint readers with the current state of affairs of such systems and with the guidelines that can be used to design web interfaces for them.

2.1 User Interfaces and Interactive Systems: Principles of Good Design

Though this project has some constraints on the design of the user interface as it mimics an existing application, the principles of good user interface design warrant some attention. An exploration of this topic may contribute to future interface design decisions concerning the NeuroSys project.

2.1.1 Design Goals for Human-Computer Interactive Systems

There is a good reason for the great interest in designing interactive systems. As often is the case the ‘bottom line’ has an influence here. IBM once dedicated $20,000 towards user-centered design over the seven month lifespan of a project they developed. The first three days the product was up and running a savings of $40,000 was realized [6]. Many software products have achieved commercial success due to their superior user interfaces [25].

Ben Shneiderman, author of *Designing the User Interface* [25], lists some design goals for interactive systems:

- *Determine the necessary functionality.*

  Determine what tasks and subtasks must be accomplished and how to handle emergency and recovery situations. Inadequate functionality frustrates users who then underutilize the system or do not use it at all. Excessive functionality can increase clutter and complexity and makes the system more difficult for users to learn.

- *Ensure reliability.*

  Commands and buttons must function properly. Displayed data must accurately reflect database contents and updates to data must be recorded and saved correctly to the database. If a user has one bad experience with data integrity or gets unexpected results, he will lose the trust and willingness to use the system.

- *Ensure availability.*

  The software and hardware components and the network system must ensure a high level of uptime. Human interface design is a moot point if the system is unavailable.

- *Ensure security, privacy, and data integrity.*

  Protect the integrity of the data by preventing unauthorized access.

Shneiderman [25] also mentions the attributes of *standardization*, *consistency*, and *portability* concerning interactive system design. Standardization means using common user interfaces across different applications. Consistency can apply to many aspects of a system including action sequences, colors, layouts, and terminology used. Portability refers to the system's capability to deploy among multiple hardware and software platforms.

2.1.2 Styles of Human-Computer Interaction (HCI)

Shneiderman [25] lists five types of human interaction with a computer: *direct manipulation, menu selection, form fill-in, command language, and natural language*. Direct manipulation means visually representing tasks and objects with icons and then selecting them with a cursor or pointer. In menu selection the user selects from a list of items and then observes the effect of his choice. Form fill-in is well suited for data entry, however this method consumes screen space and users must understand the field labels and know the permissible values for fields. This style is best for knowledgeable intermittent and frequent users. Command languages are very powerful and flexible, however they are prone to high error rates, poor error handling, and they require intensive training. This style is best suited for expert frequent users. The advantage of using a natural language to interact with a computer is there is no need to learn new syntax, but the advantage is quickly lost due to the frequent need for clarification dialogs. Natural language systems can also be unpredictable.

2.1.3 User Interface Design

Usability

Once the basic design goals of a system have been addressed and the interaction style (or combination of styles) has been selected, the actual design work for a user interface may begin. A user interface design should be evaluated for a specific user community and for specific benchmark tasks. A design for one community or set of tasks may not be efficient for another set of users or another set of tasks. Shneiderman [25] names five metrics in regards to human interaction. Other authors refer to the following factors as measurements of *usability* [6].

1. *Learnability.*

   The system should be easy to learn.

2. *Efficiency.*

   Users should be able to use the system productively.

3. *Errors.*

   The system should have a low error rate and good error handling. Error messages should be concise, understandable, and should suggest a solution. Users should be able to reverse an action (i.e. undo, redo).

4. *Memorability.*

   Operation of the system should be easy to remember.

5. *Satisfaction.*

   Users should enjoy using the system and be satisfied with their results.

Optimizing all of these factors simultaneously may not be possible as two goals can conflict with each other. A designer may need to sacrifice performance speed and efficiency to achieve a low error rate. The primary goals should be clear at the outset of the design process [25].

The Eight Golden Rules of User Interface Design

Shneiderman [25] suggests adapting the following guidelines to most interactive systems:

1. *Strive for consistency.*

   Use the same sequences of actions for similar situations. Use consistent terminology in prompts, menus, and help screens. Keep the appearance consistent.

2. *Enable shortcuts for frequent users.*

3. *Offer informative feedback.*

   For every action of the user the system should offer feedback. Feedback can be modest for frequent and minor actions, and more substantial for infrequent and major actions.

4. *Design dialogs to yield closure.*

   Action sequences should have a distinct beginning, middle, and end with feedback to the user to confirm his actions and to clear the way for the next action.

5. *Offer error prevention and simple error handling.*

6. *Permit easy reversal of actions.*

   This feature relieves user anxiety knowing that errors can be undone and it also encourages more exploration of the system.

7. *Support internal locus of control.*

   Experienced users want a sense of control and initiative. Unexpected system responses or tedious action sequences lead to frustration and anxiety. Users should initiate actions rather respond to them.

8. *Reduce short-term memory load.*

   George Miller [19] theorized that humans can process "seven, plus or minus two" chunks of information at a time. This popular theory suggests that users should not be required to memorize information between multiple displays.

Authors Mosier and Smith [26] list guidelines for data display and data entry that closely mirror the ones listed above.

2.1.4 Interface Design Guidelines Relevant to the Project

Since this project involves implementing an interface for a database application, the aspects of data display, data entry, and interaction style are especially important. Guidelines in the previous section may be applied to issues related to data display and data entry. Guidelines for the selected interaction style, form fill-in, are discussed in the next section.

Form Fill-in Design Guidelines

The form fill-in style is well suited for data entry tasks. The full complement of information is visible to the user and the user is empowered with a feeling of control. Few instructions are necessary because electronic forms resemble familiar paper forms [25]. Shneiderman [25] has gathered the following guidelines for designing forms:

1. *Meaningful title.*
2. *Comprehensible instructions.*
3. *Logical grouping and sequencing of fields.*
4. *Visually appealing form layout.*
5. *Familiar field labels.*
6. *Consistent terminology and abbreviations.*

7. *Visible space and boundaries for data entry fields.*
8. *Convenient cursor movement.*
9. *Error correction for individual characters and complete fields.*
10. *Error prevention.*
11. *Error messages for unacceptable values.*
12. *Optional fields clearly marked.*
13. *Explanatory message for fields.*
14. *Completion signal to direct user's next step.*

The previous sections of this chapter review principles of good design for user interfaces and human-computer interactive systems. The remainder of the chapter introduces three other systems that provide a web-based interface for a scientific database.

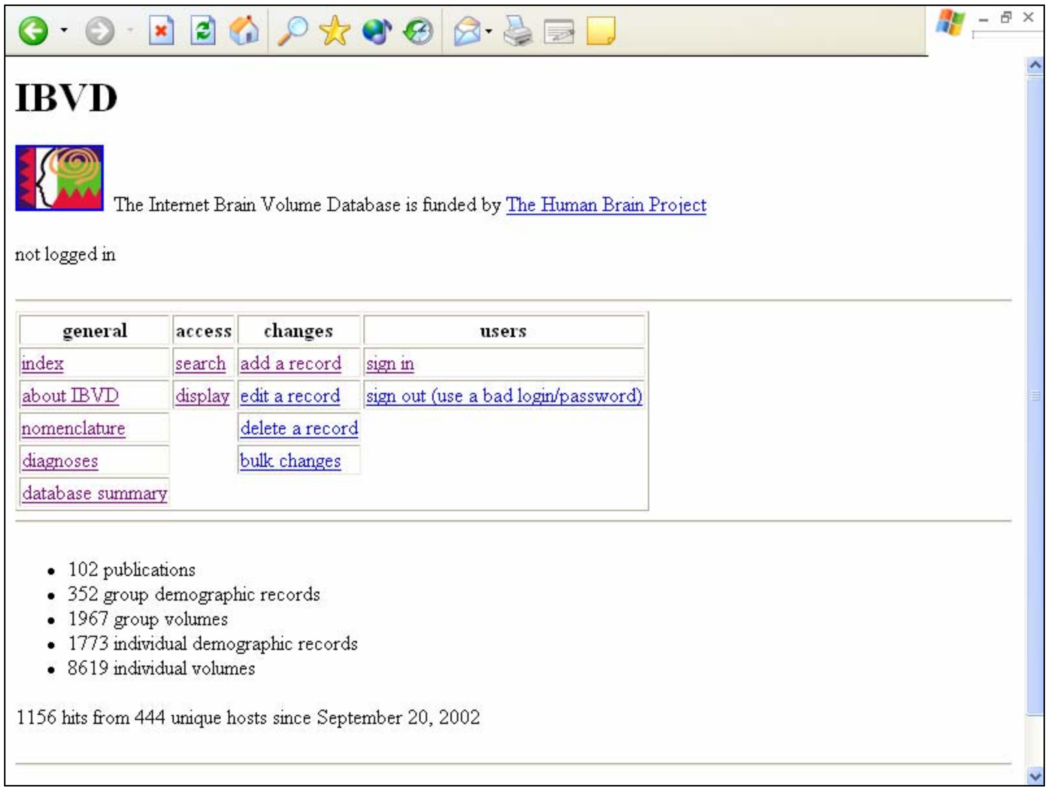
2.2 Similar Systems

Several other systems that provide a web interface to a scientific database are introduced. A survey of similar systems offers the reader a snapshot of the current state of affairs in web interfaces for scientific databases and provides a good background for a critical analysis of the NeuroSys web interface. As the systems are described, particular attention is paid to aspects relating to the appearance, design, and usability of the web interfaces. Other notable features are mentioned and Chapter 6 offers a more detailed analysis of the systems.

2.2.1 Internet Brain Volume Database (IBVD)

Researchers at Massachusetts General Hospital, Harvard Medical School, Massachusetts Institute of Technology, and several other institutions have implemented the Internet Brain Volume Database [13], a web-based resource containing data related to anatomical studies of the brain. The goal of the project is to improve the amount and quality of data available to neuroscience researchers and to overcome the paucity of data available previously only in journal articles.

The project homepage [13] is quite simple and unassuming. A table contains hyperlinks that describe and initiate each user action. The table headings and action descriptions all start with lowercase letters, which give the impression of an unfinished product (Figure 2).



IBVD

The Internet Brain Volume Database is funded by The Human Brain Project

not logged in

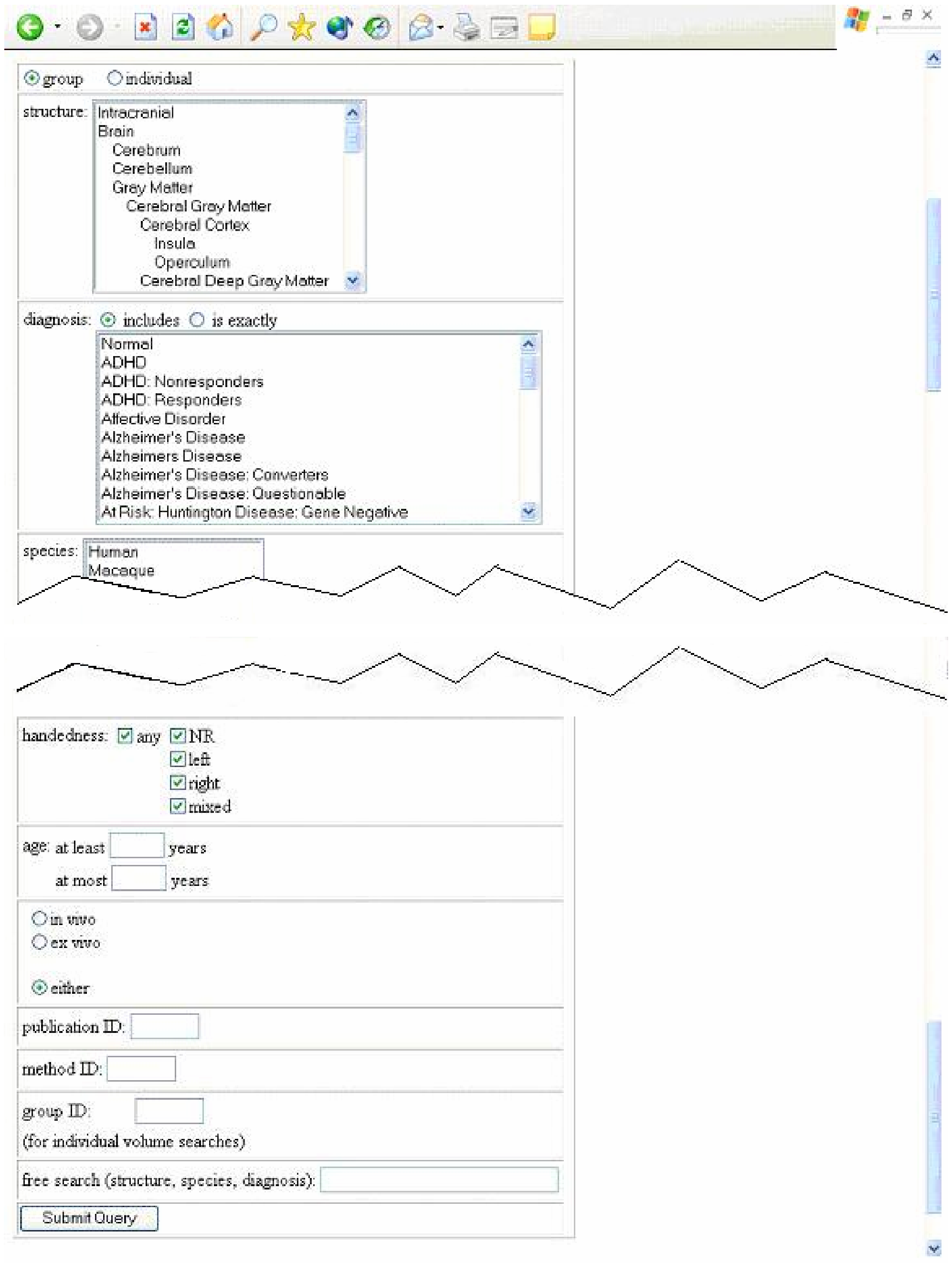
general	access	changes	users
index	search	add a record	sign in
about IBVD	display	edit a record	sign out (use a bad login/password)
nomenclature		delete a record	
diagnoses		bulk changes	
database summary			

- 102 publications
- 352 group demographic records
- 1967 group volumes
- 1773 individual demographic records
- 8619 individual volumes

1156 hits from 444 unique hosts since September 20, 2002

Figure 2 - IBVD Homepage

The '*about IBVD*' link leads to an excellent text-based walk-through tutorial of the database. Though it lists neuroscience terms in hierarchical fashion, the '*nomenclature*' link is not especially helpful since it does not provide definitions for the terms. More impressive is the '*search*' feature. When selected, the search link loads a form which spans two full screens (Figure 3). After some suggestions from the tutorial, a novice user is able to make a reasonable query. The results are displayed in an HTML table and the user has the options to graphically plot the data or to display and save the results as a flat text file. An authorized user has the option of adding, editing, or deleting a record. The placement of the ‘*login*’ link at the end of the table of user operations does not seem logical.


[● group] ○ individual

structure:
Intracranial
Brain
Cerebrum
Cerebellum
Grey Matter
Cerebral Gray Matter
Cerebral Cortex
Insula
Operculum
Cerebral Deep Gray Matter

diagnosis: [● includes] ○ is exactly
Normal
ADHD
ADHD: Nonresponders
ADHD: Responders
Affective Disorder
Alzheimer's Disease
Alzheimers Disease
Alzheimer's Disease: Converters
Alzheimer's Disease: Questionable
At Risk: Huntington Disease: Gene Negative

species:
Human
Macaque

handedness: [✓ any] [✓ NR] [✓ left] [✓ right] [✓ mixed]

age: at least ____ years
at most ____ years

○ in vivo
○ ex vivo
[● either]

publication ID:

method ID:

group ID:
(for individual volume searches)

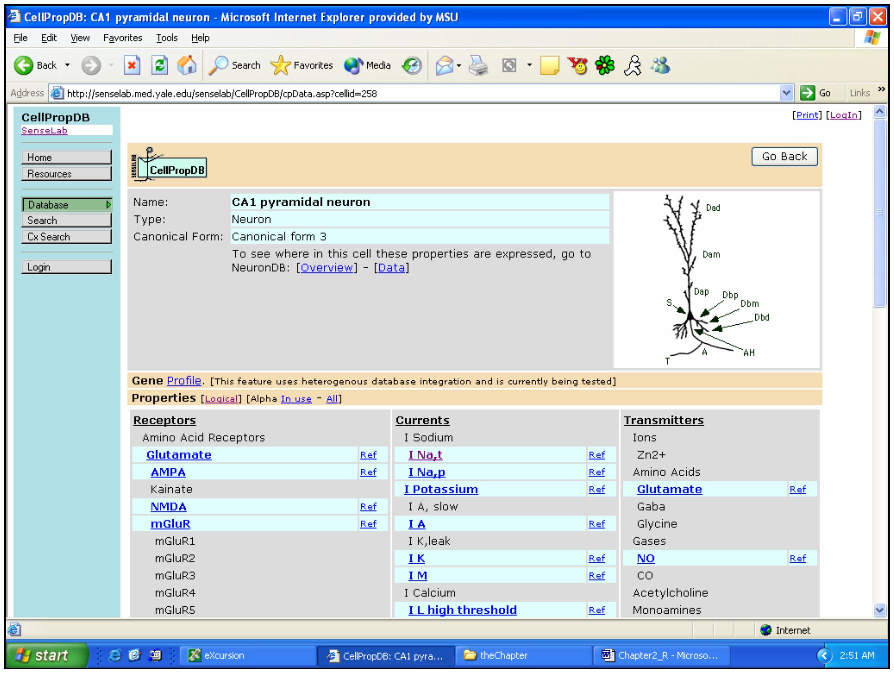
free search (structure, species, diagnosis):

Figure 3 - IBVD Search Page

2.2.2 SenseLab

The SenseLab project [24] was started in 1993 by researchers at Yale who wished to manage different types of neuroinformatics data. It was originally implemented as four separate databases with a common web interface. As the amount and the diversity of the data increased, so did the complexity of interconnecting the databases and the web. By adopting a different framework for the database schema, the researchers were able to consolidate the four databases into a single physical database [20]. There are currently six neuronal and olfactory databases housed at the SenseLab homepage [24].

Upon entering the Cellular Properties Database (CellPropDB) [7] and selecting from a column of hyperlinked neuronal cell names, the user is presented with the screen shown in Figure 4. The data is neatly displayed with all relevant fields highlighted by an aqua background. Clicking a relevant attribute which is hyperlinked will show a table of all other cells with the same attribute. Selecting the '*Ref*' link to the right of the attribute leads to a detailed description of the attribute and related journal references, which are also hyperlinked to sources for the articles (Figure 5).



http://senselab.med.yale.edu/senselab/CellPropDB/cpData.asp?cellid=258

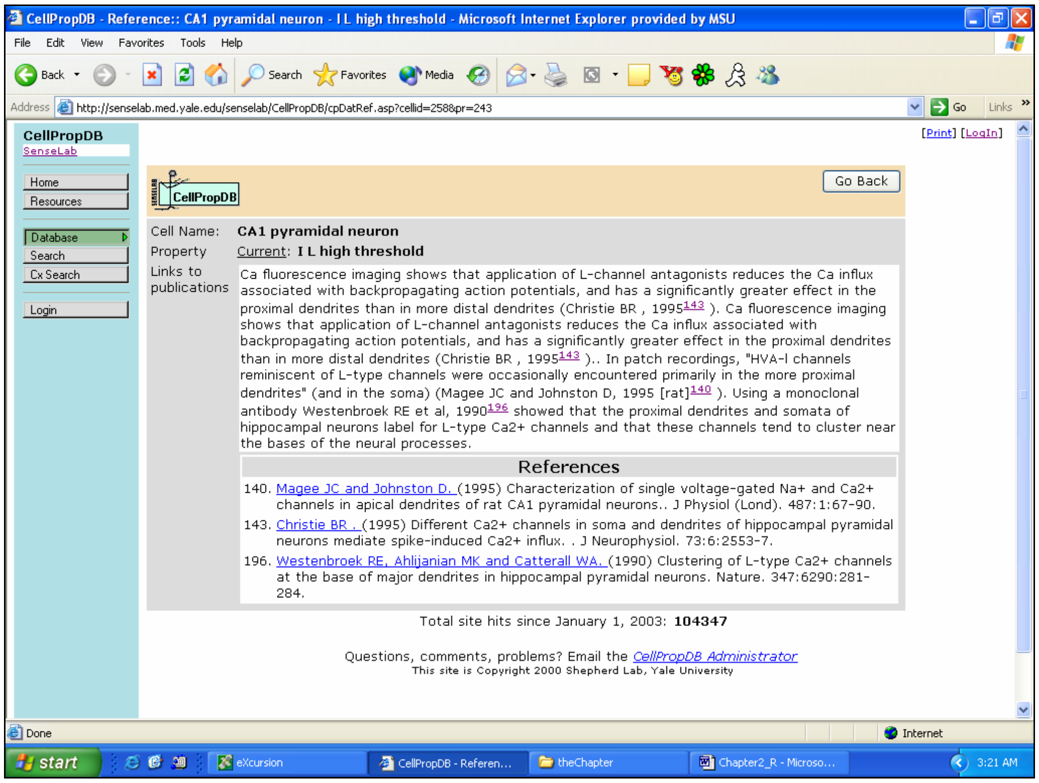
Name:	CA1 pyramidal neuron
Type:	Neuron
Canonical Form:	Canonical form 3
	To see where in this cell these properties are expressed, go to NeuronDB: [Overview] - [Data]

Gene Profile. [This feature uses heterogenous database integration and is currently being tested]

Properties [Logical] [Alpha In use - All]

Receptors		Currents		Transmitters	
Amino Acid Receptors		I Sodium		Ions	
Glutamate	Ref	I Na,t	Ref	Zn2+	
AMPA	Ref	I Na,p	Ref	Amino Acids	
Kainate		I Potassium	Ref	Glutamate	Ref
NMDA	Ref	I A, slow		Gaba	
mGluR	Ref	I A	Ref	Glycine	
mGluR1		I K,leak		Gases	
mGluR2		I K	Ref	NO	Ref
mGluR3		I M	Ref	CO	
mGluR4		I Calcium		Acetylcholine	
mGluR5		I L high threshold	Ref	Monoamines	

Figure 4 - CA1 pyramidal neuron


CellPropDB - Reference:: CA1 pyramidal neuron - I L high threshold - Microsoft Internet Explorer provided by MSU
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://senselab.med.yale.edu/senselab/CellPropDB/cpDatRef.asp?cellid=258&pr=243
Go Links
CellPropDB
SenseLab
Home
Resources
Database
Search
Cx Search
Login
[Print] [LogIn]
CellPropDB
Go Back
Done
Internet
start
eXcursion
CellPropDB - Referen...
theChapter
Chapter2_R - Microso...
3:21 AM

Cell Name: **CA1 pyramidal neuron**

Property Current: **I L high threshold**

Links to publications

Ca fluorescence imaging shows that application of L-channel antagonists reduces the Ca influx associated with backpropagating action potentials, and has a significantly greater effect in the proximal dendrites than in more distal dendrites (Christie BR , 1995[143]). Ca fluorescence imaging shows that application of L-channel antagonists reduces the Ca influx associated with backpropagating action potentials, and has a significantly greater effect in the proximal dendrites than in more distal dendrites (Christie BR , 1995[143]).. In patch recordings, "HVA-l channels reminiscent of L-type channels were occasionally encountered primarily in the more proximal dendrites" (and in the soma) (Magee JC and Johnston D, 1995 [rat][140]). Using a monoclonal antibody Westenbroek RE et al, 1990[196] showed that the proximal dendrites and somata of hippocampal neurons label for L-type Ca2+ channels and that these channels tend to cluster near the bases of the neural processes.

References

140. Magee JC and Johnston D. (1995) Characterization of single voltage-gated Na+ and Ca2+ channels in apical dendrites of rat CA1 pyramidal neurons.. J Physiol (Lond). 487:1:67-90.

143. Christie BR . (1995) Different Ca2+ channels in soma and dendrites of hippocampal pyramidal neurons mediate spike-induced Ca2+ influx. . J Neurophysiol. 73:6:2553-7.

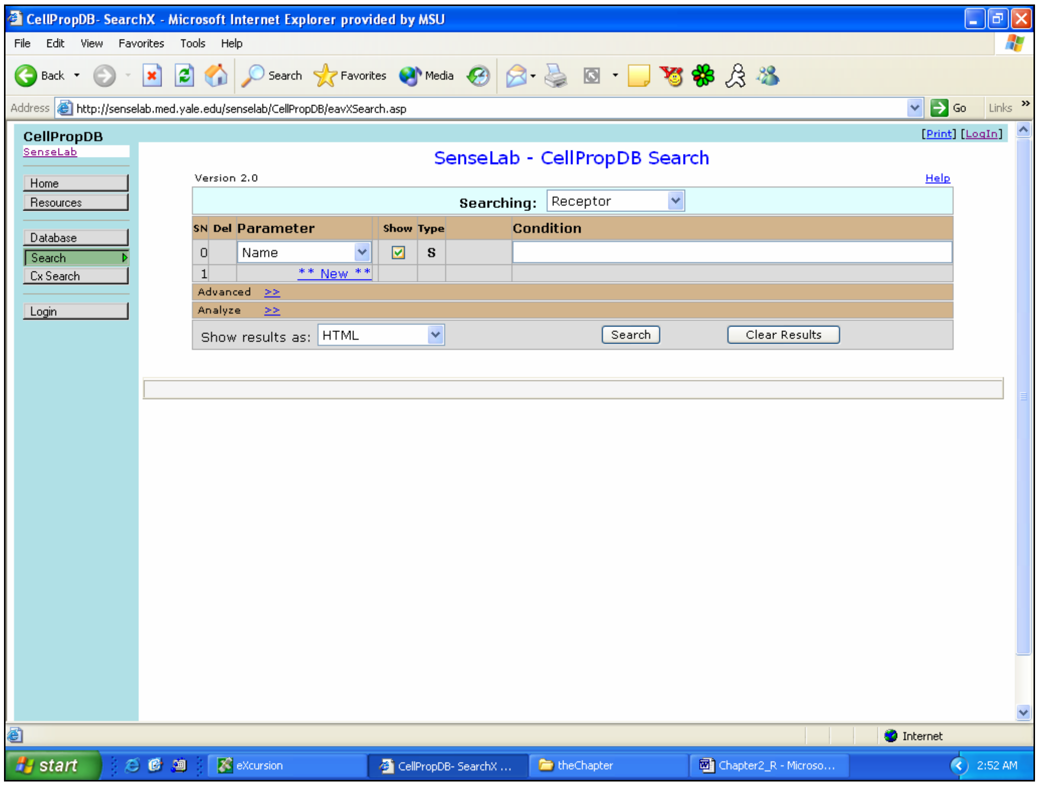
196. Westenbroek RE, Ahlijanian MK and Catterall WA. (1990) Clustering of L-type Ca2+ channels at the base of major dendrites in hippocampal pyramidal neurons. Nature. 347:6290:281-284.

