Web Application for the Dual Readout Calorimeter Database

Jennifer Karkoska

Hajim School of Engineering & Applied Sciences

University of Rochester

Rochester, NY 14627

Hans Wenzel

Computing Division

Fermilab National Accelerator Laboratory

Batavia, IL 60510

Summer Internship in Science and Technology (SIST) Program

August 9, 2011

Table of Contents

Web Application for a Dual Readout Calorimeter Database

Jennifer Karkoska
University of Rochester
Supervisor: Hans Wenzel
(Dated: August 9, 2011)

ABSTRACT:

The Dual Readout Calorimeter Project hopes to find the best materials to make a Dual Readout Calorimeter, which measures the energy response to both Cherenkov and scintillation light, as accurate as possible. All of the data and plots from the simulations are stored in the Dual Readout Calorimeter Image Database, where every plot can be described by a category and various tag names and values. A Web application allows users to easily find, view, and analyze these plots. Every time a client makes a request, the Web application establishes a connection with the Structured Query Language (SQL) database, which uses prepared statements to quickly return information stored in the database. All of the information the client sees is displayed using JavaServer Pages (JSP), a language based on a combination of Java and HTML. The Web page also incorporates the JavaScript language to increase functionality and user-interactivity. This paper discusses the various programming components that are linked together behind-the-scenes to create the Web application the client actually sees, as well as a guide to using the different features.

1. Introduction:

The Dual Readout Calorimeter project at Fermilab hopes to find the best materials to make a dual readout calorimeter as accurate as possible. A dual readout calorimeter measures the energy response to both Cherekov and scintillation light. The team working on the project uses the Geant4 software to run simulations, and their goal is to make the results from these models as close to the results from actual experiments as possible. In order to keep improving the models, the team must be able to quickly find and compare the hundreds of plots that are produced and stored in a database. Every plot can be identified by a category, such as crystals, various tag names, including type of crystal, sample, and detector used, and specific values for these names, such as BSO or PbF2. As more simulations are run, the database continues to grow to include additional categories, names, values, and plots, making it increasingly difficult to sift through the data. A Java-based Web application allows a user to easily and efficiently find, view, and compare these different plots.

Originally, this Web application allowed a user to choose a plot based only on an automatically generated ID number. This did not make use of the fact that every plot can be described using various user-defined tags. The goal of this project is to allow the user to narrow down the large collection of plots based on categories and tags. The Web application makes efficient use of the database when searching through the plots with the user's specifications. The application also displays the data in a user-friendly manner and allows the client to compare multiple plots at one time.

2. Materials and Methods

2.1. Netbeans

The DRImage Web Application uses the NetBeans Integrated Development Environment (IDE) to easily compile, debug, and run its many components. NetBeans allows the user to import all required libraries, connect to the database, and organize Java, CSS, JavaServer Pages, and all other necessary files. The IDE automatically compiles and builds all of the files, and displays useful messages to aid in debugging. The various components that are used to build the web application are described below.

2.2. SQL and PostgreSQL

SQL, or Structured Query Language, allows a user to access and modify databases. Every record in the database can be inserted, updated, retrieved, and deleted from the database using SQL statements. SQL also accepts queries about the specific data and will return entries that meet specifications [2]. The DR Image Database uses PostgreSQL, an open source object-relational database system, to access its plots and data [9]. All of the data is stored in tables, which can be queried and searched through in a matter of seconds. Every table is identified by a name and contains rows of data. The DR Image Database has tables called “categories,” “tags,” and “images,” each of which has unique identifiers that make up the columns of the table. In the “tags” table, for example, the columns are labeled with “id”, a number for a specific entry in the table, “name”, the type of tag, “value”, a specific value for a tag, and “iid”, the number of the plot with those tags.

In order to access the data in these tables, the DR Image Database uses prepared statements, which are precompiled statements that are sent directly to the database [10]. These statements allow the user to have a great deal of control over which entries are displayed and in what order they appear. This control is the basis for the entire DRImage web application; the user's selections can be entered on the web page and prepared statements will be created on the fly. The prepared statements are sent to the database, a connection is established, and results are returned in seconds. The web application uses an ImageServiceAdapter interface, where every Java method creates a prepared statement to perform a different task.

2.2.1. SQL INSERT INTO

The SQL syntax includes a variety of clauses and operators to access the database. Once a table is defined, entries can be added to the table using the INSERT INTO statement [2]. To insert a new row in the "tags" table, the following SQL statement is used:

```
INSERT INTO tags (id, name, value, iid)
VALUES (63, 'crystal_type', 'BSO', 16)
```

After the data has been inserted into the database, the SELECT FROM statement can retrieve the appropriate information and display the results in a table. The statement

```
SELECT * FROM "public".tags
```

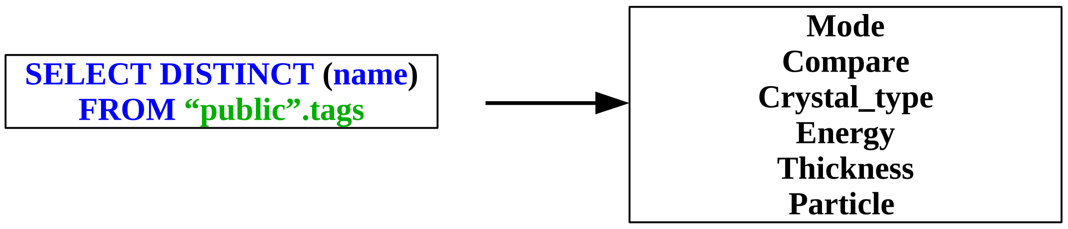
for example, will display a table of all entries from the "tags" table, as can be seen in table 2.1.

id	**name**	**value**	**iid**
64	sample	BSO7_A	16
65	mode	transmission	17
66	compare	vs_sample_10	17
67	crystal_type	BSO	17
68	sample	BSO8_A	17
69	mode	transmission	18
70	compare	vs_sample_10	18
71	crystal_type	BSO	18
72	sample	BSO9_A	18

Table 2.1: Example data from a SELECT FROM SQL statement. This is just a portion of the data that displays in Netbeans when a connection to the DR Image Database is established and the prepared statement "SELECT * FROM "public".tags is used.

2.2.2 SQL SELECT DISTINCT

The SQL syntax also includes a SELECT DISTINCT statement. In the case of duplicate values in a table, the user can specify to select only distinct values, so as not to print out the same value twice [2]. This is particularly useful with the DRImage web application when the user wishes to select a plot by category, tag names, and values. The application displays a check-box list of categories, names, and values, and the SELECT DISTINCT statement ensures that every option will only be printed once (see figure 2.1).