Total site hits since January 1, 2003: **104347**

Questions, comments, problems? Email the *CellPropDB Administrator*

This site is Copyright 2000 Shepherd Lab, Yale University

Figure 5 - I L high threshold reference



SenseLab - CellPropDB Search

Version 2.0

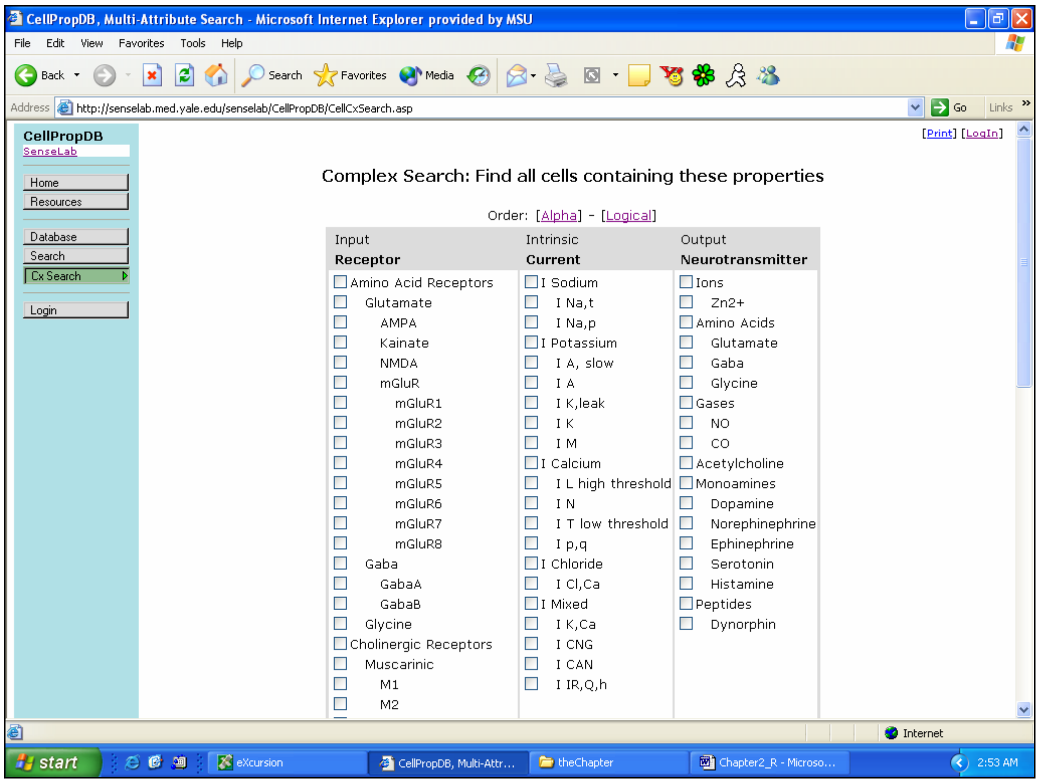
Searching: Receptor

SN	Del	Parameter	Show	Type	Condition
0		Name	✓	S	
1		** New **			

Show results as: HTML

Figure 6 - Standard Search Screen

The database has an excellent search capability. The search buttons are located in the left frame and a standard search screen is shown in Figure 6. The complex search button (*Cx Search*) brings up a table of checkboxes that can be selected in any combination desired. A nice feature of the complex search screen is the user's ability to control the display of the properties either alphabetically or logically (Figure 7).


CellPropDB, Multi-Attribute Search - Microsoft Internet Explorer provided by MSU
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://senselab.med.yale.edu/senselab/CellPropDB/CellCxSearch.asp
Go Links
CellPropDB
SenseLab
Home
Resources
Database
Search
Cx Search
Login
[Print] [LogIn]
Internet
start
eXcursion
CellPropDB, Multi-Attr...
theChapter
Chapter2_R - Microso...
2:53 AM

Complex Search: Find all cells containing these properties

Order: [Alpha] - [Logical]

Input	Intrinsic	Output
Receptor	**Current**	**Neurotransmitter**
☐ Amino Acid Receptors	☐ I Sodium	☐ Ions
☐ Glutamate	☐ I Na,t	☐ Zn2+
☐ AMPA	☐ I Na,p	☐ Amino Acids
☐ Kainate	☐ I Potassium	☐ Glutamate
☐ NMDA	☐ I A, slow	☐ Gaba
☐ mGluR	☐ I A	☐ Glycine
☐ mGluR1	☐ I K,leak	☐ Gases
☐ mGluR2	☐ I K	☐ NO
☐ mGluR3	☐ I M	☐ CO
☐ mGluR4	☐ I Calcium	☐ Acetylcholine
☐ mGluR5	☐ I L high threshold	☐ Monoamines
☐ mGluR6	☐ I N	☐ Dopamine
☐ mGluR7	☐ I T low threshold	☐ Norephinephrine
☐ mGluR8	☐ I p,q	☐ Ephinephrine
☐ Gaba	☐ I Chloride	☐ Serotonin
☐ GabaA	☐ I Cl,Ca	☐ Histamine
☐ GabaB	☐ I Mixed	☐ Peptides
☐ Glycine	☐ I K,Ca	☐ Dynorphin
☐ Cholinergic Receptors	☐ I CNG	
☐ Muscarinic	☐ I CAN	
☐ M1	☐ I IR,Q,h	
☐ M2		

Figure 7 - Complex Search Screen

2.2.3 Axiope Catalyzer

Axiope Catalyzer [3] is a database system for managing and sharing scientific data that was developed by researchers at the Informatics Division at Edinburgh University in Scotland. It is more similar to the NeuroSys project than the centralized Internet Brain Volume Database and SenseLab systems in that it provides client-side software for building user-defined forms on the fly. This capability is touted as the system's most important feature [9].

In the Axiope Catalyzer system records are stored in ‘*catalogs*’, analogous to a database name, and conform to a particular ‘*class*’, or schema. A catalog can contain records belong to any number of classes. The catalog can be converted to a browsable set of web pages by selecting the ‘*Publish Catalog*’ link from the application’s main menu (Figure 8). The catalog website can be viewed from a local directory or can be uploaded to a web server for public access (Figure 9).

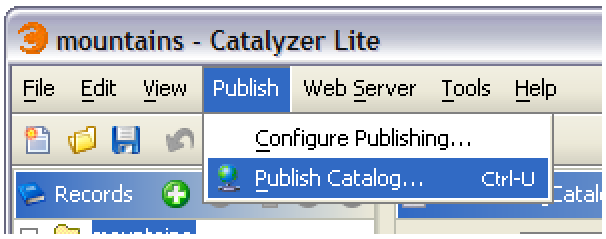
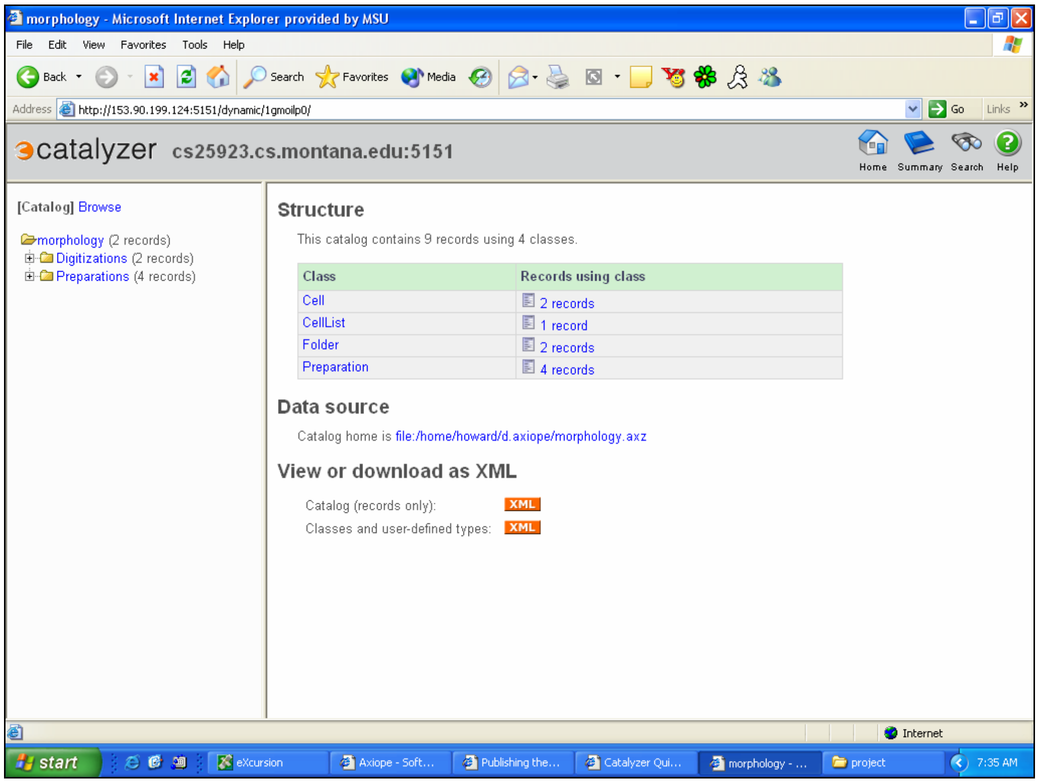


Figure 8 - Axiope Catalyzer Publish Menu



[Catalog] Browse

morphology (2 records)
Digitizations (2 records)
Preparations (4 records)

Structure

This catalog contains 9 records using 4 classes.

Class	Records using class
Cell	2 records
CellList	1 record
Folder	2 records
Preparation	4 records

Data source

Catalog home is file:/home/howard/d.axiope/morphology.axz

View or download as XML

Catalog (records only): XML
Classes and user-defined types: XML

Figure 9 - Axiope Catalyzer Catalog Website

It should be noted that the web interface does not allow the user to add, edit, or delete records. The functionality of the web interface will be discussed in more detail in Chapter 6.

This chapter reviewed literature related to the topic of user interface design and introduced several other scientific database systems that offer a web-based interface. Chapter Three describes the underlying technologies used in the implementation of the NeuroSys web interface project.

3. Underlying Technologies

The challenge of mimicking a Java Swing-based [17] application with a server-side HTML interface involves the use of varying technologies:

- HTML
- Java Servlets to create dynamically generated HTML
- Apache Tomcat servlet container
- JavaScript scripting language for dynamic updating of frames

This project also demands familiarity with the back end Apache Xindice XML database server [2] and the XPath query language [32]. This chapter describes these underlying technologies and where appropriate, offers reasons for the choice of a particular technology.

3.1 HTML

The project application produces HTML that conforms to the HTML 4.01 Specification [10]. If the reader is not familiar with HTML (HyperText Markup Language), consult the following websites for more information:
http://www.w3.org/MarkUp/ [27]
http://www.w3schools.com/html/default.asp [29]

The next section describes technologies that are often used in web applications.

3.2 Web Application Technology

There are several technologies for producing web applications that generate dynamic content in response to client requests. The following sections describe two of those technologies: *Common Gateway Interface* and *Java Servlets*.

3.2.1 Common Gateway Interface (CGI)

The Common Gateway Interface was the first technique for creating dynamically generated web pages. Though it was intended as a standard for communication between information servers and external applications, the use of CGI quickly became a standard for creating web applications [12]. With this technique a server passes a client request to an external program which creates some content and then sends a response back to the client. Though CGI allowed web developers exciting, new options for adding functionality to web pages, its life cycle is not optimal for a web server environment [12].

When a server receives a request that calls a CGI program, it creates a new process to run the CGI code (Figure 10) [8]. This not only leads to high overhead with one process per client request, but also limits the number of concurrent requests that a server can handle. Another disadvantage is that a CGI program can not interact with the web server or utilize its capabilities once it begins execution because it is a separate process [12].

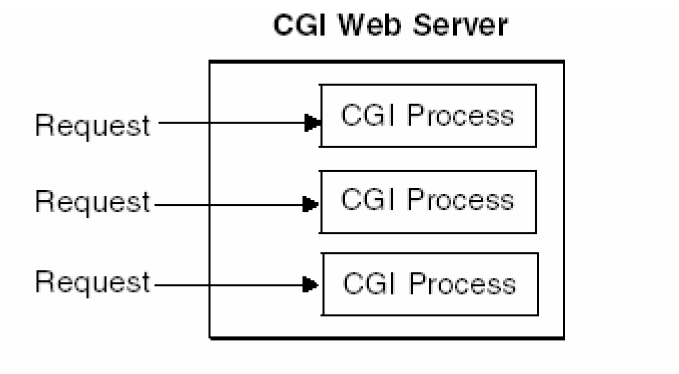


Figure 10 - CGI Processes

3.2.2 Java Servlets

A servlet is compiled Java code that can be loaded dynamically into a running server to extend its functionality. Servlets are most often used to extend web servers and to create dynamic web page content in response to user interaction, thus creating a web *application* versus a static web *page* [12].

Servlets are similar to CGI in that they allow a program to process a request and produce a dynamic response [8]. The main difference between the two methods is the life cycle. Unlike the multiple process paradigm of CGI, servlets run from within a single process and handle client requests via multiple lightweight threads (Figure 11) [8]. This approach leads not only to a more efficient use of resources, but is also scalable and allows the parent process to access and share resources with their threads and other servlets [8].

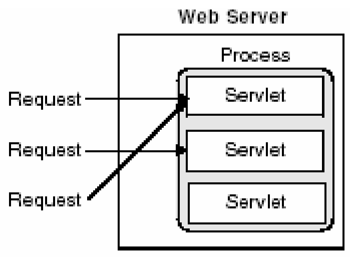


Figure 11 - Servlet Process

There are other reasons for using Java servlets. Java code is highly portable among different operating systems and server implementations [12]. Java servlets also have access to a standard and powerful API (Application Programming Interface) that allows networking, database connectivity, and many other useful features. In addition the Java Servlet API [15] handles routine tasks associated with servlets. Errors can be handled gracefully with Java's exception handling capabilities and Java's automatic garbage collection can help prevent memory leaks.

3.2.2.1 Servlet Life Cycle

Knowing the basics of the servlet life cycle is helpful for comparing servlet technology to other paradigms and is important for understanding the Servlet API. Servlets follow a simple three-phase life cycle: *initialization*, *service*, and *destruction* (Figure 12) [8].

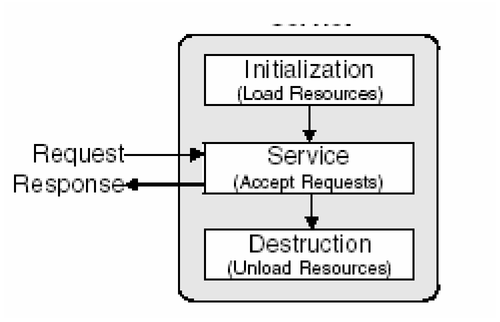


Figure 12 - The Servlet Life Cycle

The initialization phase represents the creation and initialization of resources required to service any requests. All servlets must implement the `javax.servlet.Servlet` interface which defines the corresponding `init()` method for this phase of the cycle. The service phase involves handling zero or more requests from clients. The `Servlet` interface defines the `service()` method for this phase. The `service()` method is invoked once per request and is responsible for generating a reply. As mentioned before servlets create a thread for each request and it is this thread that executes the `service()` method. The destruction phase entails a call to the `destroy()` method of the `Servlet` interface which disallows more requests to a particular servlet instance, destroys the servlet, and performs garbage collection [8].

3.2.2.2 Apache Tomcat Servlet Container

The servlet environment requires not only Java support but a *container* for the servlets to reside in. A container is a piece of software that manages the servlet life cycle and is responsible for loading, executing, and unloading the servlets. A container can be a module installed within an existing web server or in some cases can act as the web server itself, as is the case with the Apache Tomcat container [8]. The Java Servlet Specification [15] defines the API that a container must implement and many different vendors implement servlet containers.

The CCB chose the open source Apache Tomcat container [1] because it is the official Sun J2EE Reference Implementation for servlets. Downloads and documentation are available at the Apache Tomcat homepage [1].

3.2.3 JavaScript

In addition to leveraging servlets for producing dynamic content, the application also employs a scripting language call JavaScript [16]. JavaScript was developed by Netscape and is now supported by all major web browsers. JavaScript consists of executable code that can be embedded directly in HTML pages. JavaScript code can place dynamic text into an HTML page, read and write HTML elements, and react to events [30]. In this project JavaScript is used primarily for dynamically updating a frame displaying a form if a change has been made to the form. A sequence of screenshots that involve JavaScript will be shown in Chapter 4.

3.3 Apache Xindice XML Database

Implementation of a web-based front end for a database application requires some knowledge of the inner workings of the back end database. The developers at the CCB chose to use an open source XML database system called Apache Xindice [2]. Xindice is a database designed solely to store XML data and thus can be referred to as a *native* XML database. XML documents are stored in '*collections*' which are arranged in hierarchical fashion like a UNIX file system. Data is stored and retrieved as simple XML text files without the possibly complex mapping between tables that a relational database requires. The desire to avoid the complexity inherent in relational database design and implementation is the primary reason for the choice of an XML database for the NeuroSys project [22].

Each XML record in the database is complete and autonomous, with no bindings to other data structures. Furthermore, any node in an XML file can be uniquely identified by its path to the root element therefore allowing the node and corresponding data value to be easily mapped to a GUI object (button, form field, or any other ‘widget’) which can then be displayed on the fly. Likewise it is possible for users of a program to interactively create their own database forms and map the form elements to label-value pairs in an XML file [22]. This valuable feature is provided in the client-side Java Swing version of NeuroSys.

This autonomy does come at the price of storing redundant data. Well-designed relational databases store common and recurring data in only one table without the need for storing data redundantly. Updating multiple XML files (records) entails more overhead than updating a single table in a relational database and redundant data can introduce errors more easily [22].

For database operations like retrieving files, saving files, and deleting files the project application is able to leverage the API provided by the client-side version of NeuroSys. For development and debugging purposes it is necessary to know the basic command line database commands for listing collections and documents, retrieving and deleting documents, and adding new documents. The project application must also query the database occasionally. Xindice supports the XML Path Language (XPath) as defined by the World Wide Web Consortium (W3C) [32].

Shown below are two code snippets that define XPath query strings used in the application. The first query string is a parameter to a method that retrieves all records in the collection. The second query string retrieves documents that belong only to the current user of the interface. The code is taken from the `displayHyperlinks()` method in `htmlHandler.HtmlGenerator.java`.

```
1) String xpath = "/EML/HEAD";
2) xpath = "/EML[HEAD/OWNER[text()='" + user_name + "']]/HEAD";
```

Shown below is a XML file from the Xindice database that displays a simple form with a text field, a text area, and a pull down menu.

```
<?xml version="1.0" encoding="UTF-8"?>
<EML>
  <HEAD>
    <DISPLAYVIEW CATEGORY="CCB">
      <DISPLAYVIEW CATEGORY="STU">
        <DISPLAYVIEW CATEGORY="stuarth"></DISPLAYVIEW>
      </DISPLAYVIEW>
    </DISPLAYVIEW>
    <TEMPLATE>STU</TEMPLATE>
    <TEMPLATE_DOC_ID>STU_0.1</TEMPLATE_DOC_ID>
    <DOC_ID>STU-stuarth-03_16_2004-3</DOC_ID>
    <DESCRIPTION>mix</DESCRIPTION>
    <OWNER>stuarth</OWNER>
    <GROUP>CCB</GROUP>
    <CREATIONDATE>03_16_2004</CREATIONDATE>
  </HEAD>
  <BODY>
    <TextField1 DATATYPE="String">a</TextField1>
    <TextArea1 DATATYPE="String">b</TextArea1>
    <PullDown1 DATATYPE="String">opt7</PullDown1>
  </BODY>
</EML>
```

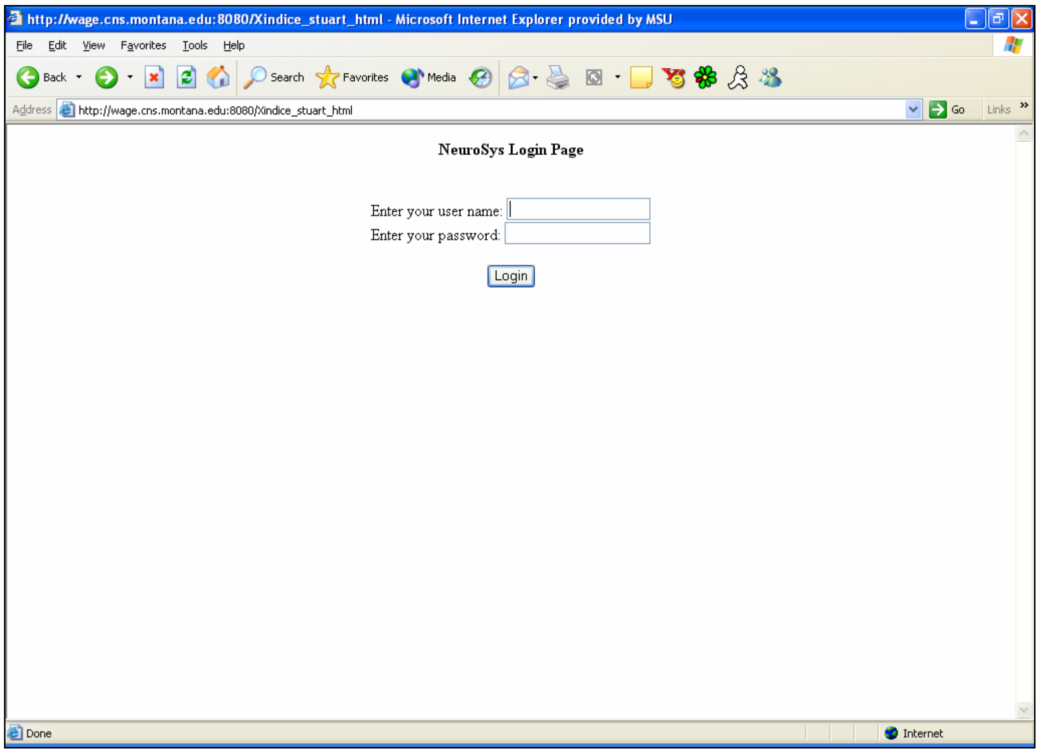
This chapter discusses the underlying technologies that are used in the implementation of the web-based interface for the NeuroSys Database Project. Chapter Four describes how to use the web-based interface and serves as a manual for the end user.

4. Functional Specifications for Users

This chapter serves as a manual for an end user. The steps of logging in, navigating the interface, browsing records, and editing and saving records are described and corresponding screenshots are displayed.

4.1 Logging In

A simple log in screen is provided for secure access to a database. The user enters a user name and password and presses the button labeled '*login*' (Figure 13). If the username and password are legal, the user is presented with the database interface described in the next section.


http://wage.cns.montana.edu:8080/Xindice_stuart_html - Microsoft Internet Explorer provided by MSU
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://wage.cns.montana.edu:8080/Xindice_stuart_html
Go Links
Done
Internet

NeuroSys Login Page

Enter your user name:

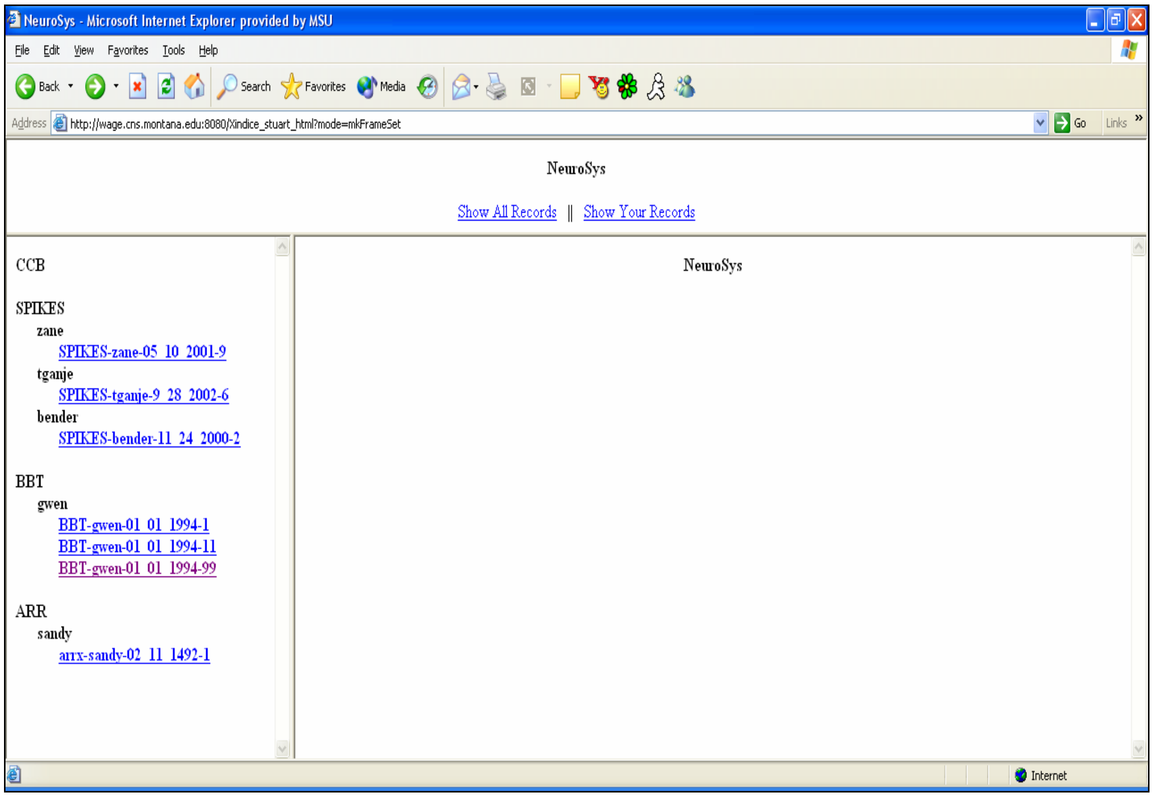
Enter your password:

Login

Figure 13 - Login Page

4.2 Database Interface

The database interface consists of three frames (Figure 14). The top frame displays the database name and two hyperlinks that are described in the next section. The left frame serves as the navigation panel for the database and contains a list of hyperlinked record names. The right frame is the form display area and initially appears with only some database identification information. Selection of a hyperlinked record name in the left frame displays the record in form fashion in the right frame (Figure 15).


NeuroSys - Microsoft Internet Explorer provided by MSU
File Edit View Favorites Tools Help
Back Search Favorites Media
Address
Go Links
Internet

http://wage.cns.montana.edu:8080/Xindice_stuart_html?mode=mkFrameSet

NeuroSys

Show All Records || Show Your Records

CCB

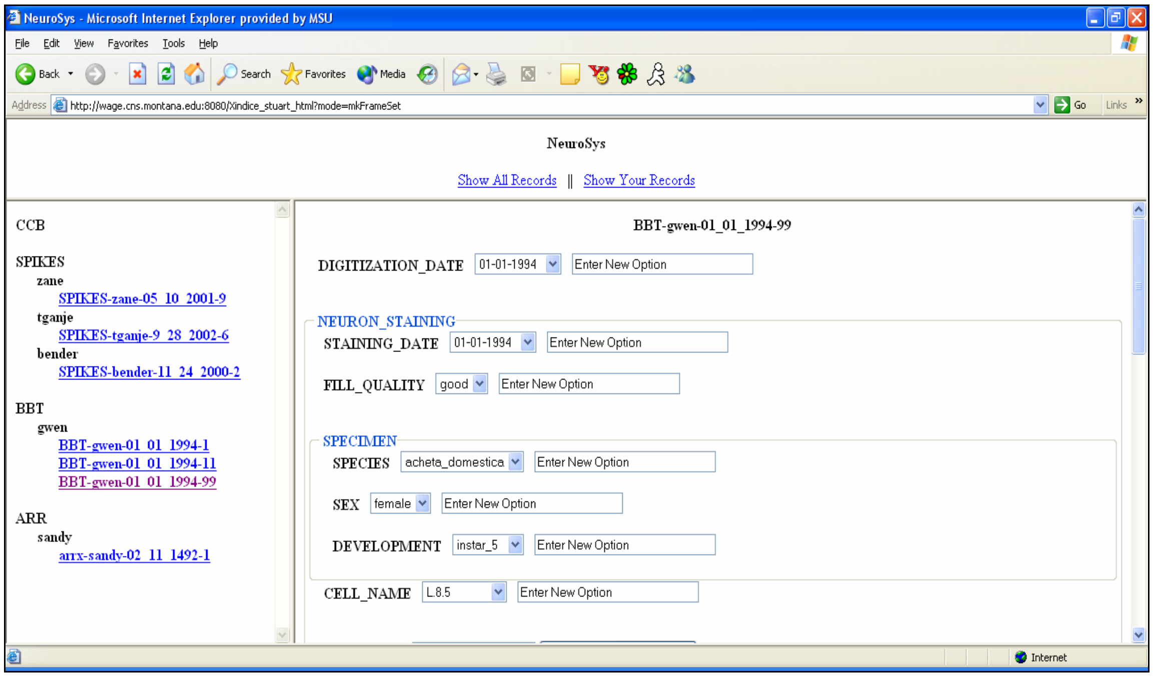
SPIKES
zane
SPIKES-zane-05_10_2001-9
tganje
SPIKES-tganje-9_28_2002-6
bender
SPIKES-bender-11_24_2000-2

BBT
gwen
BBT-gwen-01_01_1994-1
BBT-gwen-01_01_1994-11
BBT-gwen-01_01_1994-99

ARR
sandy
arrx-sandy-02_11_1492-1

NeuroSys

Figure 14 - Initial Frame Layout


NeuroSys - Microsoft Internet Explorer provided by MSU
File Edit View Favorites Tools Help
Back Search Favorites Media
Address
Go Links
Internet

http://wage.cns.montana.edu:8080/Xindice_stuart_html?mode=mkFrameSet

NeuroSys

Show All Records || Show Your Records

CCB

SPIKES
- zane
  - SPIKES-zane-05_10_2001-9
- tganje
  - SPIKES-tganje-9_28_2002-6
- bender
  - SPIKES-bender-11_24_2000-2

BBT
- gwen
  - BBT-gwen-01_01_1994-1
  - BBT-gwen-01_01_1994-11
  - BBT-gwen-01_01_1994-99

ARR
- sandy
  - arrx-sandy-02_11_1492-1

BBT-gwen-01_01_1994-99

DIGITIZATION_DATE 01-01-1994 Enter New Option

NEURON_STAINING
STAINING_DATE 01-01-1994 Enter New Option
FILL_QUALITY good Enter New Option

SPECIMEN
SPECIES acheta_domestica Enter New Option
SEX female Enter New Option
DEVELOPMENT instar_5 Enter New Option

CELL_NAME L8.5 Enter New Option

Figure 15 - Form Displayed in Right Frame

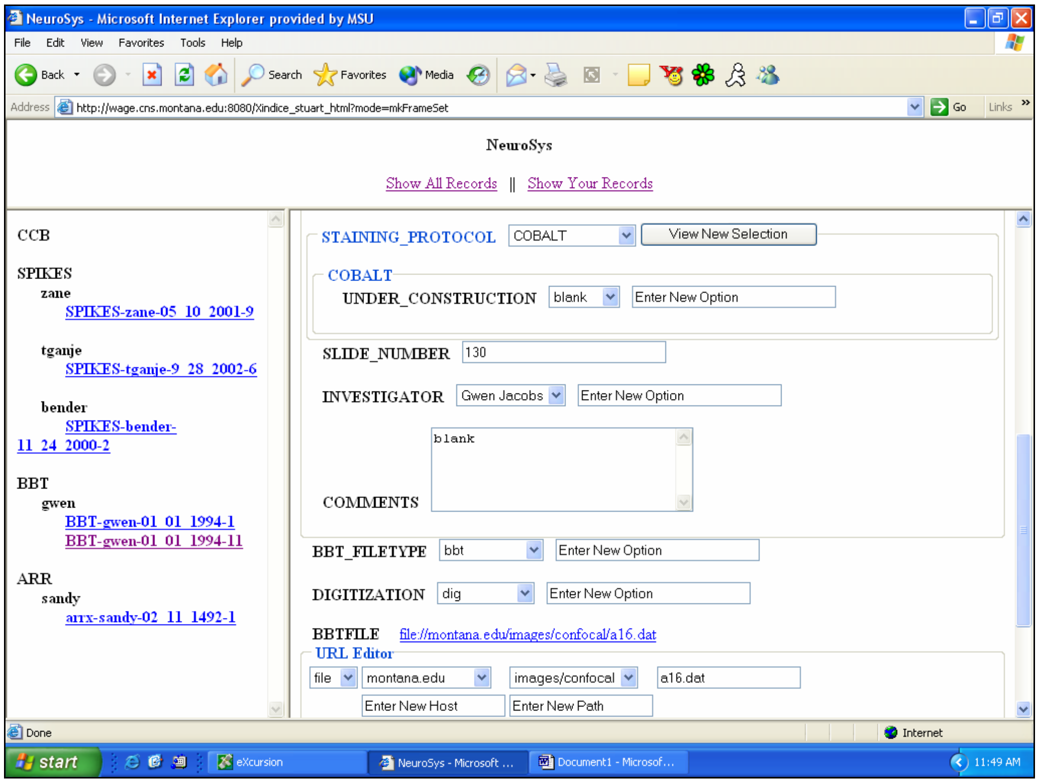
4.3 Browsing Records

Browsing a record is a simple matter of viewing a form in the right frame once a link has been selected in the left navigation panel. Information in the form is displayed in the general format: <*field name*> <*field value*>.

A user may access all records in the database by selecting the '*Show All Records*' link in the top frame. This is the default behavior when a user logs in. Selection of the '*Show Your Records*' link displays only the records that belong to the user that is currently logged in. The list of records in the left frame is hierarchically organized by the following headings in a top down fashion: *group*, *template*, *owner*, *document id*. The *group* heading designates the laboratory group that the owner of the record belongs to. The *template* heading groups records that have same schema. The last two headings are self explanatory.

Some fields in the form are simple text fields and text areas. Many of the fields are drop down menus with an arrow icon on the far right of the field area. The value displayed in the field is the current value.

The items in the drop down menu are not relevant for browsing the data except in the case of a 'ScreenSelect' field. A ScreenSelect field is identified by the presence of a drop down menu field enclosed within a border immediately to the right of a field name. The STAINING_PROTCOL field in Figure 16 is a ScreenSelect field. Selection of a different item in a ScreenSelect menu displays a new subform with format and data corresponding to the selected item. (Figures 17, 18). A ScreenSelect subform may be edited as any other portion of the form. Editing a form is described in the next section.


NeuroSys - Microsoft Internet Explorer provided by MSU
File Edit View Favorites Tools Help
Back Search Favorites Media
Address
Go Links
Done
Internet
start
eXcursion
NeuroSys - Microsoft ...
Document1 - Microsof...
11:49 AM

Address: http://wage.cns.montana.edu:8080/Xindice_stuart_html?mode=mkFrameSet

NeuroSys

Show All Records || Show Your Records

CCB

SPIKES
- zane
  - SPIKES-zane-05_10_2001-9
- tganje
  - SPIKES-tganje-9_28_2002-6
- bender
  - SPIKES-bender-11_24_2000-2

BBT
- gwen
  - BBT-gwen-01_01_1994-1
  - BBT-gwen-01_01_1994-11

ARR
- sandy
  - arrx-sandy-02_11_1492-1

STAINING_PROTOCOL: COBALT [View New Selection]

COBALT
- UNDER_CONSTRUCTION: blank | Enter New Option

SLIDE_NUMBER: 130

INVESTIGATOR: Gwen Jacobs | Enter New Option

COMMENTS: blank

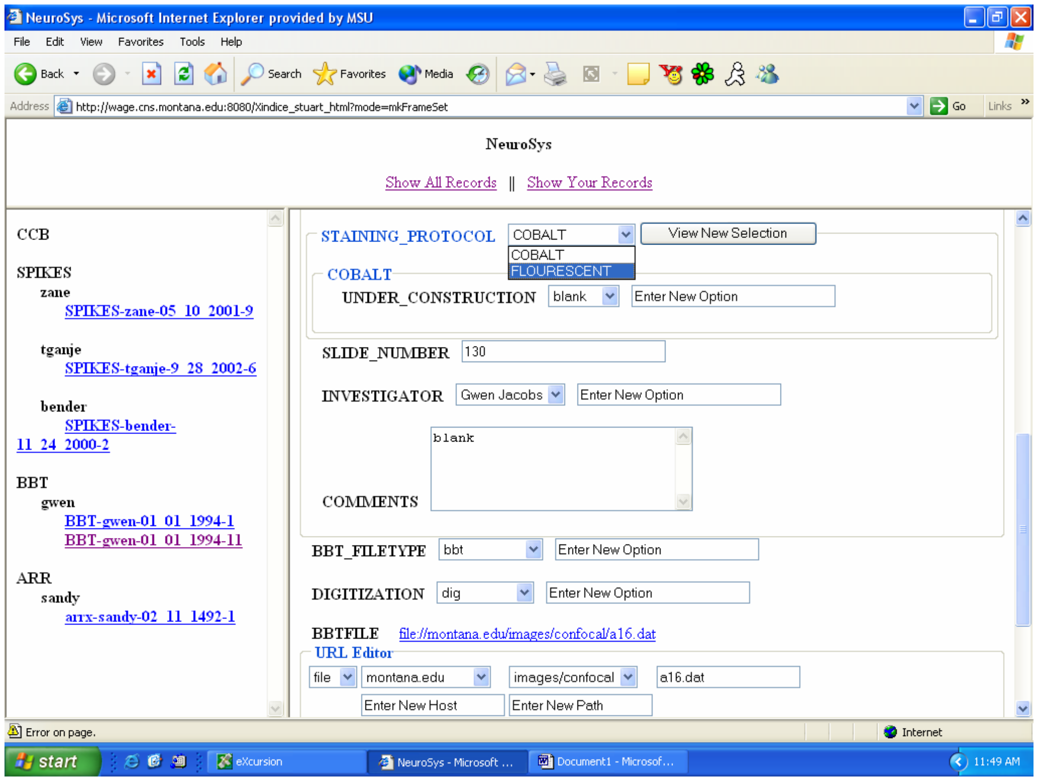
BBT_FILETYPE: bbt | Enter New Option

DIGITIZATION: dig | Enter New Option

BBTFILE: file://montana.edu/images/confocal/a16.dat

URL Editor
- file | montana.edu | images/confocal | a16.dat
- Enter New Host | Enter New Path

Figure 16 - ScreenSelect Field



http://wage.cns.montana.edu:8080/Xindice_stuart_html?mode=mkFrameSet

NeuroSys

Show All Records || Show Your Records

CCB

SPIKES
zane
SPIKES-zane-05_10_2001-9

tganje
SPIKES-tganje-9_28_2002-6

bender
SPIKES-bender-11_24_2000-2

BBT
gwen
BBT-gwen-01_01_1994-1
BBT-gwen-01_01_1994-11

ARR
sandy
arrx-sandy-02_11_1492-1

STAINING_PROTOCOL: COBALT (COBALT / FLOURESCENT)

COBALT
UNDER_CONSTRUCTION: blank — Enter New Option

SLIDE_NUMBER: 130

INVESTIGATOR: Gwen Jacobs — Enter New Option

COMMENTS: blank

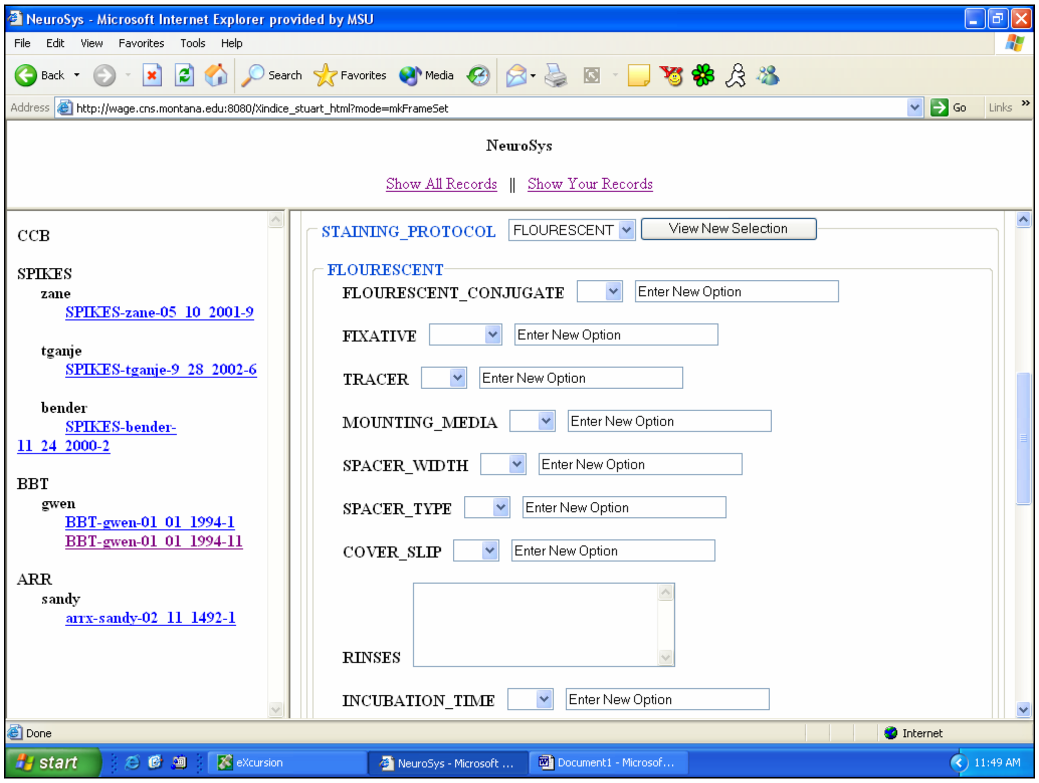
BBT_FILETYPE: bbt — Enter New Option

DIGITIZATION: dig — Enter New Option

BBTFILE: file://montana.edu/images/confocal/a16.dat

URL Editor
file — montana.edu — images/confocal — a16.dat
Enter New Host — Enter New Path

Figure 17 - Selecting a New Screen


NeuroSys - Microsoft Internet Explorer provided by MSU
File Edit View Favorites Tools Help
Back Search Favorites Media
Address
Go Links
View New Selection
Done
Internet
start
eXcursion
NeuroSys - Microsoft ...
Document1 - Microsof...
11:49 AM

http://wage.cns.montana.edu:8080/Xindice_stuart_html?mode=mkFrameSet

NeuroSys

Show All Records || Show Your Records

CCB

SPIKES
zane
SPIKES-zane-05_10_2001-9

tganje
SPIKES-tganje-9_28_2002-6

bender
SPIKES-bender-11_24_2000-2

BBT
gwen
BBT-gwen-01_01_1994-1
BBT-gwen-01_01_1994-11

ARR
sandy
arrx-sandy-02_11_1492-1

STAINING_PROTOCOL: FLOURESCENT

FLOURESCENT
FLOURESCENT_CONJUGATE — Enter New Option

FIXATIVE — Enter New Option

TRACER — Enter New Option

MOUNTING_MEDIA — Enter New Option

SPACER_WIDTH — Enter New Option

SPACER_TYPE — Enter New Option

COVER_SLIP — Enter New Option

RINSES

INCUBATION_TIME — Enter New Option

Figure 18 - Display of New Screen

4.4 Editing Records

A text field or text area may be edited by highlighting the displayed text and typing a new entry. The displayed text may also be edited by use of the blinking cursor that behaves like those seen commonly in text editors and word processors.

Drop down menu fields may be changed by selecting a new item in the drop down menu. Many of these fields have the option of entering a new item as the field value and adding it to subsequent displays of the menu. Such fields are distinguished by the presence of a text field immediately to the right of the drop down menu field. An entry in the text field takes precedence over a newly selected item from the drop down menu and the text field entry is the value that is written to the database using the save operations described in the next section.

4.5 Saving and Deleting Records

Any user may edit and save a copy of any record listed in the left navigation panel by pressing the '*Save As*' button at the bottom of the corresponding form in the right frame. The record is saved to the database with the user as the owner and with the record name following the convention:

<template name>_<user name>-<MM>_<DD>_<YY>-<sequence number>

A user that owns a record has the option of editing and overwriting the record by pressing the '*Save*' button at the bottom of the form. The record is saved to the database with its original name. A user may also delete one of his records by pressing the '*Delete*' button. The '*Save*' and '*Delete*' buttons are not displayed for records that do not belong to the current user.

This chapter consisted of instructions for the end user. The steps of logging in to a database, navigating the interface, browsing records, and editing and saving records were described and relevant screenshots were displayed. The next chapter discusses design specifications of the system and the reasons for design decisions that were made over the life of the project. The chapter is intended for future programmers on the project and for those interested in the technical aspects of the system.

5. Design Specifications for Programmers

This chapter discusses design specifications of the system and is intended as a manual for future programmers on the project.

5.1 System Layout

Implementation of this project required constructing a Java servlet that accepts HTTP requests from a client machine and generates HTML in response. The HTML servlet leverages the API of another class that communicates with the XML database servlet (Figure 19). The database servlet had already been constructed for use with the NeuroSys client-side Java Swing application.

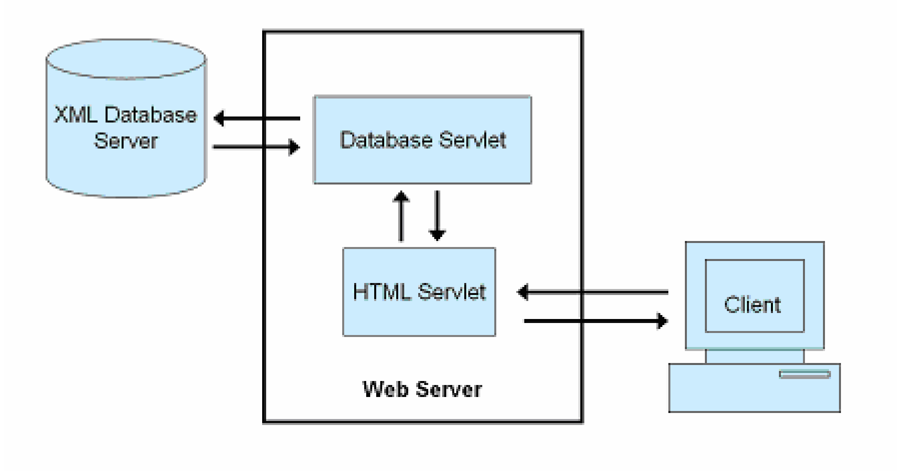


Figure 19 - System Layout

The next section describes the classes associated with the servlets, concentrating primarily on the HTML servlet.

5.2 Class Layout

5.2.1 The Database Servlet

The servlet class that handles low level database operations, `XindiceServer.java`, resides in package `xmlserver`. A programmer working on the web interface project must know how to install and configure this servlet on a web server, but need not be intimately familiar with the API of this class. An auxiliary class, `StringServletClient.java`, acts as a layer between `XindiceServer.java` and the HTML servlet described in the next section. `StringServletClient.java` facilitates database operations for the HTML servlet by providing an API for performing tasks such as connecting to the database server, pulling files, running queries, adding and saving files, and deleting files. The primary methods in this class are:

- `logon()` // connects to the database server
- `pullfile()` // retrieves a file
- `saveWithString ()` // saves (overwrite) a file
- `saveAsWithString()` // saves a file by a different name (copy)
- `deleteFile()` // deletes a file
- `runQuery()` // runs a query using an XPath string

A Unified Modeling Language (UML) diagram of the `StringServletClient` is shown in Figure 20. The class named `htmlHandler` appears in the upper right of the diagram and a dotted line with an open arrow tip points from `htmlHandler` to the `StringServletClient` class. `htmlHandler` is the HTML servlet and the arrow with the dotted line indicates that `htmlHandler` depends on `StringServletClient` by using an object of that class, which is consistent with the HTML servlet instantiating a `StringServletClient` object and utilizing its API. The `htmlHandler` class is discussed in more detail in the next section.

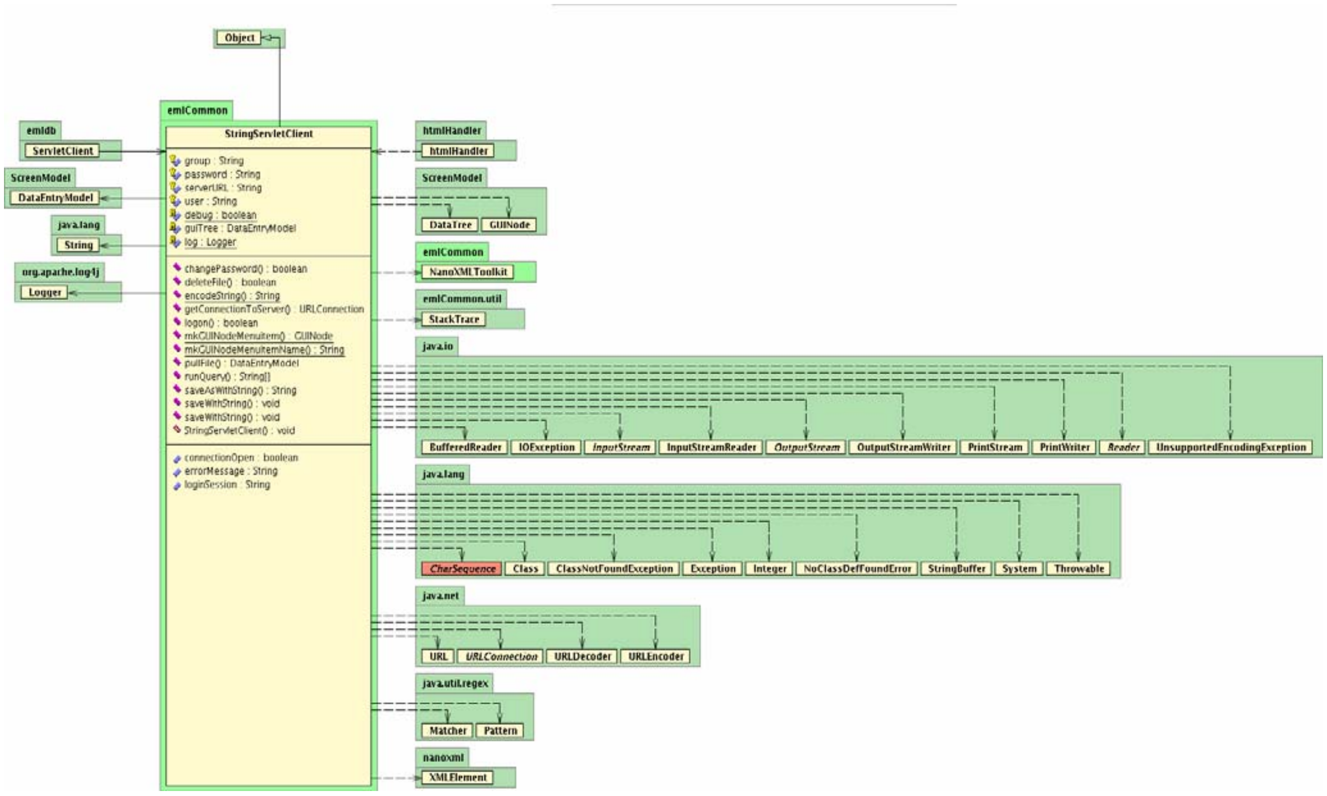


Figure 20 - StringServletClient UML Diagram

5.2.2 The HTML Servlet

The servlet class that processes client requests and enables dynamically generated responses is named `htmlHandler.java` and is the key class in the `htmlhandler` package. The `htmlHandler.java` `doPost()` method serves as the launching point for the servlet and dispatches calls to various methods depending on parameters delivered in an `HttpServletRequest` object. Pseudocode for the `doPost()` method is shown below.

```
doPost()
{
  if mode is null
            generate login page
  else if mode is mkFrameSet
            generate the 3-frame interface layout
  else if mode is defaultLeft
            generate the left navigation frame
  else if mode is defaultTop
            generate the top header frame
  else if mode is messagePage
            generate a generic HTML page that displays a given string
  else if mode is save
            pull the file from the database
            compare each element with HTTP POST parameters
            update the file with any changes
            display the new file
            save the new file to the database
  else if mode is save_as
            same as “save”
  else if mode is delete
            delete the file
  else if mode is screenSelect
            pull the file from the database
            compare each element with HTTP POST parameters
            update the file with any changes
            display the new file
  else if mode is pullFile
            pull the file from the database
            display the file
}
```

The first several cases in the sequence of if-else statements in `doPost()` handle requests to construct HTML frames that comprise the database interface. The method `login()` builds the login page shown in Figure 13. A successful login attempt prompts a call to `mkFrameSet()` which defines the frame layout shown in Figure 14. The `mkFrameSet()` method in turn calls the methods `defaultTop()`, `defaultLeft()`, and `messagePage()` to populate the top, left, and right frames respectively with HTML content (Figure 21).

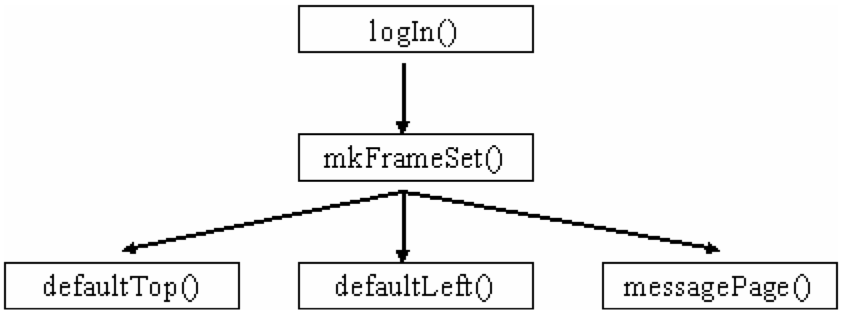


Figure 21 - Method Call Sequence in doPost()

When a user clicks a hyperlinked record name in the left frame, *‘mode=pullFile’* is sent to `doPost()` as an HTTP request parameter and the corresponding file is pulled from the database as a `DataEntryModel` object which is sent as a parameter to the method `makeDisplay()`. The `makeDisplay()` method begins a recursive process that displays the record as a form in the right frame (Figure 22).

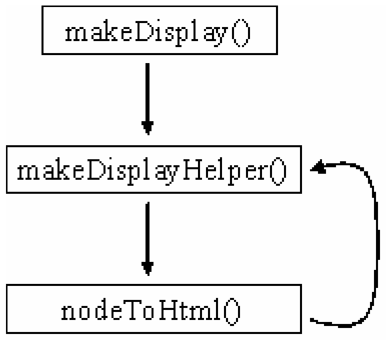


Figure 22 - Method Call Sequence for makeDisplay()

The `makeDisplay()` method generates the required opening and closing tags for an HTML form page. Calls to the auxiliary methods `makeDisplayHelper()` and `nodeToHtml()` recurse through the `GUINode` objects that make up the `DataEntryModel` tree and dispatch the `GUINode`s to methods that generate HTML for a particular type of 'widget' (i.e. text field, text area, drop down menu, etc.). The remaining cases in the `doPost()` if-else statements handle operations such as saving or deleting a record and redisplaying it if changes have been made.

If a record is altered, the `fixWholeTree()` method loops through the `DataEntryModel` object that represents the original record and compares the value of each node in that data structure with the corresponding HTTP POST parameter. If the two values differ, the value of the node is updated with the new parameter.

5.2.3 Auxiliary Classes

The `htmlHandler` class utilizes some custom classes that perform certain tasks. The 'bucket' classes `GroupBucket, TemplateBucket, OwnerBucket, and DescriptionBucket` serve as nested containers that ultimately contain `Descripton` objects that hold basic information about a record. These classes assist in grouping the records in their appropriate categories.