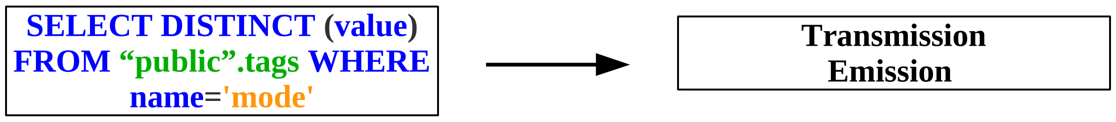
```
SELECT DISTINCT (name)
FROM "public".tags
```

```
Mode
Compare
Crystal_type
Energy
Thickness
Particle
```

Figure 2.1: Example of a SELECT DISTINCT SQL statement. When this statement is used with the DR Image Database, the list of tags on the right displays in Netbeans. The SELECT DISTINCT statement is particularly useful when displaying options for the user to choose from to narrow down the plots in the database.

2.2.3 SQL WHERE Clause

The WHERE clause can be used in a prepared statement to return only records that meet certain criteria [2]. If the user wants to display plots based on the "mode" tag, for example, the application should only display values that are associated with "mode" (figure 2.2). The fact that the application creates these prepared statements on the fly is particularly useful during this process, because the user's selections will change with every use.



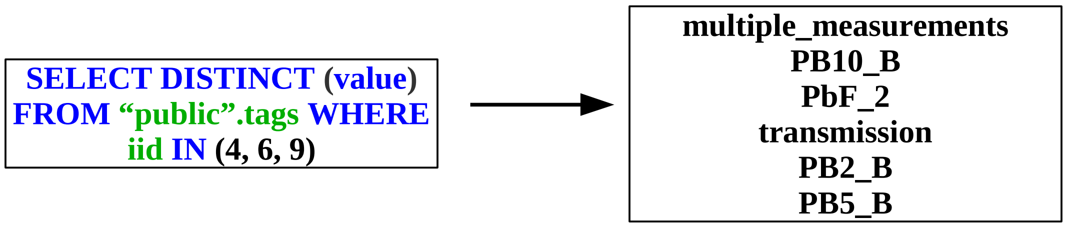
```
SELECT DISTINCT (value)
FROM "public".tags WHERE
name='mode'
```

```
Transmission
Emission
```

Figure 2.2: Example of an SQL WHERE clause. This clause is used whenever the user chooses a category, tags, and values. If the WHERE clause was not used, every single distinct type of value stored in the database.

2.2.4 SQL IN Operator

The IN operator can be used in conjunction with the WHERE clause to specify multiple values in the WHERE clause [2]. Every time a query is made to the "images" table, the plots that meet the specifications are returned as a list of ID numbers to those plots. Before any selections are made, the image list is comprised of every single plot in the database. Instead of having to search through the entire image list after every new request is made, the application searches through the image list that results from the previous query. For example, if the user chooses the "crystals" category, the IDs for all of the plots within the crystals category are saved as a list (see figure 2.3). When the user next chooses one or more tags, the prepared statement queries only the IDs in the list of current images, rather than the entire database. As each selection is made, the list of images decreases in size. By querying this smaller image list, rather than the entire database, the results can be returned more quickly.



```
SELECT DISTINCT (value)
FROM "public".tags WHERE
iid IN (4, 6, 9)
```

```
multiple_measurements
PB10_B
PbF_2
transmission
PB2_B
PB5_B
```

Figure 2.3: Example of the IN SQL operator. Instead of querying the entire entire database of plots, a list of plot Ids is sent to the database. Only the distinct values associated with plots 4, 6, and 9 are returned, allowing for greater efficiency and less complex prepared statements to create.

2.3 The Java Programming Language: Accessing the Database

The Java programming language is used to create prepared statements and access the database via the Web application. An interface defines methods that create prepared statements to query the database and return the desired information. Every method creates a different prepared statement to perform a different task. One method, for example, provides a list of plot ID numbers as a parameter and returns a list of tag names associated with the list. Another method returns a list of distinct categories in the database. There are also methods used to store, retrieve, and delete plots in the database.

2.4 JavaServer Pages (JSP): The Web Application

The Java Programming Language accesses the database, but there also must be a link between Java and the Web application. JavaServer Pages (JSP) provides this link. JSP, developed by Sun Microsystems, is a combination of Java and HTML, where Java processes the information and HTML dictates the actual appearance of the web page. When the client makes a request to the server via the Web application, the server uses a JSP converter to change the JSP file (.jsp) into a Java servlet (.java file). A Java servlet is a program that runs on a Web server and creates Web pages on the fly. This is necessary for the Web application because every page is created based on user input. The servlet is next compiled into a bytecode (.class) file, which can be run on the server and provides direct access to the database [7] (see figure 2.4).

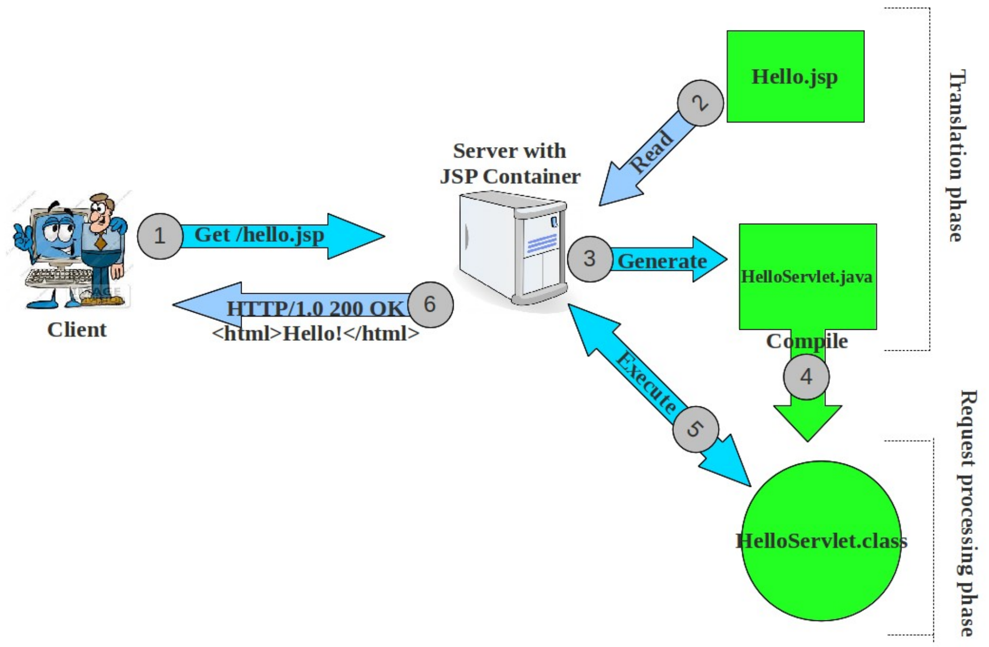


Figure 2.4: JSP page translation and processing phases [1]. This figure provides a graphical representation of the translation of a JSP page into first a servlet (.java) file and then into a bytecode (.class) file.