The method `htmlHandler.defaultLeft()` receives an array of records from the database, creates a `Descripton` object for each record, and then puts the record description into its appropriate `DescriptionBucket`. The `DescriptionBucket` is put into the appropriate `OwnerBucket`, which is put into the appropriate `TemplateBucket`, which is finally put into the appropriate `GroupBucket`. After all the records are processed, a hashtable of `GroupBucket` objects is sent to the `defaultLeftDisplay()` method which is responsible for unwinding the nested buckets and displaying the record names in their respective groupings.

The `HtmlGenerator` class contains static methods that generate HTML components such as text fields, text areas, checkboxes, pull down menus, and hyperlinks. To aid in generating valid HTML there are methods that produce elements with DOCTYPE and version information. The UML diagram for `HtmlGenerator` is shown in Figure 23.

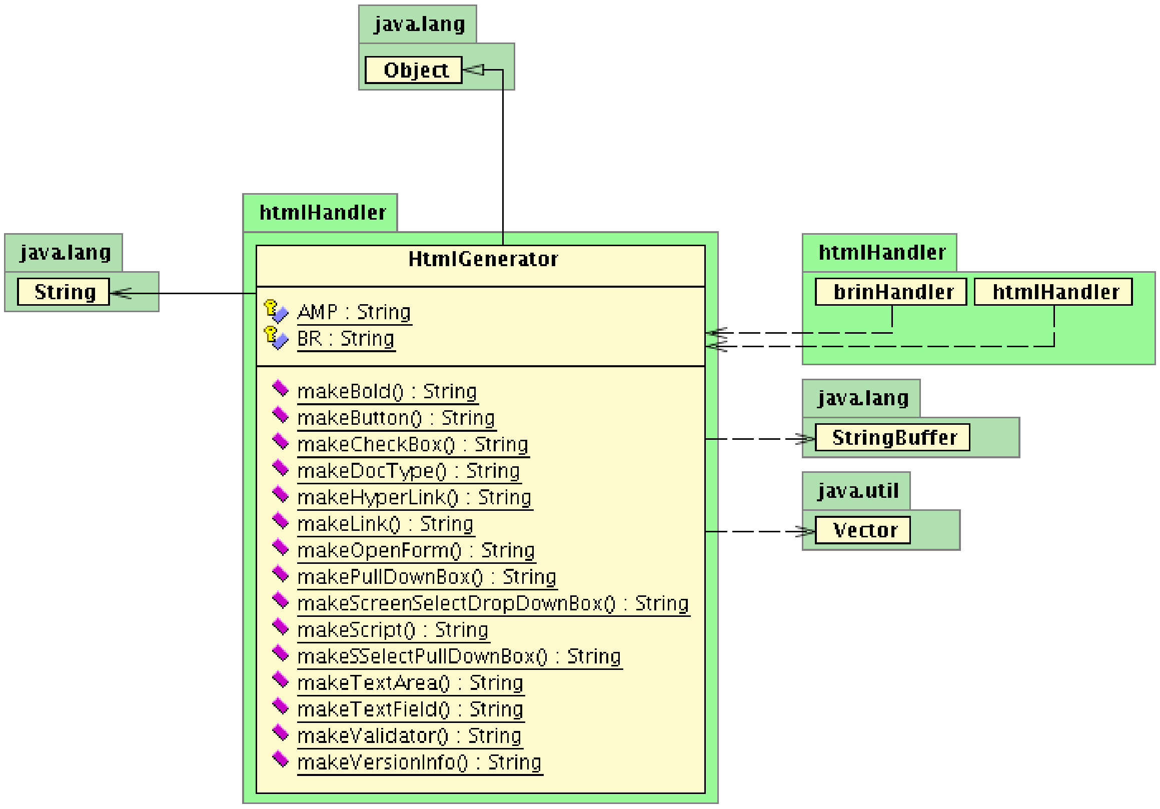


Figure 23 - HtmlGenerator UML Diagram

5.3 Design Issues

This project initially defined four servlets to generate the web interface. Each frame was generated by a separate servlet and these servlets were called by `htmHandler.java`. To increase the simplicity and to reduce the amount of code, it was decided to merge all the functionality into the `htmHandler` servlet. Previous calls to multiple servlets are now replaced by method calls within `htmHandler.java`. As a result only one servlet class must be compiled and loaded into the web server.

Two methods in `htmHandler.java` are defined for debugging purposes. The `showDebuggingInfo()` method displays information on HTTP POST parameters and session variables. Calls to this method are currently placed in the methods `makeDisplay()` and `defaultLeft()`. The calls are made if a servlet initiation parameter named '*debug*' is set to '*true*.' This parameter is set in the servlet's *web.xml* file.

Another useful debugging method is `debugTree()` which takes a `DataEntryModel` object as a parameter and returns a string that contains all attributes and values for every node in the tree. This string may be sent to the `messagePage()` method which generates a generic HTML page that displays the string.

This chapter discussed design specifications of the system. Chapter Six compares and contrasts the Neurosys project to the database systems introduced in Chapter Two.

6. Results and Comparisons

This chapter is dedicated to comparing the web interface for NeuroSys with the web interfaces for similar scientific database systems. The first section lists the evaluation criteria. The subsequent sections list results of evaluating the three similar systems introduced in section 2.2 and the NeuroSys project with these criteria. The last section summarizes the evaluation results in a table.

6.1 Comparative Criteria

The author of this paper has selected eight criteria for evaluating web interfaces for database systems. The criteria are selected primarily from usability and user interface design guidelines cited in Chapter 2 [6, 25]. The author has added two criteria concerning help documentation and valid HTML. The criteria are summarized below.

- *Visual Design* – The system should be aesthetically appealing and have a minimalist design.
- *Learnability* – The system should be easy to learn.
- *Efficiency* – The system should allow users to perform their tasks productively (i.e. provide shortcuts for experienced users, avoid redundant actions).
- *Errors* – The system should have a low error rate and good error handling. Error messages should be concise, understandable, and should suggest a solution.
- *Consistency* – The system should be consistent in all aspects (i.e. layout, colors, text format, terminology, action sequences, etc.).
- *User Control* – The system should the allow user to modify the data entry and data display format.
- *Help and Documentation* – The system should have Question and Answer and/or FAQ pages.

- *Valid HTML* – The system should produce web pages that conform to the HTML 4.01 standard [10] and that can be verified by the World Wide Web Consortium (W3C) validator [28].

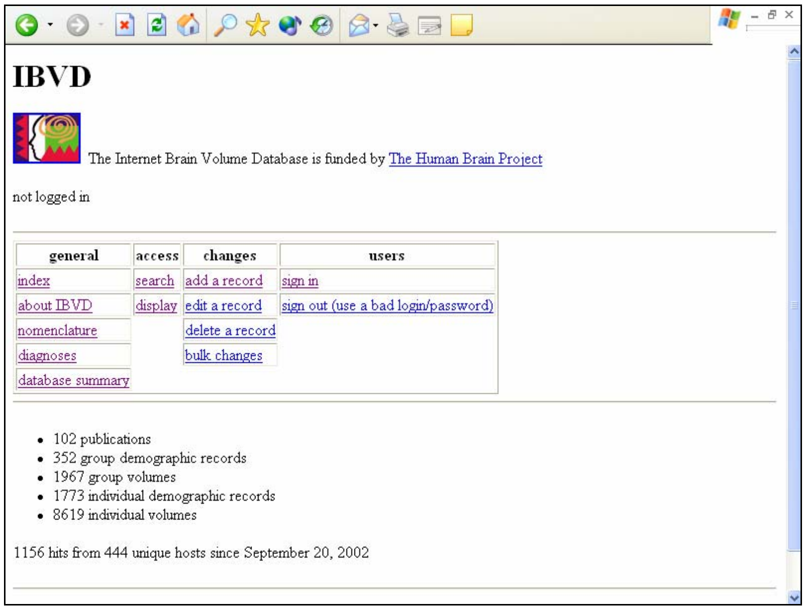
Each database system reviewed will be scored on each criterion with the following scoring scale:

1 – poor 2 – fair 3 – average 4 – good 5 – excellent

6.2 Internet Brain Volume Database (IBVD)

Visual Design

At first glance, the Internet Brain Volume Database homepage [13] is simple and uncluttered (Figure 24).



IBVD

The Internet Brain Volume Database is funded by The Human Brain Project

not logged in

general	access	changes	users
index	search	add a record	sign in
about IBVD	display	edit a record	sign out (use a bad login/password)
nomenclature		delete a record	
diagnoses		bulk changes	
database summary			

- 102 publications
- 352 group demographic records
- 1967 group volumes
- 1773 individual demographic records
- 8619 individual volumes

1156 hits from 444 unique hosts since September 20, 2002

Figure 24 - IBVD Homepage

That said, the right one third of the screen area is whitespace which could be better utilized or redistributed to balance the scarcity of whitespace on the left side of the screen. The heading, IBVD, is prominent but not centered in the page and it is not clear that it is an acronym for Internet Brain Volume Database. The user most likely can piece together what the heading means when he sees the expanded name down the page and to the right of the colored logo. The user may accidentally discover that the colored logo is a hyperlink. The target of the hyperlink is unclear. One might suspect that the link leads to an important section of the database since it is adjacent to the expanded name of the database. Experimentation shows that the logo is linked to the homepage of the project supporting the development of the database instead. To the far right of the expanded database name is hyperlinked text that more explicitly links to the homepage of the organization funding the project. Two links to the same location introduce redundancy and may confuse the user.

Below the logo appears the statement "*not logged in*." This appears to be a confirmation of the user's login status, but there is not a label to confirm this. The prominent placement of this statement is puzzling.

A table midway down the page serves as the launching point for the website and database operations. The table contains hyperlinked text that initiates each operation. The meanings of the table headings and hyperlinks are for the most part self-explanatory except for the '*bulk changes*' hyperlink under the '*changes*' heading. The parenthesized portion of the '*sign out (use a bad login/password)*' link under the '*users*' heading is confusing. All headings and hyperlinks in the table start with lowercase letters which give the page the look of an unfinished prototype.

A bulleted list of database statistics appears below the table and at the bottom of the page the statement "*no errors reported*" appears. This statement does not specify what types of errors have not been reported and may make the user hesitate to explore the system for fear of making errors.

The author considered giving a score of ***good*** for visual design because the IBVD index page is simple and uncluttered. Because of the issues with balance of whitespace, lowercase table entries, and confusing hyperlinks and status messages the score for visual design is ***average***.

Learnability

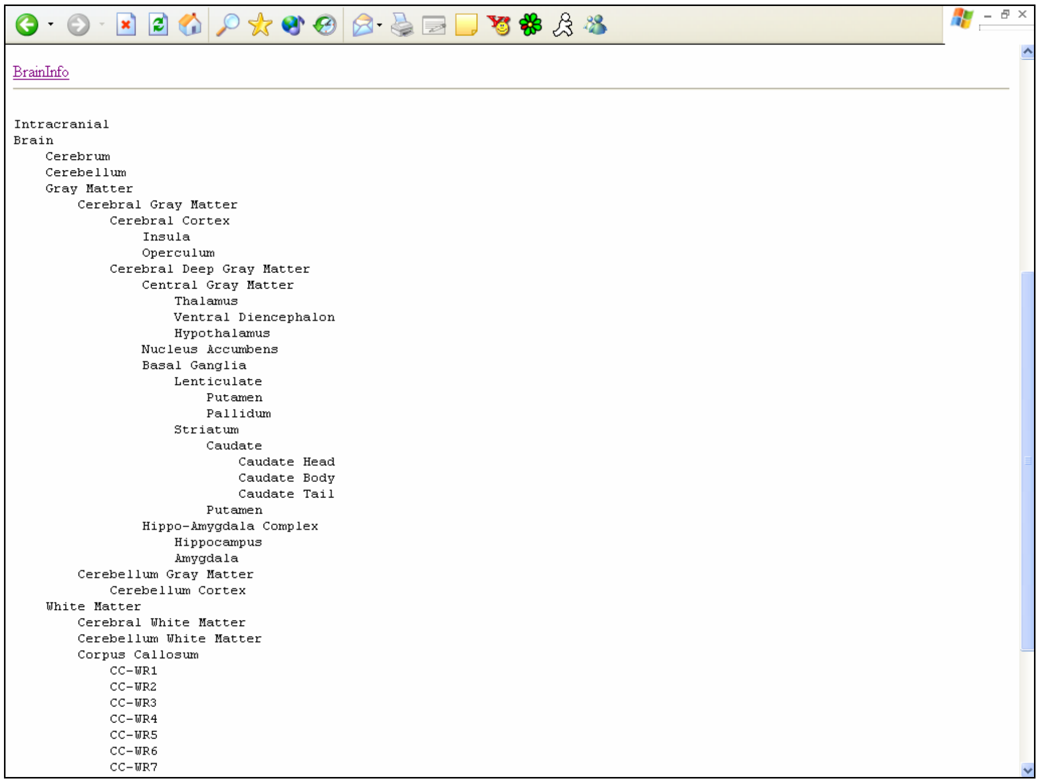
The simplicity of the index page and a walk-through tutorial of the system contribute to making this system fairly easy to learn. The location of the tutorial is not obvious however and it is found a couple of layers below the '*about IBVD*' link. This system gets a score of ***good*** for learnability.

Efficiency

Several operations in the database require navigating through multiple pages or links where better design would allow one user action to perform the same task. One example mentioned in the previous section is locating the database tutorial. A direct link from the index page labeled '*Tutorial*' would suffice.

Another operation requiring multiple user actions is following the '*nomenclature*' link on the index page. This link leads to a page containing a hierarchical listing of brain-related terminology, but no apparent way to find the definition of a term (Figure 25). Further browsing reveals that the '*BrainInfo*' link in the upper left corner of the display leads to an external website [4] that allows a user to search for information on brain structures by name. A search for the definition of the term *Operculum* requires the user to:

1) highlight and copy the term text
2) click on the '*BrainInfo*' link to reach the external website
3) press the '*Search By Name*' button
4) paste the term text into the dialog box
5) press the '*Submit Query*' button
6) select a link from the search matches



```
Intracranial
Brain
    Cerebrum
    Cerebellum
    Gray Matter
        Cerebral Gray Matter
            Cerebral Cortex
                Insula
                Operculum
            Cerebral Deep Gray Matter
                Central Gray Matter
                    Thalamus
                    Ventral Diencephalon
                    Hypothalamus
                Nucleus Accumbens
                Basal Ganglia
                    Lenticulate
                        Putamen
                        Pallidum
                    Striatum
                        Caudate
                            Caudate Head
                            Caudate Body
                            Caudate Tail
                        Putamen
                Hippo-Amygdala Complex
                    Hippocampus
                    Amygdala
        Cerebellum Gray Matter
            Cerebellum Cortex
    White Matter
        Cerebral White Matter
        Cerebellum White Matter
        Corpus Callosum
            CC-WR1
            CC-WR2
            CC-WR3
            CC-WR4
            CC-WR5
            CC-WR6
            CC-WR7
```

Figure 25 - Nomenclature Page

Though the information provided about a term on the external website is very complete, a hyperlinked term that led to a definition of the term would suffice and would relieve the user of five extra actions.

On the plus side for efficiency is the presence of the table of database operations at the top of every page in the website. A user is not required to backtrack if he suddenly wishes to perform another operation.

This system receives a score of ***average*** for efficiency.

Errors

As mentioned in the section on visual design, the status messages "*not logged in*" and "*no errors reported*" appear on the index page. Although the messages are concise, their meaning and origin are not completely clear.

The author was able to elicit an error message by clicking on the restricted action '*add a record*' under the '*changes*' heading. If the user presses the '*Cancel*' button on the password dialog that appears, the following message appears:

```
You must log in with a valid usename and password to add to the
database.
```

The message is clear and suggests a solution, though 'username' is misspelled. A similar sequence of actions on the '*sign in*' link produced the following response:

```
sign in (id is -1)

array(7) {
  ["Accept"]=>
  string(164) "image/gif, image/x-xbitmap, image/jpeg, image/pjpeg,
application/vnd.ms-excel, application/vnd.ms-powerpoint,
application/msword, application/x-shockwave-flash, */*"
  ["Referer"]=>
  string(45) "http://www.cma.mgh.harvard.edu/ibvd/index.php"
  ["Accept-Language"]=>
  string(5) "en-us"
  ["Accept-Encoding"]=>
  string(13) "gzip, deflate"
  ["User-Agent"]=>
  string(69) "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; .NET
CLR 1.0.3705)"
  ["Host"]=>
  string(23) "www.cma.mgh.harvard.edu"
  ["Authorization"]=>
  string(10) "Basic Og=="
}
```

The message is unclear, there is no suggested solution, and the display of some code appears to be an oversight of the programmer. Perhaps it is debugging output that was not meant to be displayed to the user.

This system receives a score of ***fair*** for errors and error handling.

Consistency

The appearance of the database operations table in practically every page in the website along with the same logo and background offer a sense of consistency. The lowercase table and field labels, though not desirable, are carried throughout the website and used consistently in all other table and field labels.

This system receives a score of ***excellent*** for consistency.

User control

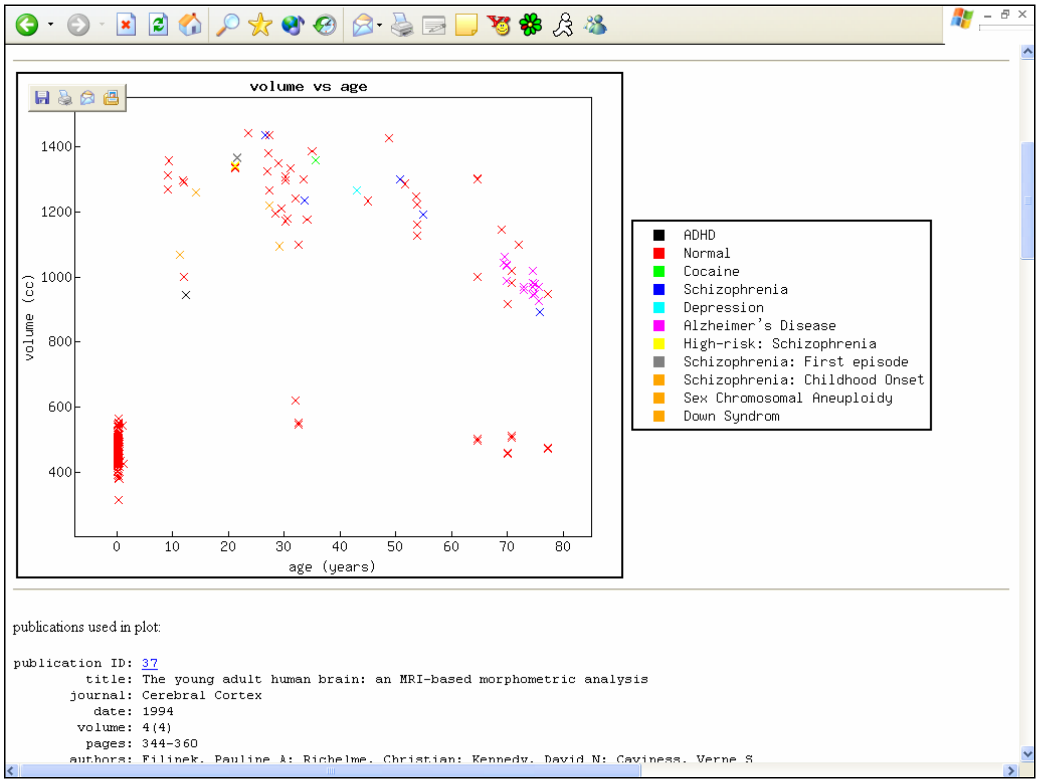
It is not possible for an unauthorized user to determine if the data entry format may be modified. Data from a query is available however and user control over the display of this data will serve as the basis for the score.

A particular query displays search results in an HTML table by default. The user has the option to display the data in two other ways. Pressing the '*flat text table*' button displays the search results in the form of a text file. The fields are separated by tabs by default, but the user has the option of entering a one character custom field separator. The user also has the option of plotting the search results in a graph by pressing the '*plot volume*' button (Figure 26). This appears to be a very useful and powerful feature.

This system receives a score of ***excellent*** for user control.

Help and Documentation

As mentioned previously, the location of the system tutorial is buried two levels below the index page and the '*nomenclature*' link is not intuitive to use. The system does make an attempt to provide help and documentation pages so it receives a score of ***average*** for this category.



publications used in plot:

```
publication ID: 37
         title: The young adult human brain: an MRI-based morphometric analysis
       journal: Cerebral Cortex
          date: 1994
        volume: 4(4)
         pages: 344-360
       authors: Filipek, Pauline A; Richelme, Christian; Kennedy, David N; Caviness, Verne S
```

Figure 26 - Plot Display

HTML Validation

Entry of the IBVD homepage URL [13] into the W3C HTML validator produced the following message in bright red font:

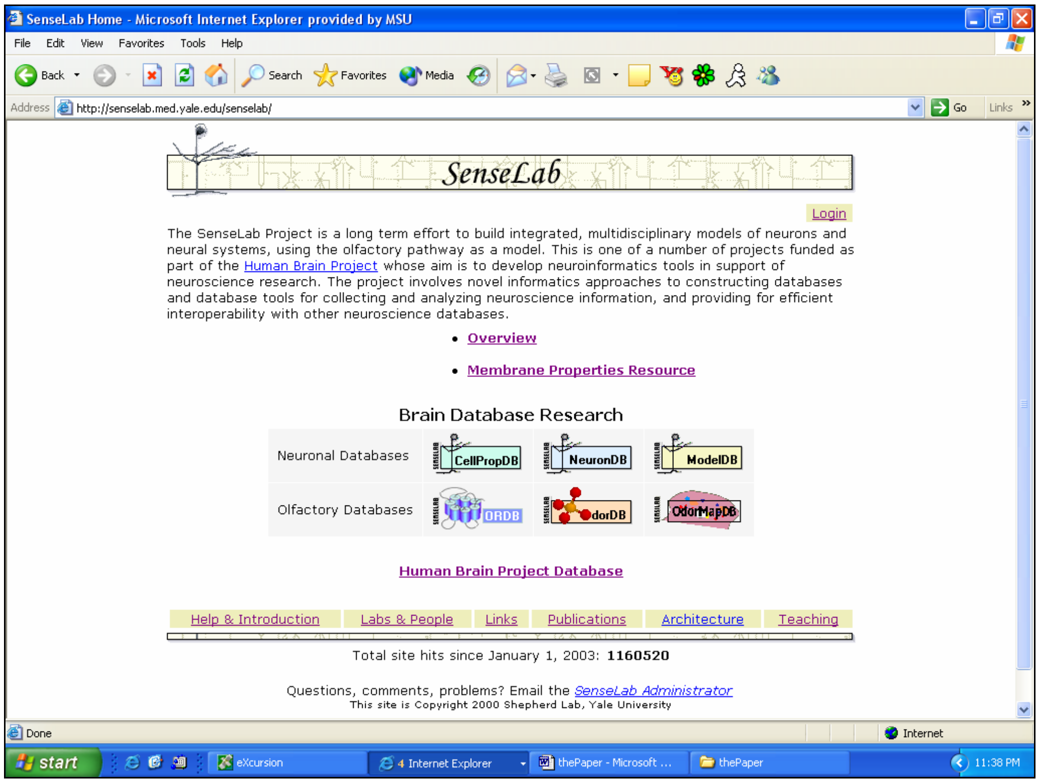
Fatal Error: No DOCTYPE specified!

This system receives a score of ***poor*** for valid HTML.

6.3 SenseLab

Visual Design

The SenseLab homepage [24] gives a favorable first impression (Figure 27). The graphics and logos are pleasing in both style and color, the information is centered on the page giving a sense of balance with respect to whitespace, and the complete page is visible in the browser so no scrolling is required to view all of it.



The SenseLab Project is a long term effort to build integrated, multidisciplinary models of neurons and neural systems, using the olfactory pathway as a model. This is one of a number of projects funded as part of the Human Brain Project whose aim is to develop neuroinformatics tools in support of neuroscience research. The project involves novel informatics approaches to constructing databases and database tools for collecting and analyzing neuroscience information, and providing for efficient interoperability with other neuroscience databases.

- Overview
- Membrane Properties Resource

Brain Database Research

Neuronal Databases

Olfactory Databases

Human Brain Project Database

Help & Introduction | Labs & People | Links | Publications | Architecture | Teaching

Total site hits since January 1, 2003: **1160520**

Questions, comments, problems? Email the *SenseLab Administrator*

This site is Copyright 2000 Shepherd Lab, Yale University

Figure 27 - SenseLab Homepage

Links to all of the databases in the project are displayed in a table in the center of the screen. The links are well labeled and are distinguished from each other by either color or a particular logo. The lower portion of the page has a menu bar that provides links to other pages in the site. One improvement the site administrator could make is to place the three hyperlinks that are ‘floating’ in the space above and below the database table into the menu bar at the bottom. They could also be placed '*Links*' portion of the website. The result would be slightly cleaner and simpler page.

The system receives a score of ***excellent*** for visual design.

Learnability

The '*Help and Introduction*' link in the bottom menu bar does lead to a page labeled 'Project Help and Introduction,' but the page consists of four bulleted sentences containing little information and irrelevant links. The one sentence that does not contain any links is the one that could most use one. It appears as follows:

- For information about the different databases in the SenseLab Project see the FAQ links in these projects

A link leading a user to the FAQ page of each project would be helpful. An inspection of each of the six databases in the project reveals that only three of them have FAQ or help pages. All six databases display a navigation panel on the left side of each page in the particular site. Pressing the '*Resources*' button in this panel in each database results in the message, "This page is under construction."

For such a complex database system, the help documentation is not easy to find and is incomplete. A walk-through tutorial of the system, either text-based or web-based, would be helpful for learning to use the system.

This system receives a score of ***fair*** for learnability.

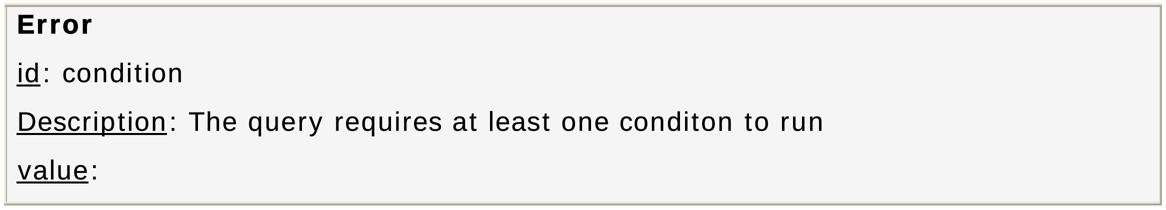
Efficiency

As mentioned in Chapter 2, relevant attributes of a particular cell in the Cellular Properties Database (CellPropDB) [7] are hyperlinked for viewing a list of other cells with the same attribute. The '*Ref*' link associated with each attribute allows the user to view a description of the cell property and to view links to related journal references. Links at the top of the form lead to related data in the Neuronal Database [21]. The abundance and orchestration of hyperlinks in the data displays allow the experienced user to quickly find the data he is searching for. A navigation panel on the left side of each page allows the user to easily perform another action, with the exception of the '*Resources*' link which is not operable. All of these features allow a user to use the system productively.

This system receives a score of ***excellent*** for efficiency.

Errors

The author was able to elicit the following error message by performing a search with no entry in the '*Condition*' field:



```
Error
id: condition
Description: The query requires at least one conditon to run
value:
```

The message identifies the field that was empty and suggests a solution, although the word 'condition' is misspelled. An attempt to log in to the system with a blank user name and password produced the following message:

```
Your web agent either does not support cookies or have them disabled.
Cookies are required to log in.
```

Again, the message is concise and suggests a solution. This system receives a score of ***excellent*** for error handling.

Consistency

Each of the six databases in the system offer consistency throughout their respective database by using the same logos, color schemes, and page layouts. The different color scheme for each database is especially helpful for a user to track his current location in the system. One deviation from the uniform color scheme seems to occur in the Neuronal Database (NeuronDB) [21] where the user has the choice to view various properties of a particular cell. The property names are hyperlinked and each link displays information about the corresponding property. The background color of the data display is different for each property. Viewing the same property for a different cell reveals that the background color remains consistent for each property.

This system receives a score of ***excellent*** for consistency.

User Control

This system allows the user a great degree of control and flexibility with respect to data display and searching. The user has the option of displaying data fields in alphabetical or logical order. Likewise the complex search option as shown in Figure 7 in Chapter 2 allows the search properties to be listed logically or alphabetically. The standard search feature allows four options for displaying the results: HTML, XML Text, XML Formatted, and Text. Another useful feature is the hyperlink labeled '[*Print*]' that appears in the upper right corner of each page that displays data. Pressing this link redisplays the data as a full screen view without the navigation panel that was previously on the left side of the page.

This system receives a score of ***excellent*** for user control.

Help and Documentation

As mentioned in the section on *Learnability*, help pages are either incomplete or are not easy to find.

This system receives a ***fair*** for help and documentation.

Valid HTML

A first attempt to validate the HTML of the SenseLab homepage [24] failed with four errors reported. The author saved a copy of the source code and commented out two '*<br/>*' elements which responsible for two of the complaints. The next attempt to validate the page produced the following message:

This Page Is Valid XHTML 1.0 Transitional!

The designers of the website obviously attempted to conform to XHTML 1.0 standards [31].

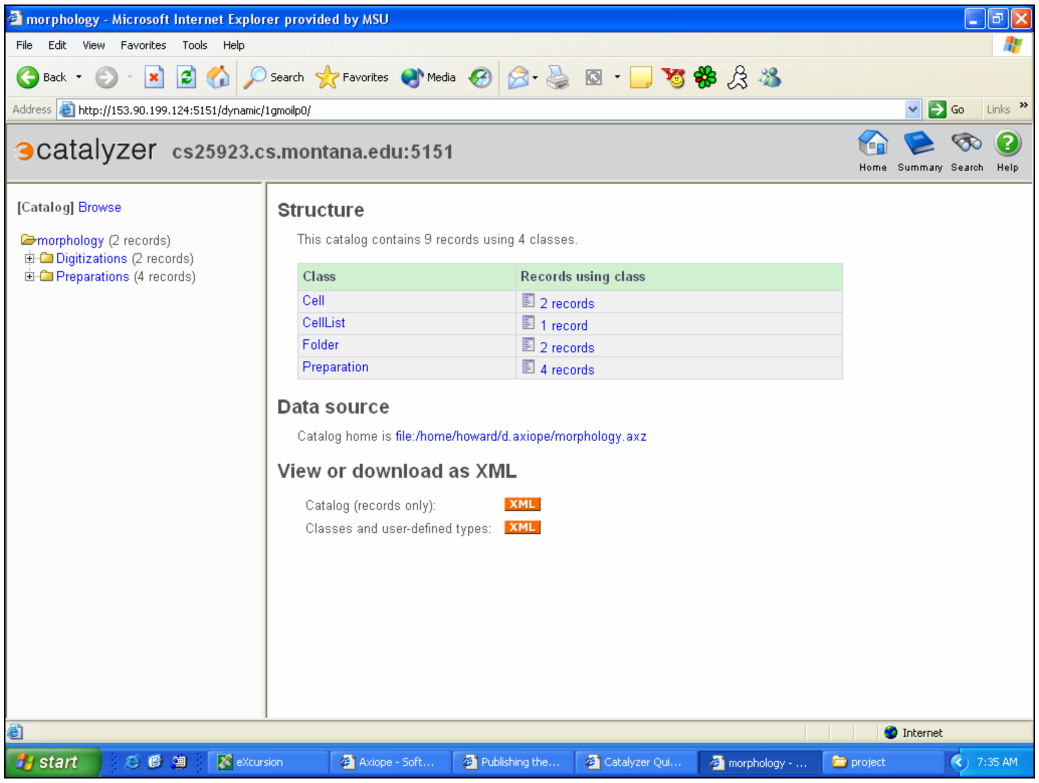
This system receives a ***good*** for valid HTML.

6.4 Axiope Catalyzer

Visual Design

The homepage of a catalog website consists of three frames (Figure 28). The top frame displays the URL of the website in bold font and provides icons that allow a user to jump to other areas of the website or to perform a search. The left frame is a navigation panel and lists the contents of a catalog in a hierarchical fashion. A user may expand and collapse the file tree as is done commonly in a Microsoft Windows file manager. All headings in the file tree are hyperlinked and lead to the corresponding level in the database. The right frame serves as a data display area. This frame by default contains a table that summarizes the contents of the catalog. The design of the homepage is clean and simple and by excellent use of hyperlinks and icons, it offers a great amount of functionality without being cluttered. The color scheme and logo combine to give a visually appealing web page.

This system receives a score of ***excellent*** for visual design.


morphology - Microsoft Internet Explorer provided by MSU
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://153.90.199.124:5151/dynamic/1gmoilp0/
Go Links
catalyzer cs25923.cs.montana.edu:5151
Home Summary Search Help
Internet
start eXcursion Axiope - Soft... Publishing the... Catalyzer Qui... morphology - ... project 7:35 AM

[Catalog] Browse

morphology (2 records)
Digitizations (2 records)
Preparations (4 records)

Structure

This catalog contains 9 records using 4 classes.

Class	Records using class
Cell	2 records
CellList	1 record
Folder	2 records
Preparation	4 records

Data source

Catalog home is file:/home/howard/d.axiope/morphology.axz

View or download as XML

Catalog (records only): XML
Classes and user-defined types: XML

Figure 28 - Axiope Catalyzer Catalog Homepage

Learnability

The top frame contains a '*Help*' icon that is always available as a user navigates through the website. The icon leads to a simple page that highlights the features of the web server. The clean design of the website pages and the easily accessible help page make this system intuitive and easy to use.

This system receives a score of ***excellent*** for learnability.

Efficiency

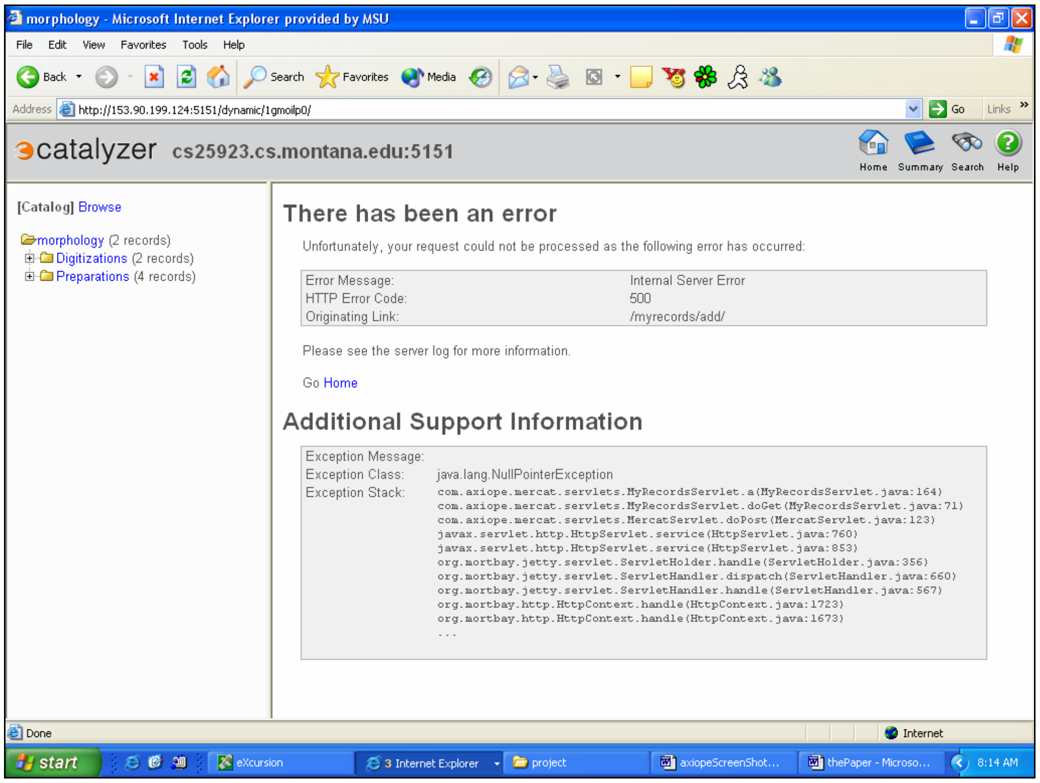
As mentioned in the section on visual design, efficient use of hyperlinks and icons provides the user with a great deal of functionality in every page of the website. A single click of the mouse initiates the desired action.

This system receives a score of ***excellent*** for efficiency.

Errors

At a point in the website a user may add records of interest to a ‘shopping basket’. The records can be downloaded as XML files which may then be saved as a catalog by the client-side application. If no records are selected and the user presses the ‘*Add To My Records*’ button, the error message shown in Figure 29 appears. The heading concisely states that an error has occurred but does not clearly specify the type of error. The remainder of the information is not understandable to the average user and seems appropriate only for the programmers that implemented the system. The message does not suggest a clear solution for avoiding the error. This error message is typical of other error messages generated by the system, although the ‘*Error Message*’ field in other messages gave more specific information about the type of error.

This system receives a score of ***fair*** for errors and error handling.


morphology - Microsoft Internet Explorer provided by MSU
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://153.90.199.124:5151/dynamic/1gmoilp0/
Go Links
catalyzer cs25923.cs.montana.edu:5151
Home Summary Search Help
[Catalog] Browse
morphology (2 records)
Digitizations (2 records)
Preparations (4 records)
Done
Internet
start
eXcursion
3 Internet Explorer
project
axiopeScreenShot...
thePaper - Microso...
8:14 AM

There has been an error

Unfortunately, your request could not be processed as the following error has occurred:

Error Message:	Internal Server Error
HTTP Error Code:	500
Originating Link:	/myrecords/add/

Please see the server log for more information.

Go Home

Additional Support Information

```
Exception Message:
Exception Class:     java.lang.NullPointerException
Exception Stack:     com.axiope.mercat.servlets.MyRecordsServlet.a(MyRecordsServlet.java:164)
                     com.axiope.mercat.servlets.MyRecordsServlet.doGet(MyRecordsServlet.java:71)
                     com.axiope.mercat.servlets.MercatServlet.doPost(MercatServlet.java:123)
                     javax.servlet.http.HttpServlet.service(HttpServlet.java:760)
                     javax.servlet.http.HttpServlet.service(HttpServlet.java:853)
                     org.mortbay.jetty.servlet.ServletHolder.handle(ServletHolder.java:356)
                     org.mortbay.jetty.servlet.ServletHandler.dispatch(ServletHandler.java:660)
                     org.mortbay.jetty.servlet.ServletHandler.handle(ServletHandler.java:567)
                     org.mortbay.http.HttpContext.handle(HttpContext.java:1723)
                     org.mortbay.http.HttpContext.handle(HttpContext.java:1673)
                     ...
```

Figure 29 - Axiope Catalyzer Error Message

Consistency

The layout of all pages in the system is consistent with the frame layout of the site homepage. The frame proportions are consistent from page to page as is the color scheme of each frame. Table headings and fields remain consistent from page to page.

This system receives a score of ***excellent*** for consistency.

User Control

The Axiope Catalyzer web interface offers users a great deal of control in displaying the data from a record or a class. Field values from a particular record may be plotted as bar graphs, line graphs, or pie chart graphs (Figure 30). When viewing the contents of a particular class (schema), the user has the option of an '[*expanded view*]' or a '[*compact* *view*]' which provide larger or smaller field areas respectively. The expanded view is helpful for viewing records that contain images.

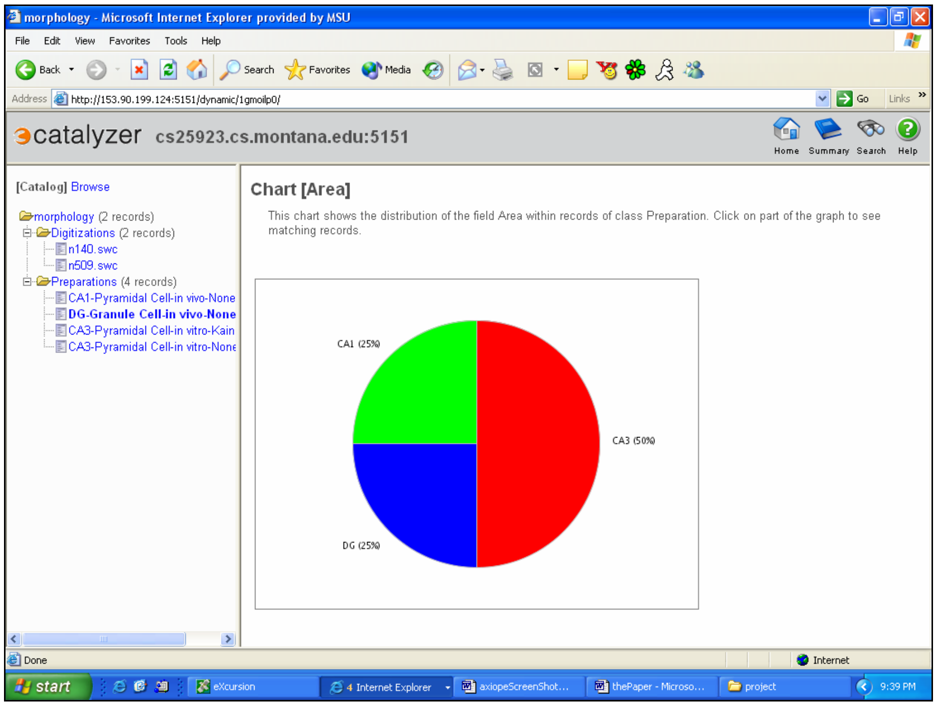
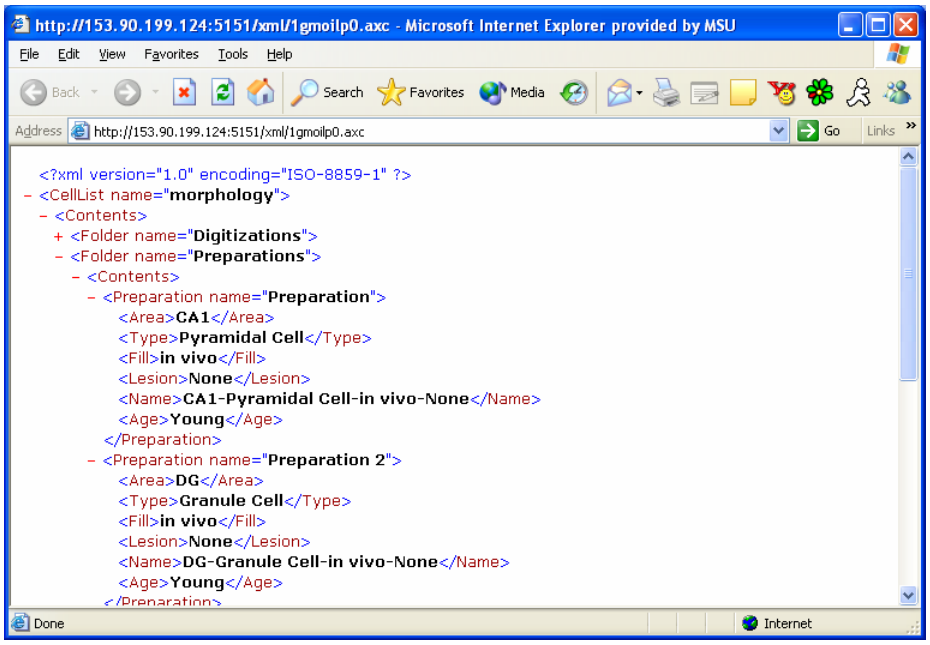

morphology - Microsoft Internet Explorer provided by MSU
http://153.90.199.124:5151/dynamic/1gmoilp0/
catalyzer cs25923.cs.montana.edu:5151
Home
Summary
Search
Help
[Catalog] Browse
morphology (2 records)
Digitizations (2 records)
n140.swc
n509.swc
Preparations (4 records)
CA1-Pyramidal Cell-in vivo-None
DG-Granule Cell-in vivo-None
CA3-Pyramidal Cell-in vitro-Kain
CA3-Pyramidal Cell-in vitro-None
CA1 (25%)
CA3 (50%)
DG (25%)

Chart [Area]

This chart shows the distribution of the field Area within records of class Preparation. Click on part of the graph to see matching records.

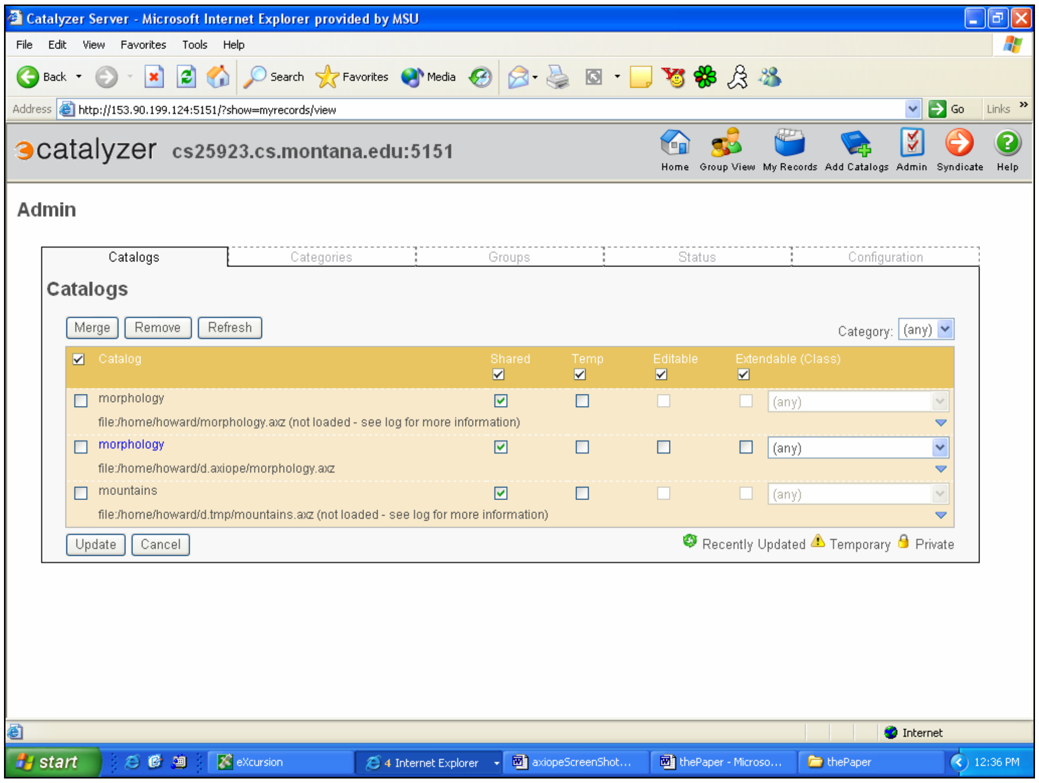
Figure 30 - Axiope Catalyzer Graph Feature


http://153.90.199.124:5151/xml/1gmoilp0.axc - Microsoft Internet Explorer provided by MSU
http://153.90.199.124:5151/xml/1gmoilp0.axc

```
  <?xml version="1.0" encoding="ISO-8859-1" ?>
- <CellList name="morphology">
  - <Contents>
    + <Folder name="Digitizations">
    - <Folder name="Preparations">
      - <Contents>
        - <Preparation name="Preparation">
            <Area>CA1</Area>
            <Type>Pyramidal Cell</Type>
            <Fill>in vivo</Fill>
            <Lesion>None</Lesion>
            <Name>CA1-Pyramidal Cell-in vivo-None</Name>
            <Age>Young</Age>
          </Preparation>
        - <Preparation name="Preparation 2">
            <Area>DG</Area>
            <Type>Granule Cell</Type>
            <Fill>in vivo</Fill>
            <Lesion>None</Lesion>
            <Name>DG-Granule Cell-in vivo-None</Name>
            <Age>Young</Age>
          </Preparation>
```

Figure 31 - Catalog in XML Format

A user may download the contents of a catalog as an XML file (Figure 31). A user can also download the contents of a class as a comma-separated value (CSV) file which can then be imported into a spreadsheet program. An especially nice user control feature is the administrative page that allows the owner of a website to manage catalogs and to configure and change network, web server, and password settings (Figure32).


Catalyzer Server - Microsoft Internet Explorer provided by MSU
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://153.90.199.124:5151/?show=myrecords/view
Go Links
catalyzer cs25923.cs.montana.edu:5151
Home Group View My Records Add Catalogs Admin Syndicate Help
Catalogs Categories Groups Status Configuration
Merge Remove Refresh
Update Cancel
Internet
start eXcursion 4 Internet Explorer axiopeScreenShot... thePaper - Microso... thePaper 12:36 PM

Admin

Catalogs

Category: (any)

✓	Catalog	Shared ✓	Temp ✓	Editable ✓	Extendable (Class) ✓	
☐	morphology file:/home/howard/morphology.axz (not loaded - see log for more information)	✓	☐	☐	☐	(any)
☐	morphology file:/home/howard/d.axiope/morphology.axz	✓	☐	☐	☐	(any)
☐	mountains file:/home/howard/d.tmp/mountains.axz (not loaded - see log for more information)	✓	☐	☐	☐	(any)

Recently Updated Temporary Private

Figure 32 - Axiope Catalyzer Administrative Page

One disappointing lack of user control is the inability for a user to edit records via the web interface. The interface provides a read-only view of the database. Despite all the other excellent user control features, the inability to edit records detracts from what easily was an excellent score.

This system receives a score of ***average*** for user control.

Help and Documentation

As mentioned in the section on *Learnability* the system has a simple but useful help page available at all times. Although not especially obvious, the Catalyzer logo in the top frame is hyperlinked to the project homepage [3] which contains extensive information on publishing and configuring the website.

This system receives a score of ***excellent*** for help and documentation.

Valid HTML

Attempts to validate the HTML for a catalog homepage and individual frames were not successful. The validator was not able to detect a character encoding label and therefore could not validate any pages.

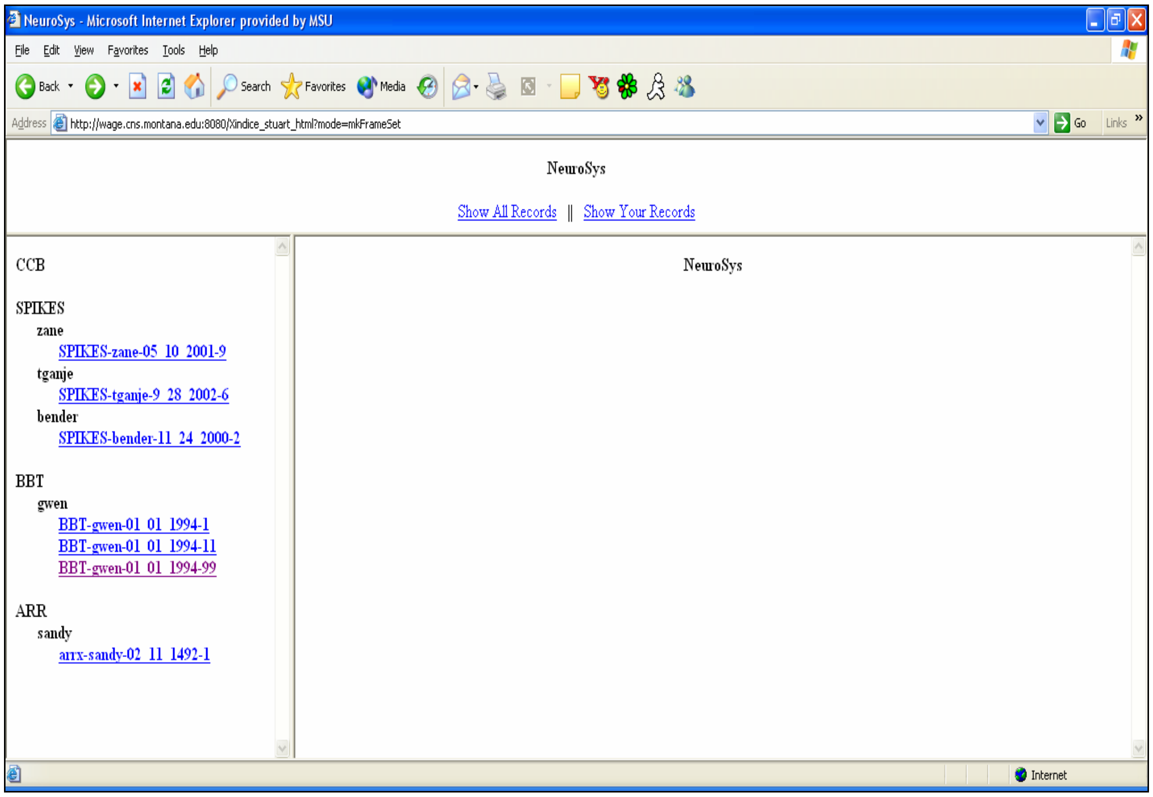
This system receives a score of ***poor*** for valid HTML.

6.5 NeuroSys

The ratings for the NeuroSys web interface, except for the category of generating valid HTML, are calculated with the input of the author which is averaged with the input of five independent evaluators. The summary for each category consists of the author's evaluation and score, the scores of the independent evaluators, and the overall score for the category.

Visual Design

The first page of the NeuroSys interface is the login page, so the subsequent page that represents the actual database interface is evaluated (Figure 33). The database interface is similar in design to that of the Axiope Catalyzer system. It consists of three frames with similar functions. The top frame displays the title, the left frame is the navigation panel, and the right frame is the data display area. The systems differ in that the left frame of the NeuroSys interface does not allow users to expand and contract the file tree icons. Each file category is expanded by default and only the record names are hyperlinked. The right frame by default displays the title of the database unlike the database summary table provided by the Axiope Catalyzer system.


NeuroSys - Microsoft Internet Explorer provided by MSU
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://wage.cns.montana.edu:8080/Xindice_stuart_html?mode=mkFrameSet
Go Links
Internet

NeuroSys

Show All Records || Show Your Records

CCB

SPIKES
zane
SPIKES-zane-05_10_2001-9
tganje
SPIKES-tganje-9_28_2002-6
bender
SPIKES-bender-11_24_2000-2

BBT
gwen
BBT-gwen-01_01_1994-1
BBT-gwen-01_01_1994-11
BBT-gwen-01_01_1994-99

ARR
sandy
arrx-sandy-02_11_1492-1

NeuroSys

Figure 33 - NeuroSys Web Interface

The design of the interface is simple and clean, though no logos or color scheme are provided. The NeuroSys interface also offers less functionality than the Catalyzer interface, so the author assigns a score of ***good*** for visual design.

The independent evaluators gave the following scores:

fair	*average*	*poor*	*good*	*fair*

The overall score for visual design is ***average***.

Learnability

The NeuroSys web interface does not currently offer links to help documentation. The system however is simple and intuitive to use. The text fields that allow a user to enter a new option for a pull down menu are clearly labeled '*Enter New Option*' and the submit buttons are clearly labeled with the associated action. Due to the lack of help and documentation the author assigns a score of ***fair*** for learnability.