One challenge that JSP poses is the fact that the Web page reloads every time the user presses a submit button. This resets all of the form fields, or unchecks all boxes and erases all text fields. This makes it difficult to have both previously submitted forms and the next set of selections remain on the same page at the same time. It is helpful for the user to be able to see what he or she selected previously and allow him or her to change these selections, without having to go back to the previous page. Frames are used to maneuver around this problem and allow the user to see the category, tags, and values that he or she selected even after pressing submit. One JSP file, Frameset.jsp, defines the layout for the main selection page and divides the page into three separate frames. Every frame has its source frame, which is either the category, tag, or value frame. Each of these individual frames includes its own JSP page to actually display the data (see figure 2.5).

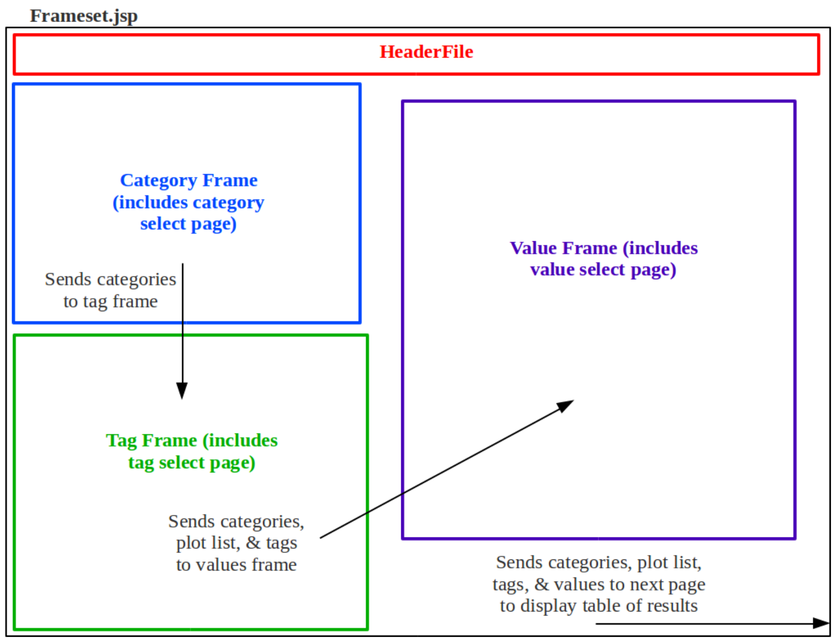

Frameset.jsp
HeaderFile
Category Frame (includes category select page)
Sends categories to tag frame
Tag Frame (includes tag select page)
Sends categories, plot list, & tags to values frame
Value Frame (includes value select page)
Sends categories, plot list, tags, & values to next page to display table of results

Figure 2.5: Overview of the use of frames in the Framset.jsp file. Frames allow multiple JSP pages to display on one Web page. This also lets the user see and change his or her past selections of categories, tags, and values. The header file and category page first appear, then the tag page, and finally the values page. Each new page appears after the user presses the submit button.

At first, only the category page is displayed on the Web page. After the user chooses a category and presses submit, the user's choice(s) are sent to the database. The list of tags associated with the plots in the chosen category will appear on the tag page, which is below the category page. The list of categories with the user's selection will remain on the page, allowing the user to see and modify his or her past choices. The same is true after the user selects from the list of tags and the values list appears.

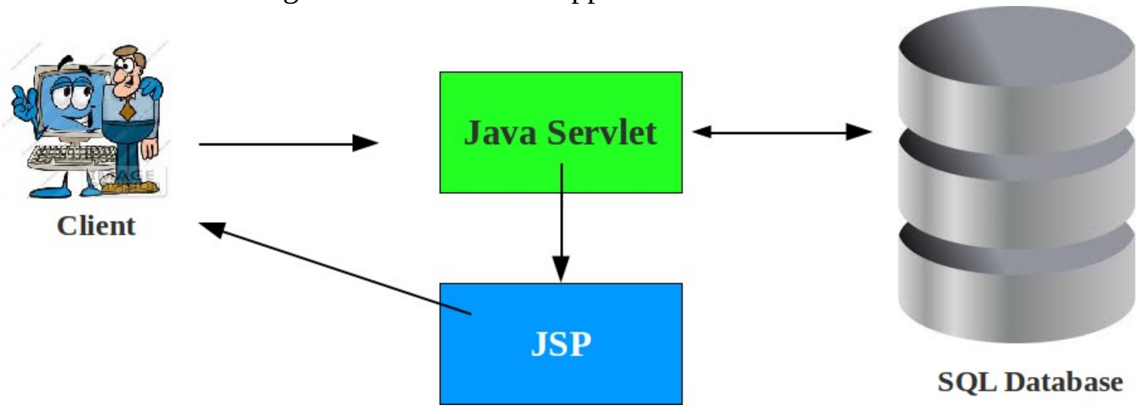


Figure 2.6: Overview of program flow for a servlet-centric application [3]. A Java Servlet bridges the gap between the client's requests and the SQL database. The servlet sends the data back to the JSP pages, which displays the results to the client.

2.5 JavaScript: Making The Web Page More Interactive

JSP is very useful when collecting user data via forms, but it does not provide many other capabilities for user interaction. JavaScript, on the other hand, adds a great deal of interactivity between the user and the Web page. The JavaScript code can be inserted directly into an HTML document using a <script> tag. JavaScript can be used to react to events, such as a mouse click or mouse hover, and can be used to validate forms. Given all of these capabilities, it might seem better to use JavaScript than JSP, but JavaScript has some drawbacks. It acts only on the client side and can therefore only deal with information available in the client's environment; hence, it cannot directly access a database on the server side. If a programmer uses JavaScript in his or her Web page, he or she must also use a server side language, such as JSP, to access the database [5]. The DR Image Web Application uses JSP to allow interaction between the server and client, and JavaScript to make the client-side more interactive.

This interactivity is particularly useful for the table of plots that displays after the user chooses categories, tags, and values. JSP can easily display a static table on the Web page, but this is not helpful for the user if the table contains a large quantity of data. JavaScript introduces a dynamic table with sorting, searching, and pagination features. A simple script tag in the header of the JSP file and the appropriate function calls in the HTML code allows JavaScript to be easily included (figure 2.7).

```
<script type="text/javascript" language="javascript" src="/FILENAME.js"></script>
```

Figure 2.7: An example HTML <script> tag. JavaScript files, functions, and plug-ins can be easily added to any JSP file, simply by adding this line of code with the appropriate JavaScipt file as the source

The JSP page can access the database and return the desired plots to the client, and the client can use the JavaScript functionality to manipulate the data table as he or she likes. The DR Image Web Application also uses JavaScript for tooltips, plot previews, and image zooming.