The independent evaluators gave the following scores:

fair	*good*	*good*	*average*	*fair*

The overall score for learnability is ***average***.

Efficiency

The NeuroSys interface does not force users to make redundant actions to perform a task. However it does not provide access to a table or frame containing links for database operations as the previously reviewed systems did. For this reason the author assigns the system a score of ***fair*** for efficiency.

The independent evaluators gave the following scores:

average	*good*	*fair*	*good*	*average*

The overall score for efficiency is ***average***.

Errors

Though the system is designed to avoid errors with data entry, it lacks in its capability to provide error messages. For instance if user does enter a user name or password and presses the '*Login'* button, the login page is simply redisplayed with no prompt for the missing information. The author assigns this system a score of ***poor*** for error handling.

The independent evaluators gave the following scores:

poor	*fair*	*excellent*	*average*	*good*

The overall score for errors and error handling is ***average***.

Consistency

As with the Axiope Catalyzer system the frame layout is consistent from page to page. The frame proportions are consistent as are the font and terminology for field labels in a form. The author assigns a score of ***excellent*** for consistency.

The independent evaluators gave the following scores:

good	*average*	*excellent*	*good*	*good*

The overall score for consistency is ***good***.

User Control

The NeuroSys system offers little user control over the data display. The data is displayed in form fashion only. The user may control the selection of which record names to display in the left frame by selecting either the ‘*Show Your Records’* or ‘*Show All Records’* link in the top frame. The system does however offer the ability edit and delete files remotely, unlike the Axiope Catalyzer system. The author assigns a score of ***average*** for user control.

The independent evaluators gave the following scores:

fair	*good*	*average*	*good*	*poor*

The overall score for user control is ***average***.

Help and Documentation

The system currently offers no *Question and Answer* or *FAQ* pages and the author assigns a score of ***poor*** for help and documentation.

The independent evaluators gave the following scores:

poor	*fair*	*poor*	*poor*	*poor*

The overall score for help and documentation is ***poor***.

Valid HTML

The URL's of both the login page and the initial database interface were entered into the W3C HTML validator and the follow message was displayed:

This page is valid HTML 4.01 Transitional!

This system receives a score of ***excellent*** for producing valid HTML.

6.6 Evaluation Summary

This section summarizes the evaluation results in a table.

System	Visual Design	Learnability	Efficiency	Errors	Consistency	User Control	Help and Documentation	Valid HTML	Average Score
IBVD	3	4	3	2	5	5	3	1	3.2
SenseLab	5	2	5	5	5	5	2	4	4.1
Axiope	5	5	5	2	5	3	5	1	3.9
NeuroSys	3	3	3	3	4	3	1	5	3.1

1 – poor 2 – fair 3 – average 4 – good 5 – excellent

The numeric results indicate that the overall score for each system is as follows:

Internet Brain Volume Database	***average***
SenseLab	***good***
Axiope	***good***
Neurosys	***average***

Chapter Seven summarizes this paper and project, critically reviews the results of the evaluation, and offers recommendations for future work on the project.

7. Conclusions and Future Directions

This chapter summarizes this paper and critically reviews the web-based interface implemented for the NeuroSys database project. The chapter ends with recommendations for future work on the project.

7.1 Summary

This paper introduces the NeuroSys Database Project [22] and explains the motivation for the development of a web-based interface for the project. Literature related to the topic of user interface design is reviewed and several other scientific database systems that provide a web-based interface are introduced. The underlying technologies used in the implementation of this project are discussed and information about the system is provided for both end users and future programmers. The web interfaces for this project and the other systems are evaluated primarily on criteria that serve as guidelines for usability and user interface design.

7.2 Conclusions

The original objective of this project was to implement a web-based interface for the NeuroSys project that allows browsing, data entry, and query capabilities for a remote database. The first two goals were fully met. The goal of a useful query mechanism was not met. The system has the limited query capability of either pulling records that belong to a particular user or pulling all the files in the database.

The study and evaluation of three other web-based database systems provides a good background for further evaluating the NeuroSys web interface. The NeuroSys system scores strongly in the areas of consistency and generating valid HTML. The system does not fare as well in the areas of visual design, learnability, efficiency, error handling, and adequate help and documentation pages; receiving ***average*** or ***poor*** scores in those areas.

7.3 Future Directions

The NeuroSys web interface project is a fully functional prototype that meets the original goals of allowing users to browse and edit a remote database. A flexible query mechanism would greatly enhance the system. Programmers for the client-side NeuroSys application [22] are currently enhancing the query mechanism and future programmers on the web interface project may be able to utilize the API of the client-side program.

Visual design of the system could be improved by adding graphics and a logo and by adopting a consistent color scheme throughout the site. The system has an obvious lack of help documentation. A help icon in the top frame that links to a simple help page would be a useful addition. Axiope Catalyzer [3] offers this feature. A walk-through tutorial, as provided by the IBVD system [13], would also enhance the system.

Effort should be put into evaluating all the scenarios that can generate an error and appropriate routines for handling the errors need to be implemented. The SenseLab system [24] does an excellent job of handling errors and would be a good system to emulate.

The efficiency of the system could be improved by the addition of hyperlinked icons in the top frame, as in the Axiope Catalyzer system. Such icons allow quick and easy navigation to other areas of the web site. Expandable and collapsible file trees in the left navigation frame would be useful. Axiope Catalyzer provides this feature and it also hyperlinks all levels of the file tree hierarchy.

Lastly, user control for data display and data entry could be improved. All of the other systems allowed a great deal of flexibility in how to display the data, including text and

XML format, tables, forms, and various types of graphs. A new data entry feature for the NeuroSys system could be the ability to expand the form in the right frame to a full screen view as is done in one of the SenseLab databases. Another nice user control feature would be the addition of an administrative page to control web site and network settings as in the Axiope Catalyzer system.

The foundation for an operational web interface for the NeuroSys Database Project has been built. Addition of the enhancements mentioned above will offer challenges and rewards for future programmers on the project and will result in a world class web interface for the system.

8. References

[1] Apache Tomcat, http://jakarta.apache.org/tomcat/ (2004).

[2] Apache Xindice, http://xml.apache.org/xindice/ (2004).

[3] Axiope Catalyzer, http://www.axiope.com/ (2004).

[4] BrainInfo, http://braininfo.rprc.washington.edu/mainmenu.html (2004).

[5] Cannon, R. C., Howell, F. W., Goddard, N. H. and De Schutter, E. Non-curated Distributed Databases for Experimental Data and Models in Neuroscience. *Network: Computation In Neural Systems* **13**, 415-428 (2002).

[6] Cato, J. *User-centered Web Design* (Pearson Education Limited, 2001).

[7] Cellular Properties Database, http://senselab.med.yale.edu/senselab/CellPropDB/default.asp (2004).

[8] Falkner, J., and Jones, K. *Servlets and JavaServer Pages: the J2EE Web Tier* (Addison Wesley, 2003).

[9] Goddard, N. H., Cannon, R. C. and Howell, F. W. Axiope Tools for Data Management and Data Sharing. *Neuroinformatics Journal* **1**, 271-284 (2003).

[10] HTML 4.01 Specification, http://www.w3.org/TR/html4/ (2004).

[11] Human Brain Project, http://www.nimh.nih.gov/neuroinformatics/index.cfm (2004).

[12] Hunter, J., and Crawford, W. *Java Servlet Programming* (O'Reilly and Associates, Inc., 1998).

[13] Internet Brain Volume Database, http://www.cma.mgh.harvard.edu/ibvd/ (2004).

[14] Java 2 Platform, Standard Edition (J2SE) 1.4.2 API Specification, http://java.sun.com/j2se/1.4.2/docs/api/ (2004).

[15] Java Servlet API Specification - Version 2.3, http://java.sun.com/products/servlet/reference/api/index.html (2004).

[16] JavaScript, http://developer.netscape.com/docs/manuals/javascript.html (2004).

[17] Java Swing, http://java.sun.com/products/jfc/ (2004).

[18] Java Version 1.4, http://java.sun.com/j2se/1.4.2/docs/index.html (2002).

[19] Miller, G. A. The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information. *The Psychological Review* **63**, 81-97 (1956).

[20] Nadkarni, P., Marenco, L., Chen, R., Skoufos, E., Shepherd, G., and Miller, P. Organization of Heterogeneous Scientific Data Using the EAV/CR Representation. *Journal of the American Medical Informatics Association* **6**, 478-493 (1999).

[21] Neuronal Database, http://senselab.med.yale.edu/senselab/NeuronDB/default.asp (2004).

[22] NeuroSys, http://neurosys.cns.montana.edu (2004).

[23] Pittendrigh, S. and Jacobs, G. Neurosys: A Semistructured Laboratory Database. *Neuroinformatics Journal* **1**, 167-176 (2003).

[24] SenseLab, http://senselab.med.yale.edu/senselab/ (2004).

[25] Shneiderman, B. *Designing the User Interface: Strategies for Effective Human-Computer Interaction* (Addison-Wesley, 1998).

[26] Smith, S., and Mosier, J. Guidelines for Designing User Interface Software (Technical Report ESD-TR-86-278). Hanscom Air Force Base, MA: USAF Electronic Systems Division (1986).

[27] W3C HTML Home Page, http://www.w3.org/MarkUp (2004).

[28] W3C MarkUp Validation Service, http://validator.w3.org/ (2004).

[29] W3Schools HTML Tutorial, http://www.w3schools.com/html/default.asp (2004).

[30] W3Schools JavaScript Tutorial, http://www.w3schools.com/js/default.asp (2004).

[31] XHTML 1.0 Recommendation, http://www.w3.org/TR/xhtml1/ (2004).

[32] XPath, http://www.w3.org/TR/xpath (2004).

Appendix A - Usability Test and Questionnaire

Usability Test for the Web-Based Interface for the
NeuroSys Database Project

Thank you for helping evaluate my web interface! This is an **anonymous** evaluation and it should take about **10 minutes**.

Below are directions for some simple tasks to perform. Attached is a questionnaire to fill out after performing the tasks.

1. go to: **http://wage.cns.montana.edu:8080/Xindice_stuart_html**

2. the user name and password are hard-coded, so press **'Login'**

3. **select** one of the '**gwen'** records in the left frame

4. **edit** various fields and press **'Save As'** to save a copy of the record
   (i.e. try changing the URL field at the bottom to your homepage URL)

   PLEASE NOTE the name of the new record that appears. It will look something like **BBT-stuarth-04_05_2004-XX** (where XX is a sequence number).
   If other users are testing, there will be other files with a similar name but a different sequence number. Be sure you are editing and/or deleting your file.

5. **note** if the changes were saved under the newly created file
   (i.e. does the new URL hyperlink reflect your changes? Does it work?)

6. **attempt** to '*break'* the interface
   (i.e. can you cause an exception or error condition?)

7. **delete** the file that you saved

8. **fill out** the attached questionnaire

Thanks again for your help!

Stuart Howard

Usability Questionnaire for the Web-Based Interface for the NeuroSys Database Project

Adapted from: Cato, John *User-centered Web Design* (Pearson Education Limited, 2001).

Date:

What did you like?

What did you dislike?

Did you meet your objectives?

Please read the statements below and score each with a number between 1 and 5 to indicate how true the statement is for you. Feel free to add any other comments.	**Totally disagree**				**Totally agree**
About the system	**1**	**2**	**3**	**4**	**5**
I found the visual presentation excellent.					
It is very easy to understand straight away.					
I found it a very efficient way to get things done.					
I never made any mistakes.					
I never had to change or correct anything I entered.					
The system was coherent and consistent (i.e. page layout, text format, terminology, etc.).					
I am able to modify the data entry and data display format.					
I always feel in control using the system.					
I could always get help quickly (i.e. Help page, FAQ).					
I achieved what I wanted very effectively.					

Other comments (optional):

Appendix B – Source Code

```
package htmlHandler;

import java.util.Vector;
import java.util.Arrays;

/**
 * HtmlGenerator.java
 *
 * Utility class that offers static methods for generating
 * HTML elements.
 *
 * @author Stuart Howard
 * @version 1.0
 */
public class HtmlGenerator
{
  protected static final String BR = "<br/>";       // line break
  protected static final String AMP = "&amp;";      // ampersand

  /**
   * Constructor (no-op).
   */
  public HtmlGenerator()
  {
  }

  /**
   * Returns HTML string that represents a text field.
   *
   * @param name name of the text field
   * @param value value of the text field
   * @param size size of the text field
   * @return HTML string to generate text field
   */
  public static String makeTextField(String name, String value, int size)
  {
    String txt = "";                         // text field string
    txt += "<input type=\"text\" name=\"";
    txt += name;
    txt += "\" value=\"";
    txt += value;
    txt += "\" size=\"";
    txt += String.valueOf(size);
    txt += "\">";
    return txt;
  }

  /**
   * Returns HTML string that represents a text area.
   *
   * @param name name of the text area
   * @param rows depth of the text area
   * @param cols width of the text area
   * @param value value of the text area
   * @return HTML string to generate text area
   */
  public static String makeTextArea(String name, int rows, int cols, String value)
  {
    String txt = "";                         // text area string
    txt += "<textarea name=\"";
    txt += name;
    txt += "\" rows=\"";
    txt += String.valueOf(rows);
    txt += "\" cols=\"";
    txt += String.valueOf(cols);
    txt += "\">\n";
    txt += value;
    txt += "\n";
    txt += "</textarea>";
    return txt;
  }
```

```
/**
   * Generates an HTML check box.
   *
   * @param name the name of the check box
   * @param checked 'true' if checked, otherwise 'false'
   * @return HTML string that generates a check box
   */
  public static String makeCheckBox(String name, boolean checked)
  {
    String txt = "";                         // check box string
    txt += "<input type=\"checkbox\" name=\"";
    txt += name;
    txt += "\" value=\"true\"";
    if(checked)                                      // check the box
    {
      txt += "checked=\"checked\"";
    }
    txt += ">";
    return txt;
  }

  /**
   * Generates an HTML pull down list.
   *
   * @param name the name of the select element
   * @param firstValue the first value to display in the pull down menu
   * @param guiInfo a vector containing rest of items to display
   * @return HTML string that generates a pull down list
   */
  public static String makePullDownBox(String name, String firstValue,
                                     Vector guiInfo)
  {
    String str = "";                         // pull down list string
    str += "<select name=\"";
    str += name;
    str += "\">\n";
    str += "<option value=\"";
    str += firstValue;                       // assure this appears first
    str += "\">";
    str += firstValue;
    str += "\n";
    for(int i = 0; i < guiInfo.size(); i++)  // remaining items in menu
    {
      String item = (String)guiInfo.get(i);
      if(!item.equals(firstValue))           // avoid duplicates in pull down box
      {
        str += "<option value=\"";
        str += item;
        str += "\">";
        str += item;
        str += "\n";
      }
    }
    str += "</select>";
    return str;
  }

  /**
   * Generates an HTML pull down list for a ScreenSelect widget.
   * Utilizes JavaScript to react to 'onClick' event.
   *
   * @param name the name of the select element
   * @param firstValue the first value to display in the pull down menu
   * @param guiInfo a vector containing rest of items to display
   * @param selfURL the target URL for the JavaScript call
   * @param docID the document ID number
   * @return HTML string that generates a pull down list for ScreenSelect widgets
   */
  public static String makeSSelectPullDownBox(String name, String firstValue,
                                            Vector guiInfo, String selfURL,
                                            String docID)
  {
    String str = "";                         // pull down list string
    String getstr = selfURL + "?mode=pullFile" + AMP + "docID=" + docID
        + AMP + "screenSelect=yes" + AMP + "freshFromDB=no" + "#" + name;
    str += "<select name=\"" + name + "\" onClick=\"postform('0','" + getstr + "')\">\n";
    str += "<option value=\"";
    str += firstValue;                       // assure this appears first
    str += "\">";
    str += firstValue;
    str += "\n";
    int size = guiInfo.size();
    for(int i = 0; i < size; i++)            // remaining items in menu
    {
      String item = (String)guiInfo.get(i);
      if(!item.equals(firstValue))           // avoid duplicates in the pull down box
      {
        str += "<option value=\"";
        str += item;
        str += "\">";
        str += item;
        str += "\n";
      }
    }
    str += "</select>";
    return str;
  }

  /**
   * Generates an HTML anchor element(hyperlink).
   *
   * @param in_link he target URL of the link
   * @param target where to open the target URL(_blank, _self, _parent, _top)
   * @param label the text to hyperlink
   * @return HTML string that generates a hyperlink
   */
  public static String makeHyperLink(String in_link, String target, String label)
  {
    String link = "<a href=\"" + in_link + "\"";    // the hyperlink string
    if(target != null)
    {
      link += " target=\"" + target + "\">";
    }
    link += label + "</a>";
    return(link);
  }

  /**
   * Generates element that defines the relationship between two linked documents.
   *
   * @param rel defines the relationship between the current document and the targeted
document
   * @param href he target URL of the resource
   * @param type specifies the MIME type of the target URL
   * @return HTML string that generates a link element
   */
  public static String makeLink(String rel, String href, String type)
  {
    String link = "";                        // the link string
    link += "<link rel=\"";
    link += rel;
    link += "\" href=\"";
    link += href;
    link += "\" type=\"";
    link += type;
    link += "\">";
    return(link);
  }

  /**
   * Defines a script.
   *
   * @param language specifies the scripting language
   * @param type indicates the MIME type of the script
   * @param src defines a URL to a file that contains the script
   * @return HTML string that generates a script element
   */
  public static String makeScript(String language, String type, String src)
  {
    String script = "";                             // the script string
    script += "<script language=\"";
    script += language;
    script += "\" type=\"";
    script += type;
    script += "\" src=\"";
    script += src;
    script += "\">";
    script += "\n";
    script += "</script>";
    return(script);
  }
  /**
   * Generates bold text.
   *
   * @param toBold the string to bold
   * @param classString he class of the element
   * @return HTML string that generates bold element
   */
  public static String makeBold(String toBold, String classString)
  {
    String bold = "";                      // the bold string
    bold += "<b class=\"";
    bold += classString;
    bold += "\">";
    bold += "\n";
    bold += toBold;
    bold += "\n";
    bold += "</b>";
    return bold;
  }

  /**
   * Generates the beginning element for a form.
   *
   * @param action a URL that defines where to send the data when the submit button is
pushed
   * @param target where to open the target URL(_blank, _self, _parent, _top)
   * @param method the HTTP method for sending data to the action URL(get, post)
   * @return HTML string that generates the opening form element
   */
  public static String makeOpenForm(String action, String target, String method)
  {
    String openForm = "";                  // the opening form element string
    openForm += "<form action=\"";
    openForm += action;
    openForm += "\" target=\"";
    openForm += target;
    openForm += "\" method=\"";
    openForm += method;
    openForm += "\">";
    return openForm;
  }

  /**
   * Generates HTML string that represents a button element.
   *
   * @param name name of the button
   * @param value value of the button
   * @return HTML string that generates the button element
   */
  public static String makeButton(String name, String value)
  {
    String bttn = "";                      // the button string
    bttn += "<input type=\"submit\" name=\"";
    bttn += name;
    bttn += "\" value=\"";
    bttn += value;
    bttn += "\">";
    return bttn;
  }

  /**
   * Generates HTML string that contains DOCTYPE information.
   *
   * @param type document type
   * @return HTML string to generate DOCTYPE element
   */
  public static String makeDocType(String type)
  {
    String title;                          // strict, loose, transitional
    if(type.equals("strict"))
    {
      title = "";
    }
    else if(type.equals("loose"))
    {
      title = " Transitional";
    }
    else
    {
      type = "frameset";
      title = " Frameset";
    }
    String front = "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.01";
    String middle = "//EN\"\n\"http://www.w3.org/TR/html4/";       // commom parts of
DOCTYPE element
    String end = ".dtd\">";
    return(front + title + middle + type + end);
  }

  /**
   * Generates an HTML string that contains version information.
   *
   * @return HTML string to generate version information element
   */
  public static String makeVersionInfo()
  {
    return("<?xml version=\"1.0\" encoding=\"UTF-8\" ?>");
  }

  /**
   * Generates an HTML string that displays validator icon
   * and link to validator.
   *
   * @author http://validator.w3.org/  Modified by: Stuart Howard
   * @return HTML string to generate the validator
   */
  public static String makeValidator()
  {
    String validator = "";                   // the validator icon/hyperlink
    validator += "<p>\n";
    validator += "<a href=\"http://validator.w3.org/check/referer\">\n";
    validator += "<img border=\"0\" src=\"http://www.w3.org/Icons/valid-html401\"\n"
        + " alt=\"Valid HTML 4.01!\" height=\"31\" width=\"88\">\n";
    validator += "</a>\n";
    validator += "</p>";
    return validator;
  }
}
```

```
package htmlHandler;

import emlCommon.*;
import emldb.*;
import java.io.*;
import java.text.*;
import java.util.*;
import java.util.regex.*;
import java.net.*;
import javax.servlet.*;
import javax.servlet.http.*;
import nanoxml.*;
import org.apache.log4j.Logger;
import org.apache.log4j.PropertyConfigurator;
import xmlserver.*;
import ScreenModel.*;

/**
 * htmlHandler.java
 *
 * Servlet class that generates HTML interface for the Neurosys
 * Database Project.  Allows users to query and edit a remote XML
 * database via a web interface.
 *
 * @author Stuart Howard
 * @version 1.0
 */
public class htmlHandler
    extends HttpServlet
{
  public final Logger log;                    // message logger
  protected boolean info;                     // logger flag
  protected Properties emldbProperties;               // URL properties
  protected Properties protocolProperties;   // database properties
  protected String serverURL;                 // database server URL
  protected String selfURL;                   // htmlHandler servlet URL
  protected String dataTemplateCollection;   // template collection name
  protected String dataDataCollection;               // data collection name
  protected String debug;                     // flag for debug display on/off
  protected final String BR;                  // <br/> element
  protected final String AMP;                 // &amp;
  protected final String SPC;                 // &nbsp;
  protected int count;                        // session count

  /**
   * Constructor.
   *
   * @return an instance of htmlHandler
   */
  public htmlHandler()
  {
    log = Logger.getLogger(this.getClass().getName());
    info = log.isDebugEnabled();
    serverURL = "";
    selfURL = "";
    dataTemplateCollection = "";
    dataDataCollection = "";
    debug = "";
    BR = "<br/>";
    AMP = "&amp;";
    SPC = "&nbsp;";
    count = 0;
  }

  /**
   * Initializes an instance of the servlet.
   *
   * @param config supplies servlet with initialization parameters
   */
  public void init(ServletConfig config) throws ServletException
  {
    try
    {
      super.init(config);
      debug = config.getInitParameter("debug");             // this is set in the web.xml
file
      ClassLoader cl = this.getClass().getClassLoader();
      URL url = cl.getResource("resources/log4j.properties");
      PropertyConfigurator.configure(url);
        // load property lists and retrieve values
      InputStream is = cl.getResourceAsStream("./resources/emldb.properties");
      emldbProperties = new Properties();
      emldbProperties.load(is);
      is.close();
      is = cl.getResourceAsStream("./resources/protocol.properties");
      protocolProperties = new Properties();
      protocolProperties.load(is);
      is.close();
      serverURL = emldbProperties.getProperty("serverURL"); // assign database server URL
      selfURL = emldbProperties.getProperty("selfURL");            // assign htmlHandler
servlet URL
      dataTemplateCollection = protocolProperties.getProperty("dataTemplateCollection");
      dataDataCollection = protocolProperties.getProperty("dataDataCollection");
    }
    catch(Exception e)
    {
      e.printStackTrace(System.err);
    }
  }

  /**
   * Handles HTTP GET requests.
   *
   * @param request the HTTP request object
   * @param response the HTTP response object
   */
  public void doGet(HttpServletRequest request,
                  HttpServletResponse response) throws IOException, ServletException
  {
    doPost(request, response);
  }

  /**
   * Handles HTTP POST requests.  Serves as main dispatcher for this class.
   *
   * @param request the HTTP request object
   * @param response the HTTP response object
   */
  public void doPost(HttpServletRequest request,
                   HttpServletResponse response) throws IOException, ServletException
  {
    count++;
    PrintWriter out = response.getWriter();                // output stream
    String mode = request.getParameter("mode");                   // determines servlet
action
    String save = request.getParameter("save");                   // save parameter
    String save_as = request.getParameter("save_as");             // save_as parameter
    String delete = request.getParameter("delete");        // delete parameter
    String screenSelect = request.getParameter("screenSelect");    // screen select
parameter
    StringServletClient scc = null;                        // offers API for DB
operations
    String user_name = request.getParameter("user_name");   // current user
    if(user_name != null)                                  // replace spaces with '_'
    {
      Pattern p = Pattern.compile("[\\s]+");
      Matcher m = p.matcher(user_name);
      user_name = m.replaceAll("_");
    }
    String password = request.getParameter("password");     // current user's password
    String group = request.getParameter("group");   // current user's lab group
    String message = request.getParameter("message");       // message parameter
    String docID = request.getParameter("docID");   // document ID parameter
    response.setContentType("text/html");
      // refer to http://novocode.de/doc/servlet-essentials/chapter2c.html
    response.setHeader("pragma", "no-cache");
    HttpSession session = request.getSession(true);
    if(user_name != null)                          // set session variables
    {
      session.setAttribute("user_name", user_name);
      session.setAttribute("password", password);
      session.setAttribute("group", group);
    }
    String session_user_name = (String)session.getAttribute("user_name");
    String session_password = (String)session.getAttribute("password");
    String session_group = (String)session.getAttribute("group");
    if(session_user_name != null &&
       session_password != null &&
       session_group != null)
    {
        // build StringServletClient object for use in database operations (pullfile,
save, save_as)
```

```
      scc = new StringServletClient(serverURL, session_user_name,
                              MD5.md(session_password), session_group);
    }
    if(mode == null || session_user_name == null)
    {
      out.println(logIn());                                // make login page
    }
    else if(mode.equals("mkFrameSet"))
    {
      if(session_user_name != null && !session_user_name.equals(""))
      {
        out.println(mkFrameSet());                         // make DB interface
      }
      else
      {
        out.println(logIn());                              // make login page
      }
    }
    else if(mode.equals("defaultLeft"))
    {
      out.println(defaultLeft(request, response));         // make left navigation frame
    }
    else if(mode.equals("defaultTop"))
    {
      defaultTop(request, response);                       // make top frame
    }
    else if(mode.equals("messagePage"))
    {
      out.println(messagePage(message));                   // make generic HTML page
    }
    else if(save != null)                                  // 'save' button pressed ->
save file
    {
      DataEntryModel liveTree = getGUITree(serverURL, session_user_name,
session_password,
                                         session_group, docID);

      fixWholeTree(request, liveTree);                             // update tree with
any changes
      out.println(makeDisplay(request, liveTree, session_user_name));      // display the
new tree
      saveFile(liveTree, docID, scc);                      // save new tree to database
    }
    else if(save_as != null)                               // 'save_as' button pressed -
> save copy of file
    {
      DataEntryModel liveTree = getGUITree(serverURL, session_user_name,
session_password,
                                       session_group, docID);
      fixWholeTree(request, liveTree);                             // update tree with
any changes
      String newDocID = addFile(liveTree, scc, session_user_name); // save to the
database
      out.println(messagePage(refreshLeftFrame()));        // refresh left navigation
frame
      out.println(messagePage(refreshRightFrame(newDocID)));       // refresh right frame
    }
    else if(delete != null)                                // the 'delete' button was
pressed
    {
      scc.deleteFile(docID);                               // delete the file
      out.println(messagePage(refreshLeftFrame()));        // refresh left frame
      out.println(messagePage(confirmDelete(docID)));              // confirmation
message in right frame
    }
    else if(screenSelect != null)                          // a new ScreenSelection item
was chosen
    {
      DataEntryModel liveTree = getGUITree(serverURL, session_user_name,
session_password,
                                       session_group, docID);
        // alter the tree with the new SS parameter
      fixWholeTree(request, liveTree);
        // display the new tree
      out.println(makeDisplay(request, liveTree, session_user_name));
    }
    else if(mode.equals("pullFile"))                       // pull a file from database
    {
        // pull the file from the DB
      DataEntryModel liveTree = getGUITree(serverURL, session_user_name,
session_password,
                                        session_group, docID);
        // display the form
      out.println(makeDisplay(request, liveTree, session_user_name));
    }
  }

  /**
   * Saves(overwrites) a file identified by docID to the database.
   *
   * @param liveTree the DataEntryModel object representing the file
   * @param docID the name of the file
   * @param scc the StringServletClient object that provides API for database operations
   */
  protected void saveFile(DataEntryModel liveTree, String docID, StringServletClient scc)
  {
    String dataString = liveTree.SaveData();                // save template information
    scc.saveWithString(dataString, docID, dataDataCollection);
    String userAdditionsString = liveTree.SaveUserAdditions();     // save data
information
    scc.saveWithString(userAdditionsString, docID, dataTemplateCollection);
  }

  /**
   * Copies a file to the database with a new doc ID.
   *
   * @param liveTree the DataEntryModel object representing the file
   * @param scc the StringServletClient object provides API for database operations
   * @param owner the owner of the new file
   */
  protected String addFile(DataEntryModel liveTree, StringServletClient scc, String
owner)
  {
      // get template value
    GUINode templateNode = liveTree.getNode("/EML/HEAD/TEMPLATE/");
    String template = templateNode.getValue();
      // get the OWNER node and set to new owner
    GUINode ownerNode = liveTree.getNode("/EML/HEAD/OWNER/");
    ownerNode.setValue(owner);
      // save the file to DB
    String dataString = liveTree.SaveData();
    String update = scc.saveAsWithString(dataString, dataDataCollection, owner,
template);
    String docID = update.substring(update.indexOf(":") + 1);
    GUINode docIDNode = liveTree.getNode("/EML/HEAD/DOC_ID/");
    docIDNode.setValue(docID);
    String userAdditionsString = liveTree.SaveUserAdditions();
    if(userAdditionsString != null && !userAdditionsString.equals(""))
    {
      scc.saveWithString(userAdditionsString, docID,
                              Protocol.dataTemplateCollection);
    }
    return docID;
  }

  /**
   * Refreshes right frame with file-delete confirmation message.
   *
   * @return a Javascript command to confirm that a file is deleted
   */
  protected String confirmDelete(String docID)
  {
    String jscript = null;                                  // the javascript command
    try
    {
      jscript = "";
      String message = docID + SPC + "deleted";
      message = StringServletClient.encodeString(message);
      String src = selfURL + "?mode=messagePage&message=" + message;
      jscript += ("<script language=\"JavaScript\">");
      jscript += ("setTimeout('parent.frames[2].location.replace(\"" + src +
                  "\")',800)");
      jscript += ("</script>");
    }
    catch(UnsupportedEncodingException exception)
    {
      log.error(exception.getMessage() + "\n" +
emlCommon.util.StackTrace.print(exception));
    }
    return jscript;
  }
```

```
    /**
     * Refreshes the right frame with the given document ID.
     *
     * @return a JavaScript command to refresh the right frame with the given file
     */
    protected String refreshRightFrame(String docID)
    {
      String jscript = "";                                   // the javascript command
      String src = selfURL + "?mode=pullFile&docID=" + docID + "&freshFromDB=yes";
      jscript += ("<script language=\"JavaScript\">");
      jscript += ("setTimeout('parent.frames[2].location.replace(\"" + src +
                  "\")',800)");
      jscript += ("</script>");
      return jscript;
    }

    /**
     * Refreshes the left frame.
     *
     * @return a JavaScript command to refresh the left frame
     */
    protected String refreshLeftFrame()
    {
      String jscript = "";                                   // the javascript command
      String src = selfURL + "?mode=defaultLeft";
      jscript += ("<script language=\"JavaScript\">");
      jscript += "\n";
      jscript += ("setTimeout('parent.frames[1].location.replace(\"" + src +
                  "\")',800)");
      jscript += "\n";
      jscript += ("</script>");
      jscript += "\n";
      return jscript;
    }


    /**
     * Generates an HTML login page.
     *
     * @return an HTML string that is the login page
     */
    protected String logIn()
    {
      String login = "";                                     // the HTML string
      try
      {
        login += (HtmlGenerator.makeVersionInfo());          // version element
        login += (HtmlGenerator.makeDocType("loose"));       // DOCTYPE element
        login += ("<html>");
        login += ("<head>");
        login += ("<link rel=StyleSheet href=\"http://wage.cns.montana.edu/styles/lib.css\"
  type=\"text/css\">");
        login += ("<title>");
        login += ("</title>");
        login += ("</head>");
        login += ("<body>");
        login += ("<center>");
        login += ("<b class=\"group\">");
        login += ("NeuroSys Login Page");
        login += ("</b>");
        login += (BR);
        login += (BR);
        login += ("<form  action=\"");
        String src = selfURL + "?mode=mkFrameSet";           // call mkFrameSet()
        login += (src);
        login += ("\"   method=\"post\">");
        login += (BR);
        login += ("Enter your user name: ");                 // enter user information
        login += ("<input type=\"text\" name=\"user_name\" value=\"stuarth\">");
        login += (BR);
        login += ("Enter your password: ");
        login += ("<input type=\"password\" name=\"password\" value=\"stuarth\">");
        login += ("<input type=\"hidden\" name=\"group\" value=\"CCB\">");
        login += (BR);
        login += (BR);
        login += ("<input type=\"submit\" name=\"login\" value=\"Login\">");
        login += ("</form>");
        login += ("</center>");
        login += ("</body>");
        login += ("</html>");
      }
      catch(Exception e)
      {
        e.printStackTrace(System.err);
      }
      return login;
    }

    /**
     * Generates an HTML frame layout.
     *
     * @return an HTML string that is a frame layout
     */
    protected String mkFrameSet()
    {
      String set = "";                                        // the HTML string
      try
      {
        set += ("<html>");
        set += ("<head>");
        set += ("<title>");
        set += ("NeuroSys");
        set += ("</title>");
        set += ("</head>");
        set += ("<frameset rows=\"10%,*\">");
        set += ("<frame name=\"topframe\" src=\"");           // set top frame
        String src = selfURL + "?mode=defaultTop";
        set += (src);
        set += ("\" frameborder=\"0\" scrolling=\"no\">");
        set += ("<frameset cols=\"25%,*\">");
        String xpath = "/EML/HEAD";
        src = selfURL + "?mode=defaultLeft" + AMP + "xpath=" + xpath;
        set += ("<frame name=\"leftframe\" src=\"");          // set left frame
        set += (src);
        set += ("\" scrolling=\"yes\">");
        src = selfURL + "?mode=messagePage&message=NeuroSys";
        set += ("<frame name=\"rightframe\" src=\"");         // set right frame
        set += (src);
        set += ("\" scrolling=\"yes\">");
        set += ("</frameset>");
        set += ("</frameset>");
        set += ("</html>");
      }
      catch(Exception e)
      {
        e.printStackTrace(System.err);
      }
      return set;
    }

    /**
     * Returns a sequence of '&nbsp;' for indenting an HTML page.
     *
     * @param howdeep how far to indent
     * @return the sequence of '&nbsp;' as a string
     */
    protected String indent(int howdeep)
    {
      String indent = "";                                            // indent string
      int cnt = 3 * howdeep;
      for(int i = 0; i < cnt; i++)                            // augment indent string
      {
        indent += SPC;
        indent += "\n";
      }
      return indent;
    }

    /**
     * Generates an HTML string to display the file navigation pane in the left frame.
     * @author Sandy Pittendrigh
     *
     * @param request the HTTP request object
     * @param response the HTTP response object
     * @return an HTML string responsible for displaying the file navigation pane in the
  left frame
     */
    protected String defaultLeft(HttpServletRequest request, HttpServletResponse response)
    {
      String left = "";                                       // the left frame HTML string
      try
      {
        left += (HtmlGenerator.makeVersionInfo());
        left += (HtmlGenerator.makeDocType("loose"));
        left += ("<html>");
        left += ("<head>");                              // stylesheet links(currently
  inactive)
        left +=(HtmlGenerator.makeLink("StyleSheet",
  "http://wage.cns.montana.edu/styles/lib.css","text/css"));
        /*
        left += (HtmlGenerator.makeLink("StyleSheet",
                            "http://localhost:8080/stylesheets/pages.css",
                            "text/css"));
         */

        left += ("<title>");
        left += ("</title>");
        left += ("</head>");
        left += ("<body>");
        if(debug.equals("true"))                                // display debug info
  switch
        {
          left += (showDebuggingInfo(request));
        }
        else
        {
          trackSessionCount(request);
        }
        HttpSession session = request.getSession(false);
        String user_name = (String)session.getAttribute("user_name");
        String login_group = (String)session.getAttribute("group");
        String xpath = request.getParameter("xpath");
        if(xpath == null)
        {
          xpath = "/EML/HEAD";
        }
        String passWord = (String)session.getAttribute("password");
        String md5pwd = emlCommon.MD5.md(passWord);          // construct
  StringServletClient to use DB API
        StringServletClient scc = new StringServletClient(serverURL, user_name, md5pwd,
            login_group);                                       // query DB for array
  of records
        String results[] = scc.runQuery(xpath, user_name, login_group);
        int size = results.length;
        Hashtable groupbuckets = new Hashtable();
        for(int i = 0; i < size; i++)                        // loop results and start
  messy sorting process
        {
          String res = null;
          String tmp = results[i];
          try
          {
            res = URLDecoder.decode(tmp, "UTF-8");
          }
          catch(Exception ex)
          {
            ex.printStackTrace(System.err);
          }
          XMLElement xml = new XMLElement();
          xml.parseString(res);
          String owner = NanoXMLToolkit.valueofFirstLabelInstance(xml, "OWNER");
          String template = NanoXMLToolkit.valueofFirstLabelInstance(xml, "TEMPLATE");
          String group = NanoXMLToolkit.valueofFirstLabelInstance(xml, "GROUP");
          String docID = NanoXMLToolkit.valueofFirstLabelInstance(xml, "DOC_ID");
          GroupBucket local_group_bucket;                       // make the 'buckets'
  for sorting
          TemplateBucket local_template_bucket;
          OwnerBucket local_owner_bucket;
          DescriptionBucket local_description_bucket;
          Description local_description;
          local_description = new Description(owner, template, group, docID);
          if((local_group_bucket = (GroupBucket)groupbuckets.get(group)) == null)
          {                                                  // start sorting process
            GroupBucket gb = new GroupBucket(group);
            groupbuckets.put(group, gb);
            local_group_bucket = gb;
          }
          if((local_template_bucket =
              (TemplateBucket)local_group_bucket.getBucket(template)) == null)
          {
            local_template_bucket = new TemplateBucket(template);
          }
          if((local_owner_bucket = (OwnerBucket)local_template_bucket.getBucket(owner))
             == null)
          {
            local_owner_bucket = new OwnerBucket(owner);
          }
          if((local_description_bucket =
              (DescriptionBucket)local_owner_bucket.getBucket(template)) == null)
          {
            local_description_bucket = new DescriptionBucket(template);
          }
          description2DescriptionBucket(local_description, local_description_bucket);
          descriptionBucket2OwnerBucket(local_description_bucket, local_owner_bucket);
          ownerBucket2TemplateBucket(local_owner_bucket, local_template_bucket);
          templateBucket2GroupBucket(local_template_bucket, local_group_bucket);
        }
        left += defaultLeftDisplay(response, groupbuckets);
        left += ("</body>");
        left += ("</html>");
      }
      catch(Exception e)
      {
        e.printStackTrace(System.err);
      }
      return left;
    }

    /**
     * Generates an HTML string to display the file navigation pane in the left frame.
     * @author Sandy Pittendrigh
     *
     * @param response the HTTP response object
     * @param groupBuckets a hashtable of objects that contains descriptions of database
  records
     * @return an HTML string responsible for displaying the file navigation pane in the
  left frame
     */
    protected String defaultLeftDisplay(HttpServletResponse response,
                                        Hashtable groupbuckets)
    {
      String left = "";                                      // left frame HTML substring
      try
      {
        Enumeration gb = groupbuckets.elements();
        while(gb.hasMoreElements())                          // 'unwind' the buckets
        {
          GroupBucket groupBucket = (GroupBucket)gb.nextElement();
          String group = groupBucket.getName();
          left += (HtmlGenerator.makeBold(group, "group"));   // lab group heading
          left += (BR);
          Hashtable templateBuckets = groupBucket.getBuckets();
          Enumeration tb = templateBuckets.elements();
          while(tb.hasMoreElements())
          {
            TemplateBucket templateBucket = (TemplateBucket)tb.nextElement();
            String template = templateBucket.getName();
            left += (BR);
            left += (HtmlGenerator.makeBold(template, "template"));          // record
  template heading
            left += (BR);
            Hashtable ownerbuckets = templateBucket.getBuckets();
            Enumeration ob = ownerbuckets.elements();
            while(ob.hasMoreElements())
            {
              OwnerBucket ownerBucket = (OwnerBucket)ob.nextElement();
              left += indent(1);
              left += (HtmlGenerator.makeBold(ownerBucket.getName(), "user"));      //
  record owner heading
              left += (BR);
              DescriptionBucket descriptionBucket = (DescriptionBucket)ownerBucket.
                  getBucket(template);
              Vector descriptions = descriptionBucket.getDescriptions();
              int size = descriptions.size();
              for(int i = 0; i < size; i++)
              {
                Description description = (Description)descriptions.elementAt(i);
                String target = "rightframe";
                String docID = description.getName();
                String link = selfURL + "?mode=pullFile";
                link += AMP + "docID=" + docID + AMP + "freshFromDB=yes";
                left += indent(2);
                String label = description.getName();
                String outlink = HtmlGenerator.makeHyperLink(link, target, label);
                left += (HtmlGenerator.makeBold(outlink, "label"));          // hyperlinked
  record name
                left += (BR);
              }
            }
          }
          left += (" ");
        }
      }
      catch(Exception e)
      {
        e.printStackTrace(System.err);
      }
      return left;
    }

    /**
     * Places a Description(of file) object into the DescriptionBucket's list of files
     * for a particular template.
     * @author Sandy Pittendrigh
     *
     * @param description a Description object that contains file information
     * @param description_bucket a DescriptionBucket object that groups file descriptions
     *       under template heading
     */
    protected void description2DescriptionBucket(Description description,
                                                 DescriptionBucket description_bucket)
    {
      description_bucket.putBucket(description);
    }

    /**
     * Places a DescriptionBucket object into the OwnerBucket's hashtable of
     * DescriptionBuckets (keyed by template of the DescriptionBucket).
     * @author Sandy Pittendrigh
     *
     * @param description_bucket a DescriptionBucket object that contains list of file
  descriptions
     * @param owner_bucket a OwnerBucket object that groups DescriptionBuckets (keyed by
  template)
     */
    protected void descriptionBucket2OwnerBucket(DescriptionBucket description_bucket,
                                                 OwnerBucket owner_bucket)
    {
      owner_bucket.putBucket(description_bucket.getName(), description_bucket);
    }

    /**
     * Places a OwnerBucket object into the TemplateBucket's hashtable of
     * OwnerBuckets (keyed by owner of OwnerBucket object).
     * @author Sandy Pittendrigh
     *
     * @param owner_bucket an OwnerBucket object that groups DescriptionBuckets (keyed by
  template)
     * @param template_bucket a TemplateBucket object that groups OwnerBuckets
     *        (keyed by owner of OwnerBucket object)
     */
    protected void ownerBucket2TemplateBucket(OwnerBucket owner_bucket,
                                              TemplateBucket template_bucket)
    {
      template_bucket.putBucket(owner_bucket.getName(), owner_bucket);
    }

    /**
     * Places a TemplateBucket object into the GrouptBucket's hashtable of
     * TemplateBuckets (keyed by name of TemplateBucket object).
     * @author Sandy Pittendrigh
     *
     * @param template_bucket a TemplateBucket object that groups OwnerBuckets
     *        (keyed by owner of OwnerBucket object)
     * @param group_bucket a GroupBucket object that groups TemplateBuckets
     *        (keyed by name of TemplateBucket object)
     */
    protected void templateBucket2GroupBucket(TemplateBucket template_bucket,
                                              GroupBucket group_bucket)
    {
      group_bucket.putBucket(template_bucket.getName(), template_bucket);
    }


    /**
```

```
   * Returns an HTML string that represents the 'Show All Records'
   * and 'Show Your Records' hyperlinks.
   *
   * @param request the HTTP request object
   * @return HTML string that represents two hyperlinks
   */
  protected String displayHyperLinks(HttpServletRequest request)
  {
    String link = "";                                       // the hyperlinks
    try
    {
        // XPath query string to display all files
      String xpath = "/EML/HEAD";
      String src = selfURL + "?mode=defaultLeft" + AMP + "xpath=" + xpath;
      link += HtmlGenerator.makeHyperLink(src, "leftframe", "Show All Records");
      link += SPC;
      link += "||";
      link += SPC;
      HttpSession session = request.getSession(false);
      String user_name = (String)session.getAttribute("user_name");
        // XPath query string to display owner's files only
      xpath = "/EML[HEAD/OWNER[text()='" + user_name + "']]/HEAD";
      src = selfURL + "?mode=defaultLeft" + AMP + "xpath=" + xpath;
      link += HtmlGenerator.makeHyperLink(src, "leftframe", "Show Your Records");
    }
    catch(Exception e)
    {
      e.printStackTrace(System.err);
    }
    return link;
  }

  /**
   * Generates a generic HTML page displaying the given string.
   *
   * @param message the string to display
   * @return HTML string that generates the page displaying the given string
   */
  protected String messagePage(String message)
  {
    String str = "";                                        // the HTML string
    try
    {
      str += (HtmlGenerator.makeVersionInfo());                    // version element
      str += (HtmlGenerator.makeDocType("loose"));          // DOCTYPE element
      str += ("<html>");
      str += ("<head>");
      str += ("<link rel=StyleSheet href=\"http://wage.cns.montana.edu/styles/lib.css\"
type=\"text/css\">");
      /*  //test stylesheet format
       str += (HtmlGenerator.makeLink("StyleSheet",
                                "http://localhost:8080/stylesheets/pages.css",
                                "text/css"));
       */
      str += ("<title>");
      str += ("</title>");
      str += ("</head>");
      str += ("<body>");
      str += ("<center>");
      str += ("<b class=\"group\">");
      str += (message);                                            // the string to
display
      str += ("</b>");
      str += ("</center>");
      str += ("</body>");
      str += ("</html>");
    }
    catch(Exception e)
    {
      e.printStackTrace(System.err);
    }
    return str;
  }
```

```
/**
    * Updates a DataEntryModel tree according to POST parameters.