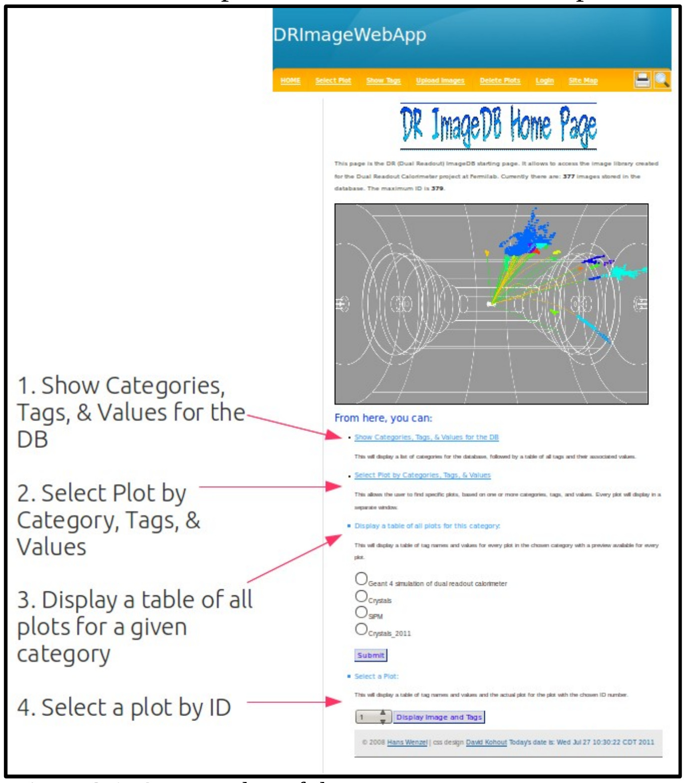
2.6 Cascading Style Sheets (CSS): Actual Web Page Design

A final component of the DR Image Web Application involves the actual design and styling of the Web page. Cascading Style Sheets (CSS) were designed to define how HTML elements are displayed on a Web page without adding long pieces of formatting code to the HTML tags. All of the formatting can be saved in an external .css file, which allows the programmer to edit the one .css file and change the appearance of all the pages that include a reference to it in their headers. CSS can be used to specify page margins, fonts, colors, table styling, image sizes, and everything in between [2].

3. Results

3.1 The Home Page

The DRImageWebApp Home Page can be accessed using any Internet browser by using the following link: http://g4validation.fnal.gov:8080/DRImageWebApp/. The home page provides a short description of the Dual Readout project and lets the user know how many plots are currently stored in the database (figure 3.1). Below the image of a simulation from Geant4, there is a list of options for how the user can proceed, all of which are described below:


DRImageWebApp
HOME
Select Plot
Show Tags
Upload Images
Delete Plots
Login
Site Map
DR ImageDB Home Page
1. Show Categories, Tags, & Values for the DB
2. Select Plot by Category, Tags, & Values
3. Display a table of all plots for a given category
4. Select a plot by ID

This page is the DR (Dual Readout) ImageDB starting page. It allows to access the image library created for the Dual Readout Calorimeter project at Fermilab. Currently there are: **377** images stored in the database. The maximum ID is **379**.

From here, you can:

- Show Categories, Tags, & Values for the DB

  This will display a list of categories for the database, followed by a table of all tags and their associated values.

- Select Plot by Categories, Tags, & Values

  This allows the user to find specific plots, based on one or more categories, tags, and values. Every plot will display in a separate window.

- Display a table of all plots for this category:

  This will display a table of tag names and values for every plot in the chosen category with a preview available for every plot.

  ○ Geant 4 simulation of dual readout calorimeter
  ○ Crystals
  ○ SIPM
  ○ Crystals_2011

  Submit

- Select a Plot:

  This will display a table of tag names and values and the actual plot for the plot with the chosen ID number.

  1 Display Image and Tags

© 2008 Hans Wenzel | css design David Kohout Today's date is: Wed Jul 27 10:30:22 CDT 2011

Figure 3.1: A screenshot of the DR ImageDB Home Page. This is the first page the user sees when he or she browses to http://g4validation.fnal.gov:8080/DRImageWebApp/

3.1.1 *Display Categories, Tags, & Values for the DB*: This page provides a general overview of the plots that are stored in the database. A table displays a list of all the categories that are in the database, followed by a list of tag names and their associated values.

3.1.2 *Select Plot by Category, Tags, & Values*: This page allows a user to find specific plots in the database by narrowing down the choices through a series of steps. The user chooses a category, one or more tags, and one or more values for the chosen tags. A searchable table will display all of the plots that meet those specifications and allow the user to view these plots in separate windows.

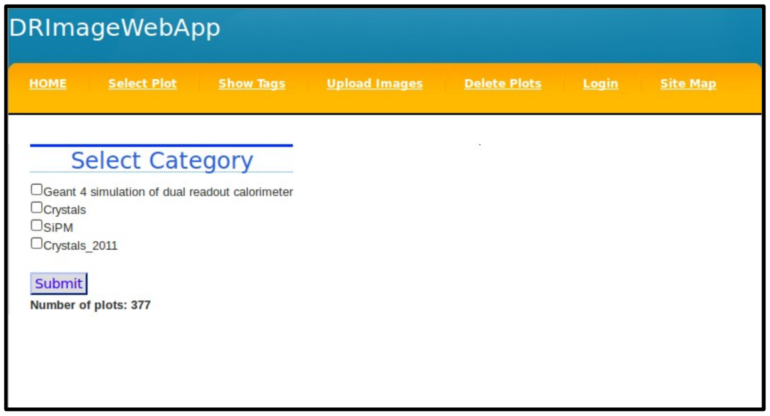
3.1.3 Display a table of all plots for a given category: The user can choose one category and a searchable table will display the tags and values for all of the plots in that category.

3.1.4 *Select a plot by ID*: Originally the only option, this select-box allows a user to choose a plot based on an ID automatically generated by the database. The plot will display in a separate window with a list of tags and values associated with the plot.

3.2 Select Plot by Category, Tags, and Values

The second option (Select Plot by Category, Tags, & Values) is the most useful for the user because it allows the user to find plots that meet specific criteria. The user must make a series of choices, and after each query is submitted, the next set of options will appear on the screen. Details about the selections are as follows:

3.2.1 Select Category



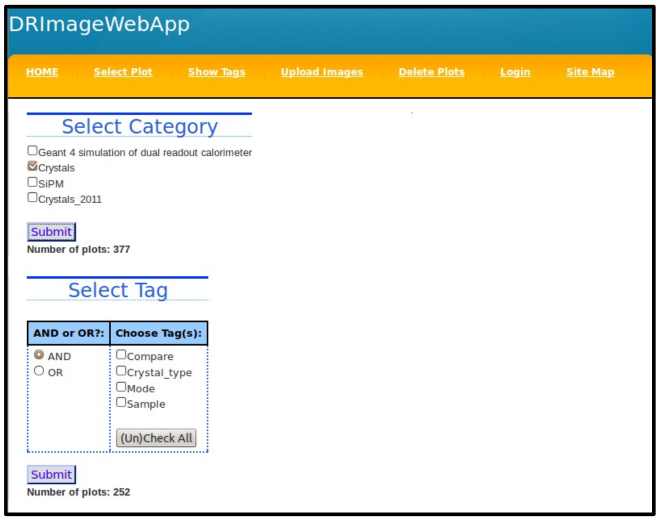
☐Geant 4 simulation of dual readout calorimeter
☐Crystals
☐SiPM
☐Crystals_2011

Number of plots: 377

Figure 3.2: A screenshot of the select category page. When the user navigates to the "Select Plot" link on the header, he or she can first narrow down the plots in the database by choosing a category.