    *
    * @param request the HTTP request object
    * @param liveTree the DataEntryModel object to update
    */

   protected void fixWholeTree(HttpServletRequest request, DataEntryModel liveTree)
   {
     String postParameter;                                  // the parameter
     String currentValue = "";                                      // node's current
value
     HttpSession session = request.getSession(false);
     String currentUser = (String)session.getAttribute("user_name");      // current
user
     GUINode bodyNode = liveTree.getNode("/EML/BODY/");             // the <BODY> element
     Enumeration enum = bodyNode.preorderEnumeration();
     while(enum.hasMoreElements())                          // loop children (preorder
traversal)
     {
       postParameter = "";
       GUINode myNode = (GUINode)enum.nextElement();
       String nodeStatus = myNode.getStatus();
       if(nodeStatus.equals(DataNode.WIDGET))                       // if its a widget
       {
         String widgetType = myNode.getWidgetType();        // widget type
         String xPath = myNode.getXPath();                  // unique xpath of widget
         String parameterArray[] = request.getParameterValues(xPath);             //
parameter array
         String pullDownParameter = request.getParameter(xPath + "_pull_down");   // new
pull down option
         if(parameterArray != null)
         {
           postParameter = parameterArray[0];
         }
         if(myNode.getValue() != null)
         {
           currentValue = myNode.getValue();
         }
           // if a new option is entered for a pull down menu, use new option
         if(pullDownParameter != null && !pullDownParameter.equals("Enter New Option") &&
            !pullDownParameter.equals(""))
         {
           myNode.setValue(pullDownParameter);
             // add the new option to the list of items
           addPullDownMenuItem(myNode, pullDownParameter, currentUser);
         }
         else if(widgetType.equals("CheckBoxWidget"))                             // if
check box
         {
           fixCheckBoxWidget(myNode, currentValue, postParameter);        // update node
         }
         else if(widgetType.equals("UrlWidget"))                          // if URL
widget
         {
           fixUrlWidget(myNode, request, currentValue, currentUser);              //
update node
         }
           // handle default case for most widgets
         else if(!currentValue.equals(postParameter))
         {
           myNode.setValue(postParameter);
         }
       }  // end if widget
     }    // end while
   }

   /**
    * Updates a UrlWidget according to POST parameters.
    *
    * @param toFix the GUINode to update
    * @param request the HTTP request object
    * @param value the current value of the UrlWidget
    * @param currentUser the user currently logged in
    */
   protected void fixUrlWidget(GUINode node, HttpServletRequest request, String value,
String currentUser)
   {
       String xPath = node.getXPath();
         // get various parameters related to a UrlWidget
       String protocolPostParameter = request.getParameter( xPath + "_protocol" );
       String hostPostParameter = request.getParameter( xPath + "_host" );
       String pathPostParameter = request.getParameter( xPath + "_path" );
       String hostPullDownParameter = request.getParameter( xPath + "_host_pull_down" );
       String pathPullDownParameter = request.getParameter( xPath + "_path_pull_down" );
       String fileNameParameter = request.getParameter(xPath + "_file_name");

       String urlInfo[] = parseUrlString(value);           // parse URL into component
parts
       String currentProtocol = (urlInfo[0]);
       String currentHost = (urlInfo[1]);
       String currentPath = (urlInfo[2]);
       String currentFileName = (urlInfo[3]);

       String newProtocol = currentProtocol;               // initialize new values with
current ones
       String newHost = currentHost;
       String newPath = currentPath;
       String newFile = currentFileName;

       boolean protocolChanged = false;                            // flags for certain
parameters
       boolean hostChanged = false;
       boolean pathChanged = false;
       boolean fileNameChanged = false;
       boolean isHostPullDown = false;
       boolean isPathPullDown = false;

       GUINode protocolNode = (GUINode)node.getChildAt(0);  // get sub-nodes
       GUINode hostNode = (GUINode)node.getChildAt(1);
       GUINode pathNode = (GUINode)node.getChildAt(2);

       if(protocolPostParameter != null && !protocolPostParameter
          .equalsIgnoreCase(currentProtocol))                      // check POST
parameters and set flags
       {
         protocolChanged = true;
       }
       if(hostPostParameter != null && !hostPostParameter
          .equalsIgnoreCase(currentHost))
       {
         hostChanged = true;
       }
       if(pathPostParameter != null && !pathPostParameter
          .equalsIgnoreCase(currentPath))
       {
         pathChanged = true;
       }
       if ( hostPullDownParameter != null && !hostPullDownParameter.equals( "" )
            && !hostPullDownParameter.equals( "Enter New Host" ) )
       {
         isHostPullDown = true;
           // remove any slashes from front or back of string
         hostPullDownParameter = removeLeadingSlashes(hostPullDownParameter);
         hostPullDownParameter = removeTrailingSlashes(hostPullDownParameter);
       }
       if ( pathPullDownParameter != null
           && !pathPullDownParameter.equals( "Enter New Path" ) )
       {
         isPathPullDown = true;
           // remove any slashes from front or back of string
         pathPullDownParameter = removeLeadingSlashes(pathPullDownParameter);
         pathPullDownParameter = removeTrailingSlashes(pathPullDownParameter);

       }
       if(fileNameParameter != null && !fileNameParameter.equals(currentFileName))
       {
         fileNameChanged = true;
           // remove any slashes from front or back of string
         fileNameParameter = removeLeadingSlashes(fileNameParameter);
         fileNameParameter = removeTrailingSlashes(fileNameParameter);
       }
      if(protocolChanged)
      {
        newProtocol = protocolPostParameter;
      }

        // if a value entered in the textfield, trump a new value in the dropdown box
      if(isHostPullDown)
      {
          // alter the host value
        newHost = hostPullDownParameter;
```

```
          // if current host or new host not in current list of hosts
          // make new host menu item(s) and add it the host node
        addUrlMenuItem( hostNode, hostPullDownParameter, currentUser);
        addUrlMenuItem( hostNode, currentHost, currentUser);
      }
      else if(hostChanged)
      {
        newHost = hostPostParameter;
        addUrlMenuItem( hostNode, hostPostParameter, currentUser);
        addUrlMenuItem( hostNode, currentHost, currentUser);
      }

        // if a value entered in the textfield, trump a new value in the dropdown box
      if(isPathPullDown)
      {
          // alter the path value
        newPath = pathPullDownParameter;
          // if current path or new path not in current list of paths
          // make new path menu item(s) and add it the path node
        addUrlMenuItem(pathNode, pathPullDownParameter, currentUser);
        addUrlMenuItem(pathNode, currentPath, currentUser);
      }
      else if(pathChanged)
      {
        newPath = pathPostParameter;
        addUrlMenuItem(pathNode, pathPostParameter, currentUser);
        addUrlMenuItem(pathNode, currentPath, currentUser);
      }
      if(fileNameChanged)
      {
        newFile = fileNameParameter;
      }
        // compose new hyperlink
      String newHyperLinkValue = newProtocol + "://" + newHost + "/" + newPath + "/" +
          newFile;
      node.setValue(newHyperLinkValue);                            // set node's value
    }

    /**
     * Adds an item to a UrlWidget menu.
     *
     * @param myNode the GUINode to update
     * @param newItem the item to add
     * @param owner the user adding the item
     */
    protected void addUrlMenuItem(GUINode myNode, String newItem, String owner)
    {
      Vector items = myNode.getChildrensValues();           // list of children's values
      if(!newItem.equals("") && !items.contains(newItem))
      {
        GUINode tempItem = StringServletClient.mkGUINodeMenuitem(newItem, owner);
        myNode.add(tempItem);
      }
    }

   /**
    * Updates a CheckBoxWidget according to POST parameters.
    *
    * @param toFix the GUINode to update
    * @param currentValue the current value of the GUINode
    * @param postParameter the POST parameter value for a given GUINode
    */
   protected void fixCheckBoxWidget(GUINode toFix, String currentValue, String
 postParameter)
   {
       // if value true and corresponding parameter is empty string => set to false
     if(currentValue.equals("true") && postParameter.equals(""))
     {
       toFix.setValue("false");
     }
       // if value false and corresponding parameter is true => set to true
     else if(currentValue.equals("false") && postParameter.equals("true"))
     {
       toFix.setValue("true");
     }
   }
```

```
/**
   * Tests a PullDownWidget node for an item and if not present
   * adds the item as child of the node.
   *
   * @param myNode the PullDownWidget node
   * @param pullDownParameter the item we are looking for
   * @param owner the owner of the PullDownWidget
   */
  protected void addPullDownMenuItem(GUINode myNode, String pullDownParameter, String
owner)
  {
      // this is the <MENUITEMS> node
    GUINode menuItems = myNode.getChild("MENUITEMS");
      // test it's children (the <ITEM>s) for the new value
    Vector childValues = menuItems.getChildrensValues();
      //if it's not in the list of items, make a new GUINode and attach it to parent
    if(!childValues.contains(pullDownParameter))
    {
      GUINode newNode = StringServletClient.mkGUINodeMenuitem(pullDownParameter, owner);
      menuItems.add(newNode);
    }
  }

  /**
   * Shows all attributes and values for each GUINode object in a
   * DataEntryModel tree.  Shows all POST parameters in a client
   * HttpServletRequest object.
   *
   * @param request the HTTP request object
   * @param liveTree the DataEntryModel object to inspect
   * @return a string composed of all attributes/values and POST parameters
   */
  protected String debugTree(HttpServletRequest request, DataEntryModel liveTree)
  {
    String str = "";                                        // debug information
    str += "<pre>";
    str += liveTree.showTree();                                     // attributes/values
of tree
    str += "\n\n\n";
    str += postParams(request);                                     // POST parameters
    str += "</pre>";
    return str;
  }

  /**
   * Builds the form that displays a database record.
   *
   * @param request the HTTP request object
   * @param rootNode the DataEntryModel object(record) to display in the form
   * @param userName the user currently logged in
   * @return a string representing an HTML form
   */
  protected String makeDisplay(HttpServletRequest request, DataEntryModel rootNode,
                              String userName)
  {
    GUINode docIDNode = rootNode.getNode("/EML/HEAD/DOC_ID/");
    String docID = docIDNode.getValue();                    // document ID number
    String form = "";
      // begin building the form
    form += (HtmlGenerator.makeVersionInfo());                      // version element
    form += "\n";
    form += (HtmlGenerator.makeDocType("loose"));           // DOCTYPE element
    form += "\n";
    form += ("<html>");
    form += "\n";
    form += ("<head>");
    form += "\n";
    form += (HtmlGenerator.makeLink("StyleSheet",
                               "http://localhost:8080/stylesheets/noSuchFile.css",
                               "text/css"));
    /*  test stylesheet format
    form += (HtmlGenerator.makeLink("StyleSheet",
                                   "http://localhost:8080/stylesheets/pages.css",
                                   "text/css"));
    */
    form += "\n";
      // reference to external javascript file that refreshes a particular frame
    form += (HtmlGenerator.makeScript("javascript", "text/javascript",
                                     "http://localhost:8080/scripts/postform.js"));
    form += "\n";
    form += ("<title>");
    form += "\n";
    form += ("</title>");
    form += "\n";
    form += ("</head>");
    form += "\n";
    form += ("<body>");
    form += "\n";
    if(debug.equals("true"))
    {
      form += showDebuggingInfo(request);
    }
    else
    {
      trackSessionCount(request);
    }
    form += ("<center>");
    form += "\n";
    form += HtmlGenerator.makeBold(docID, "group");         // print document ID at top
of form
    form += "\n";
    form += ("</center>");
    form += "\n";
    form += (BR);
    form += "\n";
    String src = selfURL + "?mode=pullFile" + AMP + "docID=" + docID + AMP +
        "freshFromDB=no";
    form += HtmlGenerator.makeOpenForm(src, "rightframe", "POST"); // make opening form
element
    form += "\n";
    GUINode bodyNode = rootNode.getNode("/EML/BODY/");                     //get <BODY>
node and recurse on it
    if(bodyNode.getChildCount() > 0)
    {
      form += makeDisplayHelper(bodyNode, docID);                  // bulk of form built
here
    }
    boolean isOwner = isOwner(userName, rootNode);
    if(isOwner)                                                            // 'Save'
button if file owner
    {
      form += HtmlGenerator.makeButton("save", "Save");
      form += "\n";
      form += BR;
      form += "\n";
    }
    form += BR;
    form += "\n";
    form += HtmlGenerator.makeButton("save_as", "Save As");        // 'Save As' button
for all users
    form += "\n";
    if(isOwner)
    {
      form += BR;
      form += "\n";
      form += BR;
      form += "\n";
      form += HtmlGenerator.makeButton("delete", "Delete");        // 'Delete' button if
file owner
      form += "\n";
    }

    form += "</form>";
    form += "\n";
    form += "</body>";
    form += "\n";
    form += "</html>";
    return form;
  }

  /**
   * Converts a GUINode object to an HTML element.
   *
   * @param node a GUINode object to display
   * @param docID the docment ID of the record
   * @return string representing the GUINode as HTML
   */
  protected String makeDisplayHelper(GUINode node, String docID)
  {
    String form = "";                                       // form elements
    if(node == null)
    {
```

```
    // base case -> empty string will be returned
  }
  else
  {
    form += nodeToHtml(node, docID);                             // add HTML elements
  }
  return form;
}

/**
 * Accepts GUINode object and generates HTML string that
 * represents a text field.
 *
 * @param node a GUINode object that is a text field
 * @return HTML string representing a text field
 */
protected String makeTextFieldWidget(GUINode node)
{
  String str = "";
  String value = node.getValue();                         // node's value
  String xPath = node.getPathString();                    // node's XPath
  str += HtmlGenerator.makeTextField(xPath, value, 30);   // make text field
  return str;
}

/**
 * Accepts GUINode object and generates HTML string that
 * represents a text area.
 *
 * @param node a GUINode object that is a text area
 * @return HTML string representing a text area
 */
protected String makeTextAreaWidget(GUINode node)
{
  String str = "";
  String value = node.getValue();                         // node's value
  String xPath = node.getPathString();                    // node's XPath
  str += HtmlGenerator.makeTextArea(xPath, 5, 30, value); // make text area
  return str;
}

/**
 * Accepts GUINode object and generates HTML string that
 * represents a page/heading banner.
 *
 * @param node a GUINode object that is a banner
 * @return HTML string representing a banner
 */
protected String makeBannerWidget(GUINode node)
{
  String str = "";
  String label = node.getLabel();                         // node's label
  str += HtmlGenerator.makeBold(label, "banner");         // make banner
  return str;
}

/**
 * Accepts GUINode object and generates HTML string that
 * represents a check box.
 *
 * @param node a GUINode object that is a check box
 * @return HTML string representing a check box
 */
protected String makeCheckBoxWidget(GUINode node)
{
  String str = "";
  String value = node.getValue();                         // node's value
  String xPath = node.getPathString();                    // node's XPath
  boolean checked = false;                                // flag for check or not
  if(value != null && value.equals("true"))
  {
    checked = true;
  }
  str += HtmlGenerator.makeCheckBox(xPath, checked);             // make check box
  return str;
}
```

```
/**
   * Accepts GUINode object and generates HTML string that
   * represents a pull down menu.
   *
   * @param node a GUINode object that is a pull down menu
   * @return HTML string representing a pull down menu
   */
  protected String makePullDownMenu(GUINode node)
  {
    String str = "";
    String value = node.getValue();                         // node's value
    String xPath = node.getPathString();                    // node's XPath
    if(node.getChildCount() > 0)
    {
      GUINode menuItems = (GUINode)node.getChildAt(0);             // the <MENUITEMS>
node
      if(menuItems.getChildCount() > 0)
      {
        Vector items = menuItems.getChildrensValues();             // the <ITEM...> nodes
          // make the pull down box
        str += HtmlGenerator.makePullDownBox(xPath, value, items);
      }
    }
    return str;
  }

  /**
   * Accepts GUINode object and generates HTML string that
   * represents a PullDownWidget.  Displays a text field to
   * right of the pull down menu for adding new options.
   *
   * @param node a GUINode object that is a PullDownWidget
   * @return HTML string representing a PullDownWidget
   */
  protected String makePullDownWidget(GUINode node)
  {
    String str = "";
    String xPath = node.getPathString();                    // node's XPath
    str += makePullDownMenu(node);                          // make pull down
    str += "\n";
    str += SPC;
    str += "\n";
      // make a textfield so new option may be entered
    str += HtmlGenerator.makeTextField(xPath + "_pull_down", "Enter New Option", 30);
    return str;
  }

  /**
   * Accepts GUINode object and generates HTML string that
   * represents a pull down menu for a ScreenSelectWidget.
   *
   * @param node a GUINode object that is a ScreenSelectWidget
   * @return HTML string representing a pull down menu for a ScreenSelectWidget
   */
  protected String makeScreenSelectPullDownMenu(GUINode node, String docID)
  {
    String str = "";
    String value = node.getValue();                         // node's value
    String xPath = node.getPathString();                    // node's XPath
    str += "\n";
    str += SPC;
    str += "\n";
    Vector items = node.getChildrensLabels();                      // make pull down menu
    str += HtmlGenerator.makeSSelectPullDownBox(xPath, value, items, selfURL, docID);
    str += "\n";
    return str;
  }

  /**
   * Accepts GUINode object and generates HTML string that
   * represents a pull down menu with no choice to add a new option.
   *
   * @param node a GUINode object that is a pull down menu
   * @return HTML string representing a pull down menu
   */
  protected String makeMustChooseWidget(GUINode node)
  {
    String str = "";
    str += makePullDownMenu(node);                          // make pull down menu
    return str;
  }
  /**
   * Accepts GUINode object and generates HTML string that
   * represents a UrlWidget.
   *
   * @param node a GUINode object that is a UrlWidget
   * @return HTML string representing a UrlWidget
   */
  protected String makeUrlWidget(GUINode node)
  {
    String str = "";                                       // URL widget
    String value = node.getValue();                        // node's value(a URL)
    if(value != null)
    {
      String xPath = node.getPathString();                 // node's XPath
      String urlInfo[] = parseUrlString(value);                    // parse URL into
component parts
      String currentProtocol = (urlInfo[0]);
      String currentHost = (urlInfo[1]);
      String currentPath = (urlInfo[2]);
      String currentFileName = (urlInfo[3]);
      if(node.getChildCount() > 0)
      {
        GUINode protocolNode = (GUINode)node.getChildAt(0); // access subnodes and get
childrens' values
        GUINode hostNode = (GUINode)node.getChildAt(1);
        GUINode pathNode = (GUINode)node.getChildAt(2);
        Vector protocolItems = protocolNode.getChildrensValues();
        Vector hostItems = hostNode.getChildrensValues();
        Vector pathItems = pathNode.getChildrensValues();
        hostItems.remove("Enter New Host");
        pathItems.remove("Enter New Path");
        String hyperLink = "";
        str += SPC;
        str += "\n";
          // compose "new" hyperlink
        String newHyperLinkValue = currentProtocol + "://" + currentHost + "/" +
            currentPath;
        if(!currentFileName.equals(""))
        {
          newHyperLinkValue += "/";
          newHyperLinkValue += currentFileName;
        }
        hyperLink = HtmlGenerator.makeHyperLink(newHyperLinkValue, "_blank",
                                        newHyperLinkValue);
        str += (hyperLink);                                // hyperlinked URL
        str += "\n";
        str += (BR);
        str += "\n";
        str += makeOpenFieldset();                         //  start of 'URL Editor"
        str += makeOpenLegend();
        str += (HtmlGenerator.makeBold("URL Editor", "user"));
        str += "\n";
        str += makeCloseLegend();
        str += ("<table border=\"0\">");
        str += "\n";
        str += ("<tr>");
        str += "\n";
        str += ("<td>");
        str += "\n";                                       // make protocol pull down
        str +=
            (HtmlGenerator.makePullDownBox(xPath + "_protocol", currentProtocol,
                                      protocolItems));
        str += "\n";
        str += ("</td>");
        str += "\n";
        str += ("<td>");
        str += "\n";                                       // make host pull down
        str +=
            (HtmlGenerator.makePullDownBox(xPath + "_host", currentHost, hostItems));
        str += "\n";
        str += ("</td>");
        str += "\n";
        str += ("<td>");
        str += "\n";                                       // make path pull down
        str +=
            (HtmlGenerator.makePullDownBox(xPath + "_path", currentPath, pathItems));
        str += "\n";
        str += ("</td>");
        str += "\n";
        str += ("<td>");
        str += "\n";                                        // make text field for file
name
        str += (HtmlGenerator.makeTextField(xPath + "_file_name",
                                    currentFileName, 20));
        str += "\n";
        str += ("</td>");
        str += "\n";
        str += ("</tr>");
        str += "\n";
        str += ("<tr>");
        str += "\n";
        str += ("<td>");
        str += "\n";
        str += ("&nbsp;");
        str += "\n";
        str += ("</td>");
        str += "\n";
        str += ("<td>");
        str += "\n";                                        // make text field for host
        str += (HtmlGenerator.makeTextField(xPath + "_host_pull_down", "Enter New Host",
                                    20));
        str += "\n";
        str += ("</td>");
        str += "\n";
        str += ("<td>");
        str += "\n";                                        // make text field for path
        str += (HtmlGenerator.makeTextField(xPath + "_path_pull_down", "Enter New Path",
                                    20));
        str += "\n";
        str += ("</td>");
        str += "\n";
        str += ("<td>");
        str += "\n";
        str += ("&nbsp;");
        str += "\n";
        str += ("</td>");
        str += "\n";
        str += ("</tr>");
        str += "\n";
        str += ("</table>");
        str += "\n";
        str += ("</fieldset>");
        str += "\n";
        str += (BR);
        str += "\n";
      } // end if node.childCount() > 0
    }   // end if value != null
    return str;
  }

  /**
   * Generates the <fieldset> and <legend> HTML elements that represent
   * a border with an enclosed labe.
   *
   * @param node GUINode object whose label is enclosed in the <legend> tags
   * @return HTML string representing the opening <fieldset> and complete <legend> tags
   */
  protected String startFieldset(GUINode node, String docID)
  {
    String str = "";
    String label = node.getLabel();                         // node's label
    Hashtable attributes = node.getAttributes();            // node's attributes
    String widgetType = (String)attributes.get("GUI_TYPE"); // node's widget type
    str += makeOpenFieldset();                                      // make <fieldset> tag
    str += makeOpenLegend();                                // make <legend> tag
    str += HtmlGenerator.makeBold(label, "group");          // make label
      // make a pull down menu and submit button for ScreenSelect widgets
    if(widgetType.equals("ScreenSelectWidget"))
    {
      str += makeScreenSelectPullDownMenu(node, docID);
      str += "<input type=\"submit\" name=\"screenSelect\" value=\"View New
Selection\">";
    }
    str += makeCloseLegend();                               // make </legend> tag
    return str;
  }
```

```
/**
   * Generates an opening <fieldset> tag.
   *
   * @return HTML string that is an opening <fieldset> tag
   */
  protected String makeOpenFieldset()
  {
    String fieldset = "";
    fieldset += "<fieldset>";
    fieldset += "\n";
    return fieldset;
  }

  /**
   * Generates an closing <fieldset> tag.
   *
   * @return HTML string that is an closing <fieldset> tag
   */
  protected String makeCloseFieldset()
  {
    String fieldset = "";
    fieldset += "</fieldset>";
    fieldset += "\n";
    return fieldset;
  }

  /**
   * Generates an opening <legend> tag.
   *
   * @return HTML string that is an opening <legend> tag
   */
  protected String makeOpenLegend()
  {
    String legend = "";
    legend += "<legend>";
    legend += "\n";
    return legend;
  }

  /**
   * Generates an closing <legend> tag.
   *
   * @return HTML string that is an closing <legend> tag
   */
  protected String makeCloseLegend()
  {
    String legend = "";
    legend += "\n";
    legend += "</legend>";
    legend += "\n";
    return legend;
  }

  /**
   * Generates a string that represents a GUINode as an HTML element.
   *
   * @param node GUINode to convert into HTML element
   * @param docID docment ID of the record
   * @return HTML string that represents the GUINode
   */
  protected String nodeToHtml(GUINode node, String docID)
  {
    String form = "";                                     // the form element
    String label = node.getLabel();                       // node's label
    String status = node.getStatus();                     // node's status
    String value = node.getValue();                       // node's value
    String xPath = node.getPathString();                  // node's XPath
    String widgetType = "";                               // widget type
    Hashtable attributes = node.getAttributes();          // node's attributes
    if(status.equals("widget"))
    {
      widgetType = (String)attributes.get("GUI_TYPE");
      form += node.getHtmlDepth();                        // indenting
        //the internal label so we can return to a particular place in form
      form += "<a name=\"" + xPath + "\"></a>";
        // display a label except for these widgets
      if(!widgetType.equals("PanelWidget") && !widgetType.equals("ScreenSelectWidget")
         && !widgetType.equals("TreeWidget") && !widgetType.equals("TreeNodeWidget")
         && !widgetType.equals("BannerWidget"))
      {
        form += HtmlGenerator.makeBold(label, "group");
        form += "\n";
        form += SPC;
        form += "\n";
      }
      if(widgetType.equals("TextFieldWidget"))                     // dispatch text field
node
      {
        form += makeTextFieldWidget(node);
      }
      else if(widgetType.equals("BannerWidget"))           // dispatch banner node
      {
        form += makeBannerWidget(node);
      }
      else if(widgetType.equals("TextAreaWidget"))         // dispatch text area node
      {
        form += makeTextAreaWidget(node);
      }
      else if(widgetType.equals("CheckBoxWidget"))         // dispatch check box node
      {
        form += makeCheckBoxWidget(node);
      }
      else if(widgetType.equals("PullDownWidget"))         // dispatch pull down node
      {
        form += makePullDownWidget(node);
      }
      else if(widgetType.equals("MustChooseWidget"))       // dispatch MustChooseWidget
      {
        form += makeMustChooseWidget(node);
      }
      else if( (widgetType.equals("PanelWidget") ||
widgetType.equals("ScreenSelectWidget")
              || widgetType.equals("TreeWidget") || widgetType.equals("TreeNodeWidget") )
              && (label != null && !label.equals("BODY")) )
      {
          // create the opening tags/labels for a <fieldset><legend> label .....
        form += startFieldset(node, docID);
      }
      else if(widgetType.equals("UrlWidget"))                      // dispatch UrlWidget
      {
        form += makeUrlWidget(node);
      }
      if(!widgetType.equals("PanelWidget") && !widgetType.equals("ScreenSelectWidget")
         && !widgetType.equals("TreeWidget") && !widgetType.equals("TreeNodeWidget"))
      {
        form += "\n";                                      // add line breaks
        form += BR;
        form += "\n";
        form += BR;
        form += "\n";
      }
    } // end if status == widget
    boolean done = false;
      // loop children of node(regardless of status) and make recursive calls on them
    Enumeration enum = node.children();
    while(enum.hasMoreElements() && !done)
    {
        // get the proper child to display if it is a ScreenSelectWidget
      if(widgetType.equals("ScreenSelectWidget"))
      {
        GUINode selectedNode = node.getChild(value);
        form += makeDisplayHelper(selectedNode, docID);
        done = true;
      }
        // else just grab the next child
      else
      {
        GUINode childNode = (GUINode)enum.nextElement();
        form += makeDisplayHelper(childNode, docID);
      }
    } // end while
      // close the <fieldset> tag for container widgets
    if( (widgetType.equals("PanelWidget") || widgetType.equals("ScreenSelectWidget")
       || widgetType.equals("TreeWidget") || widgetType.equals("TreeNodeWidget") )
       && (label != null && !label.equals("BODY")) )
    {
      form += makeCloseFieldset();
    }
    return form;
  }
```

```
/**
   * Determines if current user is the owner of a record.
   *
   * @param user_name the user currently logged in
   * @param rootNode a DataEntryModel object(record)
   * @return true if current user is owner, false otherwise
   */
  protected boolean isOwner(String user_name, DataEntryModel rootNode)
  {
    GUINode ownerNode = rootNode.getNode("/EML/HEAD/OWNER/");      // get <OWNER> node
    return user_name.equals(ownerNode.getValue());          // compare values
  }

  /**
   * Generates HTML page that is the top frame of the database interface.
   *
   * @param request the HTTP request object
   * @param response the HTTP response object
   */
  protected void defaultTop(HttpServletRequest request, HttpServletResponse response)
  {
    PrintWriter out;                                        // output stream
    try
    {
      out = response.getWriter();
      out.println(HtmlGenerator.makeVersionInfo());         // version element
      out.println(HtmlGenerator.makeDocType("loose"));             // DOCTYPE element
      out.println("<html>");
      out.println("<head>");
      out.println(
          "<meta http-equiv=\"Content-Type\" content=\"text/html; charset=UTF-8\">");
      out.println("<link rel=StyleSheet
href=\"http://wage.cns.montana.edu/styles/lib.css\" type=\"text/css\">");
      /*
      out.println(HtmlGenerator.makeLink("StyleSheet",
                                 "http://localhost:8080/stylesheets/pages.css",
                                 "text/css"));
       */
      out.println("<title>");
      out.println("</title>");
      out.println("</head>");
      out.println("<body>");
      out.println("<center>");
      out.println("<b class=\"group\">");
      out.println("NeuroSys");                                     // title
      out.println("</b>");
      out.println(BR);
      out.println(BR);
      out.println(displayHyperLinks(request));                     // hyperlinks for
selecting records to view
      out.println("</center>");
      out.println("</body>");
      out.println("</html>");
    }
    catch(Exception e)
    {
      e.printStackTrace(System.err);
    }
  }

  /**
   * Generates string that displays various debugging information
   * concering POST parameters and session variables.
   *
   * @param request the HTTP request object
   */
  protected String showDebuggingInfo(HttpServletRequest request)
  {
    String bug = "";                                        // string of debugging
information
    try
    {
      HttpSession session = request.getSession(false);
      String mode = request.getParameter("mode");           // get various parameters
      String docID = request.getParameter("docID");
      String save = request.getParameter("save");
      String save_as = request.getParameter("save_as");
      String freshFromDB = request.getParameter("freshFromDB");
      String heading;
      bug += ("Version <Enter Version Here>" + BR);
      bug += ("mode : " + mode + BR);
      bug += ("docID : " + docID + BR);
      bug += ("save : " + save + BR);
      bug += ("save_as : " + save_as + BR);
      bug += ("freshFromDB : " + freshFromDB + BR);
      bug += ("Since loading, this servlet has been accessed " + count +
              " times" + BR);
      if(session.isNew())
      {
        heading = "Welcome to Neurosys!";
      }
      else                                          // display session information
      {
        heading = "You have returned to Neurosys: " + session.getAttribute("user_name")
            + BR + "along with your password: " + session.getAttribute("password");
      }
      bug += ("<h5>" + heading + "</h5>");
      Integer cnt = trackSessionCount(request);
      bug += ("You have visited this page " + cnt +
              ((cnt.intValue() == 1) ? " time" : " times" + BR));
      bug += ("<h5>Here is your session data:</h5>" + BR);
      Enumeration enum = session.getAttributeNames();
      while(enum.hasMoreElements())
      {
        String name = (String)enum.nextElement();
        bug += (name + ": " + session.getAttribute(name) + BR);
      }
      bug += ("Session Id: " + session.getId() + BR);
      bug += ("New session: " + session.isNew() + BR);
      bug += ("Creation time: " + session.getCreationTime());
      bug += ("<i>( " + new Date(session.getCreationTime()) + ")</i>" + BR);
      bug += ("Last access time: " + session.getLastAccessedTime());
      bug += ("<i>( " + new Date(session.getLastAccessedTime()) + ")</i>" + BR);
      bug += ("Requested session ID from cookie: " +
              request.isRequestedSessionIdFromCookie() + BR);
      bug += ("Requested session ID from URL: " +
              request.isRequestedSessionIdFromURL() + BR + BR);
      bug += ("<pre>");
      bug += postParams(request);                          // display POST parameters
      bug += ("</pre>");
    }
    catch(Exception e)
    {
      e.printStackTrace(System.err);
    }
    return bug;
  }

  /**
   * Tracks session count.
   * @author Jason Hunter (Java Servlet Programming, O'Reilley and Associates)
   *
   * @param request the HTTP request object
   * @return Integer object representing session count
   */
  protected Integer trackSessionCount(HttpServletRequest request)
  {
    HttpSession session = request.getSession(false);
    Integer cnt = (Integer)session.getAttribute("tracker.count");
    if(cnt == null)
    {
      cnt = new Integer(1);
    }
    else
    {
      cnt = new Integer(cnt.intValue() + 1);
    }
    session.setAttribute("tracker.count", cnt);
    return cnt;
  }

  /**
   * Displays POST parameters.
   * @author Jason Hunter (Java Servlet Programming, O'Reilley and Associates)
   *
   * @param request the HTTP request object
   * @return string of POST parameters
   */
  protected String postParams(HttpServletRequest request)
  {
    String params = "";
    if("application/x-www-form-urlencoded".equals(request.getContentType()))
    {
      Enumeration enum = request.getParameterNames();
      while(enum.hasMoreElements())
      {
        String name = (String)enum.nextElement();
        String values[] = request.getParameterValues(name);
        if(values != null)
        {
          for(int i = 0; i < values.length; i++)
          {
            params += (name + " ( " + i + "): " + values[i]);
            params += BR;
          }
        }
      }
    }
    return params;
  }

  /**
   * Parses a URL string into its protocol, host, path, and file name components.
   *
   * @param url the URL to parse
   * @return an array of strings that represent the protocol, host, path, and file name
   */
  protected String[] parseUrlString(String url)
  {
    String protocol = null;                                 // protocol
    String host = null;                                     // host
    String path = null;                                     // path
    String fileName = null;                                 // name of file
    try
    {
      int colon = url.indexOf(":");                         // use ':' and '/' to parse
URL
      protocol = url.substring(0, colon);
      int firstSingleSlash = url.indexOf("/", colon + 3);
      host = url.substring(colon + 3, firstSingleSlash);
      int lastSingleSlash = url.lastIndexOf("/");
      path = url.substring(firstSingleSlash + 1, lastSingleSlash);
      fileName = url.substring(lastSingleSlash + 1);
    }
    catch(IndexOutOfBoundsException ex)
    {
      log.error(ex.getMessage() + "\n" + emlCommon.util.StackTrace.print(ex));
    }
    catch(NullPointerException ex)
    {
      log.error(ex.getMessage() + "\n" + emlCommon.util.StackTrace.print(ex));
    }

    String urlInfo[] = {protocol, host, path, fileName};    // place components into
array

    return urlInfo;
  }

  /**
   * Removes leading forward slashes from a string.
   *
   * @param toFix the string to remove leading forward slashes from
   * @return the altered string
   */
  protected String removeLeadingSlashes(String toFix)
  {
    String tempString = toFix;
    if(tempString.startsWith("/"))
    {
      do
      {
        tempString = tempString.substring(1);
      }
      while(tempString.startsWith("/"));
    }
    return tempString;
  }
```

```
/**
   * Removes trailing forward slashes from a string.
   *
   * @param toFix the string to remove trailing forward slashes from
   * @return the altered string
   */
  protected String removeTrailingSlashes(String toFix)
  {
    String tempString = toFix;
    if(tempString.endsWith("/"))
    {
      do
      {
        tempString = tempString.substring(0, tempString.length() - 1);
      }
      while(tempString.endsWith("/"));
    }
    return tempString;
  }

  /**
   * Pulls a file from the database.
   *
   * @param serverURL the URL of the database server
   * @param user_name the user currently logged in
   * @param password the password of the current user
   * @param group the lab group to which the user belongs
   * @param docID the document ID of the file to pull
   * @return a DataEntryModel object that represents a file
   */
  protected DataEntryModel getGUITree(String serverURL, String user_name, String
password,
                                      String group, String docID)
  {
    DataEntryModel guiTree = null;
      // instantiate StringServletClient to interface with database
    StringServletClient scc = new StringServletClient(serverURL, user_name,
        MD5.md(password), group);
    guiTree = scc.pullFile(docID);                          // pull the file
    return guiTree;
  }
}
```