A user can choose one (probably most useful) or more categories of plots to search through. As additional plots with new categories are added to the database, the list of categories will automatically update. The total number of plots is displayed below the submit button, which allows the user to know how many plots are left that meet his or her specifications (figure 3.2).

3.2.2 Select Tags



☐Geant 4 simulation of dual readout calorimeter
☑Crystals
☐SiPM
☐Crystals_2011

Number of plots: 377

AND or OR?:	Choose Tag(s):
◉ AND ○ OR	☐Compare ☐Crystal_type ☐Mode ☐Sample

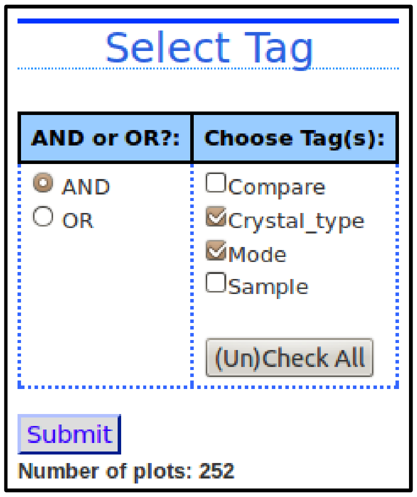
Number of plots: 252

Figure 3.3: A screenshot of the select tag(s) page. After the user chooses a category and presses the submit button, the select tag(s) page will appear, while still displaying the category page.

After the user has chosen a category (in this case, "crystals"), a list of all the tags associated with plots from the chosen category will appear. This list appears below the category select frame, which allows the user to view all of his or her previous selections (figure 3.3).

The user can easily select all of the tags by clicking the "(un)check all" button, or can manually check one or more of the tags.

The user can also specify whether the database should search for only plots that have all of the chosen tags ("AND", the default case), or all plots that have at least one of the tags ("OR"). The total number of plots left that meet the user's specifications is once again displayed.



In this example, there are 252 plots remaining, meaning that the "crystals" category contains 252 plots (figure 3.4).

Figure 3.4: A close-up of the select tag(s) page. A user can choose any number of tags, and whether the plots contain all or at least one of the chosen tags (AND or OR).

3.2.3 Select Values

Once the user chooses the tags (in this example, crystal_type AND mode), a list of all values associated with those tags will appear on the Web page (figure 3.5). The category and tag tables will still display, once again allowing the user to view his or her previous selections.

Just as with the tag selection, the user can check all or some values, and can choose "AND" or "OR" when searching through the database of plots. In the example below, it can be seen that the number of plots is still 252 and hence, every plot in the "crystals" category has at least a "crystal_type" and a "mode" tag (figure 3.6).

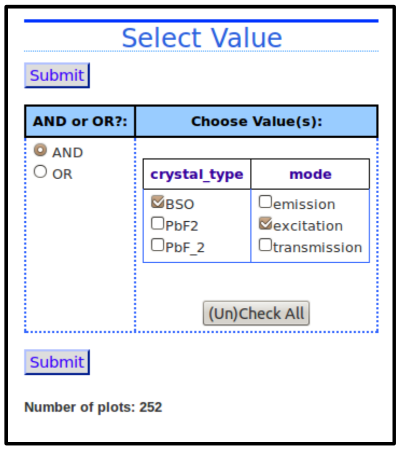


Figure 3.5: A close-up of the select value(s) page. A user can choose any number of values, and whether the plots contain all or at least one of the chosen values (AND or OR).

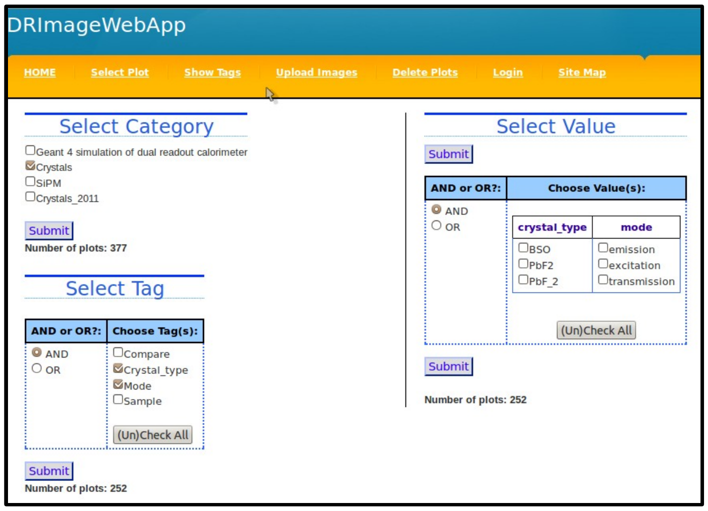
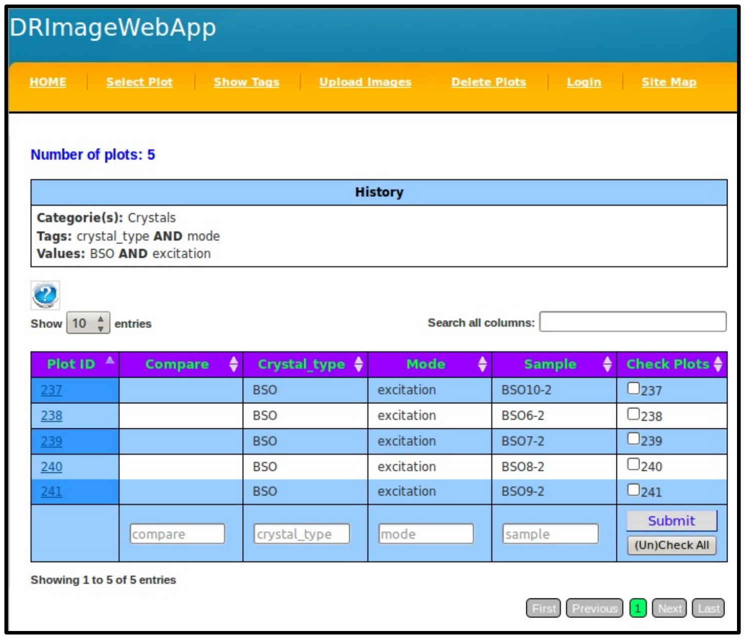


Figure 3.6: A screenshot of the select value(s) page. Because frames are used, the select category and tag(s) pages remain on the screen, even after the submit buttons are pressed. This allows the user to see all of his or her previous selections and make changes without having to reload the page.

3.2.4 Table Of Plots That Meet Specifications



Number of plots: 5

History
Categorie(s): Crystals **Tags:** crystal_type **AND** mode **Values:** BSO **AND** excitation

Show 10 entries Search all columns:

Plot ID	Compare	Crystal_type	Mode	Sample	Check Plots
237		BSO	excitation	BSO10-2	☐237
238		BSO	excitation	BSO6-2	☐238
239		BSO	excitation	BSO7-2	☐239
240		BSO	excitation	BSO8-2	☐240
241		BSO	excitation	BSO9-2	☐241
	compare	crystal_type	mode	sample	Submit (Un)Check All

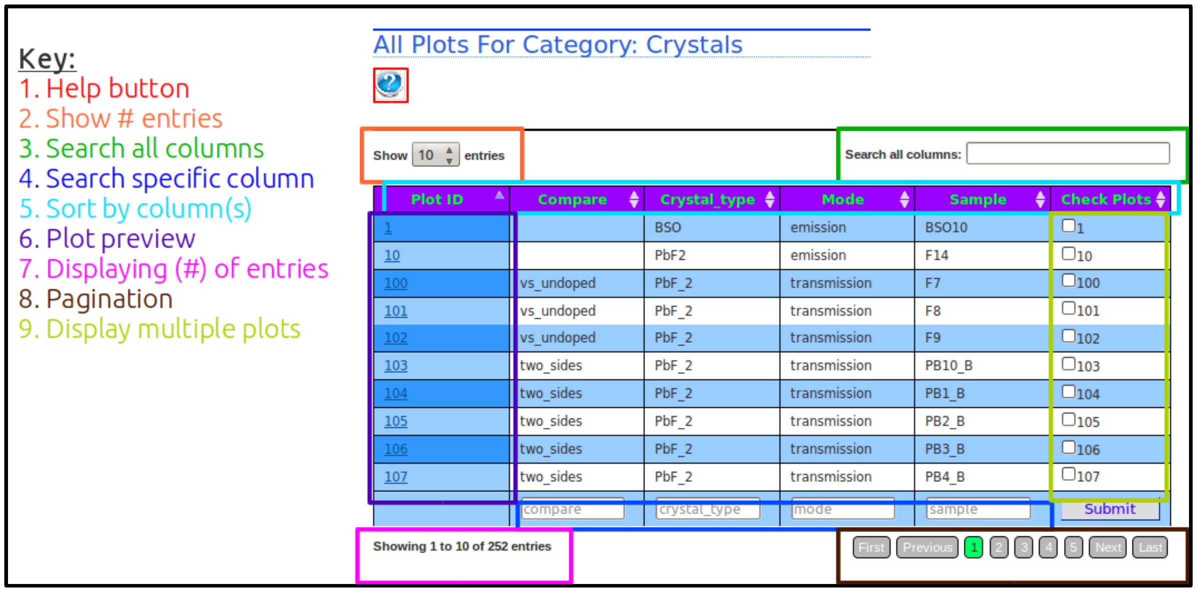
Showing 1 to 5 of 5 entries

Figure 3.7: A screenshot of a table of plots. If a user were to choose the "crystals" category, tags of "crystal_type" AND "mode",and values of "BSO" AND "excitation", the table would display. The total number of plots is displayed, as well as a history of the user's selections.

After the user chooses the specific values and presses the submit button, a new window will appear. This will display the number of plots that meet all of the specifications, a table describing the history of the user's selections (categorie(s), tag(s), and value(s)), and a table of all the tags and values for the remaining plots. The results from the example show that five plots exist in the “crystals” category, which use the crystal type “BSO” and the “excitation” mode (figure 3.7). More specific details about the table appear in the next section.

3.3 Details About Plot Table

The table uses a JavaScript plug-in to make it as user-friendly and interactive as possible. The drawing below highlights the features on a table that displays all of the plots for the “crystals” category (figure 3.8).



Key:
1. Help button
2. Show # entries
3. Search all columns
4. Search specific column
5. Sort by column(s)
6. Plot preview
7. Displaying (#) of entries
8. Pagination
9. Display multiple plots

All Plots For Category: Crystals

Show 10 entries

Search all columns:

Plot ID	Compare	Crystal_type	Mode	Sample	Check Plots
1		BSO	emission	BSO10	☐1
10		PbF2	emission	F14	☐10
100	vs_undoped	PbF_2	transmission	F7	☐100
101	vs_undoped	PbF_2	transmission	F8	☐101
102	vs_undoped	PbF_2	transmission	F9	☐102
103	two_sides	PbF_2	transmission	PB10_B	☐103
104	two_sides	PbF_2	transmission	PB1_B	☐104
105	two_sides	PbF_2	transmission	PB2_B	☐105
106	two_sides	PbF_2	transmission	PB3_B	☐106
107	two_sides	PbF_2	transmission	PB4_B	☐107
	compare	crystal_type	mode	sample	Submit

Showing 1 to 10 of 252 entries

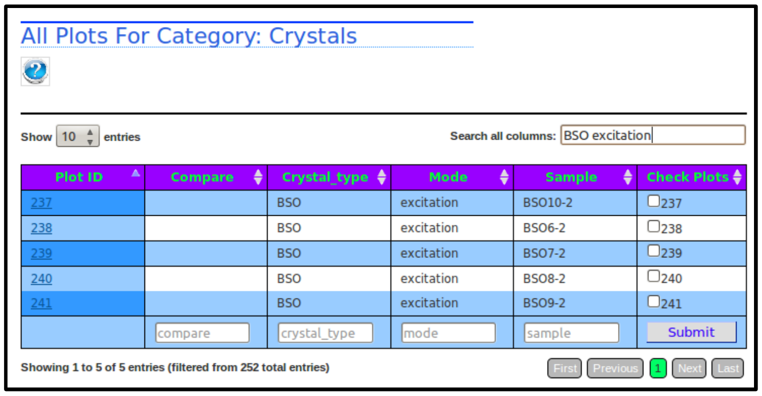
First Previous 1 2 3 4 5 Next Last

Figure 3.8: An example plot table with a key of all the features. The table of plots that meet a user's specifications uses a JavaScript plug-in to make it more user-interactive. The user can sort by column, search through the table, display a certain number of rows per page, and preview an image of each plot.

3.3.1 The Help Button: In order to explain how the table works, the user can press the help button and an informational window will pop-up.

3.3.2 Show # Entries: The user can specify how many entries appear in one page of the table at one time, where the choices are 10, 25, 50, and 100.

3.3.3 Search all columns: If the user wants only plots with specific values to appear in the table, he or she can enter one or more values (separated by spaces) into the “Search all columns” text-field. For example, if he or she types “BSO excitation,” the table will shrink to display only the five plots that have both of those values (figure 3.9).



All Plots For Category: Crystals

Show 10 entries

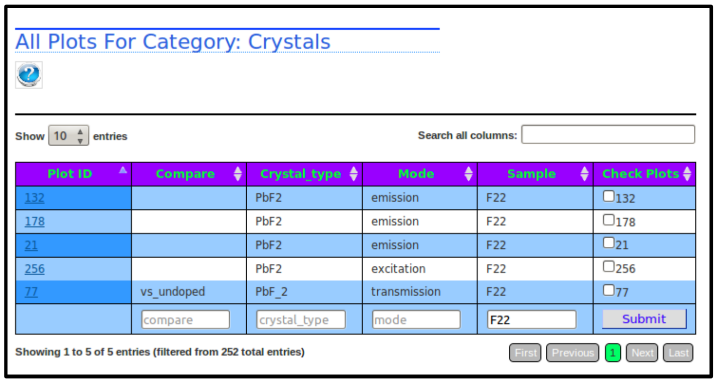
Search all columns: BSO excitation

Plot ID	Compare	Crystal_type	Mode	Sample	Check Plots
237		BSO	excitation	BSO10-2	☐237
238		BSO	excitation	BSO6-2	☐238
239		BSO	excitation	BSO7-2	☐239
240		BSO	excitation	BSO8-2	☐240
241		BSO	excitation	BSO9-2	☐241
	compare	crystal_type	mode	sample	Submit

Showing 1 to 5 of 5 entries (filtered from 252 total entries)

First Previous 1 Next Last

Figure 3.9: An example of the search feature in the plot table. A user can enter one or more values into the “Search all columns” text field to find only plots that have those values. In this example, only plots that use BSO as the crystal type and excitation as the mode are displayed.



All Plots For Category: Crystals

Show 10 entries   Search all columns:

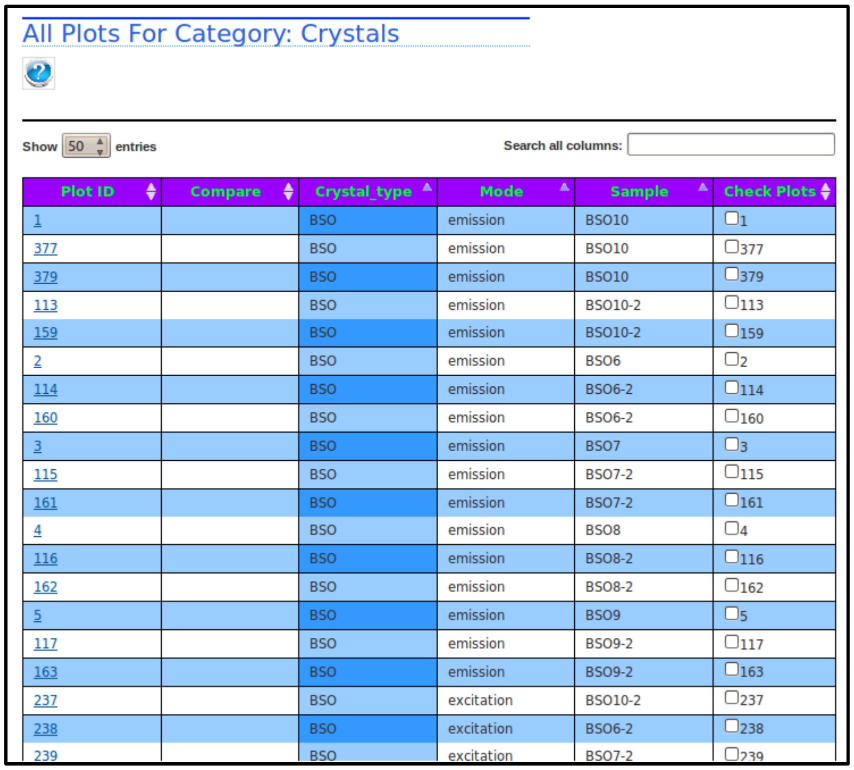
Plot ID	Compare	Crystal_type	Mode	Sample	Check Plots
132		PbF2	emission	F22	☐132
178		PbF2	emission	F22	☐178
21		PbF2	emission	F22	☐21
256		PbF2	excitation	F22	☐256
77	vs_undoped	PbF_2	transmission	F22	☐77
	compare	crystal_type	mode	F22	Submit

Showing 1 to 5 of 5 entries (filtered from 252 total entries)   First Previous 1 Next Last

Figure 3.10: An example of the "search by column" feature in the plot table. A user can search a single column for a specific value, where in this case, only plots that use F22 as a sample are displayed.

3.3.4 *Search specific column:* The user can also search for a value in a specific column by entering a value in the appropriate text-field at the bottom of the table. For example, if the user enters "F22" into the "sample" text-field, the table will display only the five plots that use F_{22} (figure 3.10).

3.3.5 *Sort by column(s):* The column headers for the table are all of the tag names associated with the set of plots being viewed. By default, the plots are displayed in the table based on their automatically generated ID number. The user can click on a column header to sort the table by a specific tag, either in ascending or descending order. If the user holds down the shift button, he or she can sort by multiple columns. In the example to the right, the table is sorted in ascending order first by crystal type, followed by mode, and finally by sample (figure 3.11).

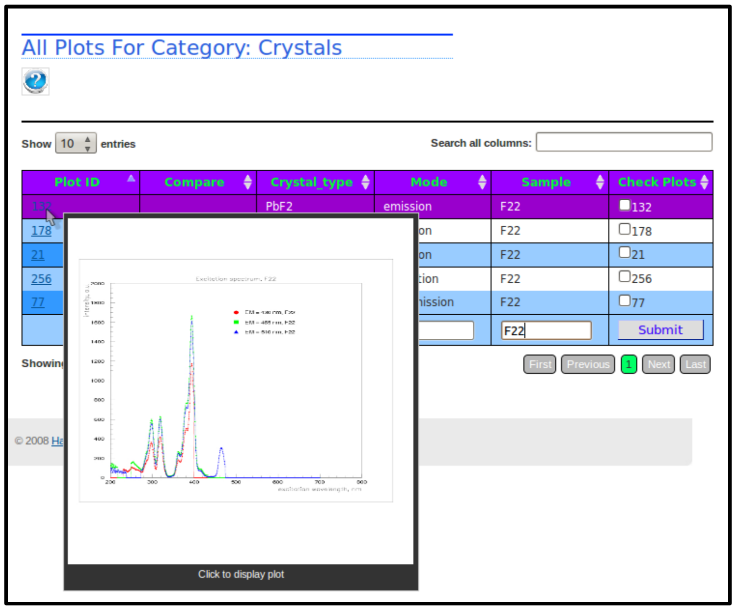


All Plots For Category: Crystals

Show 50 entries   Search all columns:

Plot ID	Compare	Crystal_type	Mode	Sample	Check Plots
1		BSO	emission	BSO10	☐1
377		BSO	emission	BSO10	☐377
379		BSO	emission	BSO10	☐379
113		BSO	emission	BSO10-2	☐113
159		BSO	emission	BSO10-2	☐159
2		BSO	emission	BSO6	☐2
114		BSO	emission	BSO6-2	☐114
160		BSO	emission	BSO6-2	☐160
3		BSO	emission	BSO7	☐3
115		BSO	emission	BSO7-2	☐115
161		BSO	emission	BSO7-2	☐161
4		BSO	emission	BSO8	☐4
116		BSO	emission	BSO8-2	☐116
162		BSO	emission	BSO8-2	☐162
5		BSO	emission	BSO9	☐5
117		BSO	emission	BSO9-2	☐117
163		BSO	emission	BSO9-2	☐163
237		BSO	excitation	BSO10-2	☐237
238		BSO	excitation	BSO6-2	☐238
239		BSO	excitation	BSO7-2	☐239

Figure 3.11: An example of the column sorting feature in the plot table. The table can be sorted by one or more column header(s) (tag name(s)) in ascending or descending order. The table above is first sorted in ascending order by crystal type, then mode, and finally by sample.


All Plots For Category: Crystals

Show 10 entries Search all columns:

Plot ID	Compare	Crystal_type	Mode	Sample	Check Plots
132		PbF2	emission	F22	☐132
178			on	F22	☐178
21			on	F22	☐21
256			tion	F22	☐256
77			nission	F22	☐77
				F22	Submit

Showin

First Previous 1 Next Last

© 2008 Ha

Click to display plot

3.3.6 *Plot preview:* If the user hovers the mouse over a plot ID (the left-most column), a small image of the plot will appear over the table (figure 3.12). The user can also click the plot ID and the image will appear as a pop-up. The user can open as many plots as he or she desires in this manner.

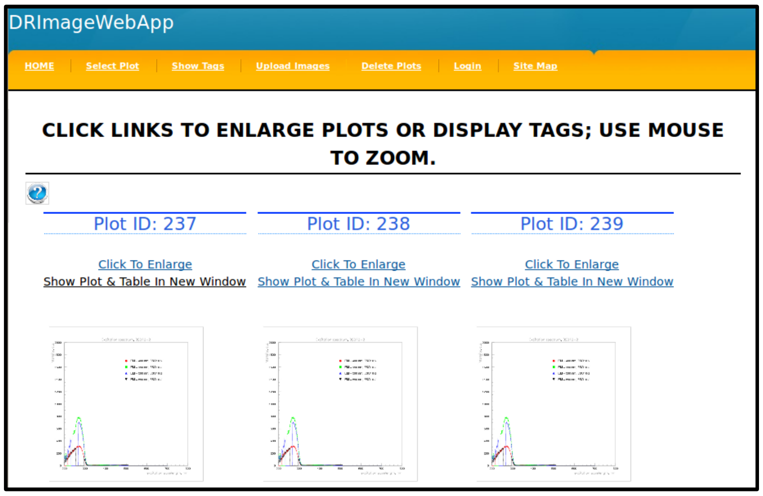
Figure 3.12: An example of the plot preview feature in the plot table. The user can see a preview of the plot by hovering over the ID link in the "Plot ID" column. If the link is pressed, the plot will display as a pop-up in its own window.

3.3.7 *Displaying (#) of entries*: Because the table might not be displaying all of the plots with the given specifications, the table displays a count of the total number of entries in the table, and how many are currently being displayed.

3.3.8 *Pagination*: In order to keep the table from becoming too large, only a certain number of entries will display at one time, where ten is the default. If there are more than ten entries in the table, the user can click the pagination buttons to go to the first, previous, next, or last page, or anywhere in between. The current page is highlighted in green.

3.3.9 Display multiple plots:

If the user wishes to view multiple plots side-by-side, he or she can check as many plots in the "check plots" column as he or she desires. Thumbnails of the plots will display together in a separate window (figure 3.13). The user can then zoom in on the plots or enlarge them.


DRImageWebApp

HOME | Select Plot | Show Tags | Upload Images | Delete Plots | Login | Site Map

CLICK LINKS TO ENLARGE PLOTS OR DISPLAY TAGS; USE MOUSE TO ZOOM.

Plot ID: 237	Plot ID: 238	Plot ID: 239
Click To Enlarge	Click To Enlarge	Click To Enlarge
Show Plot & Table In New Window	Show Plot & Table In New Window	Show Plot & Table In New Window

Figure 3.13: A screenshot of the "display multiple plots" page. If the user checks multiple plots in the table, a new window will appear with thumbnails of all the plots. Every image can be enlarged and zoomed.

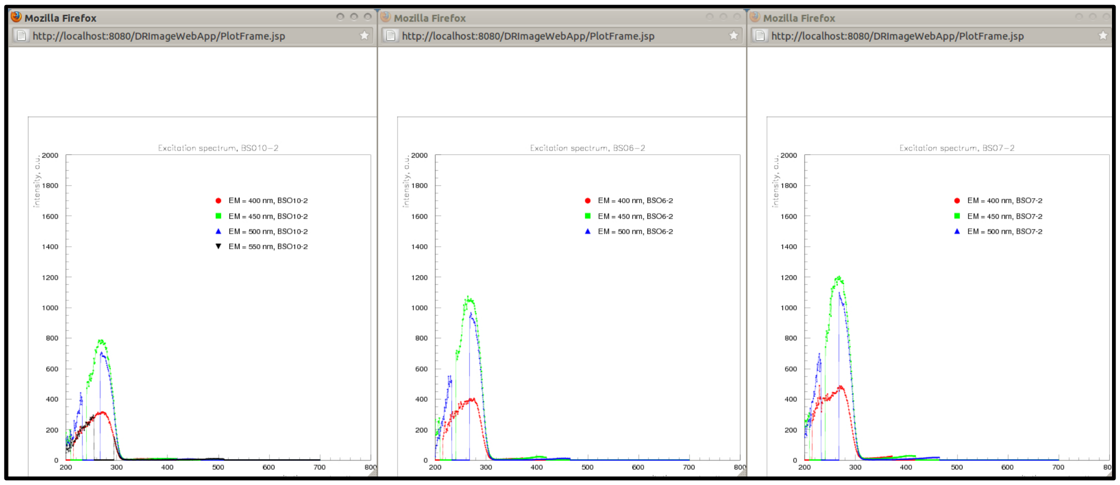
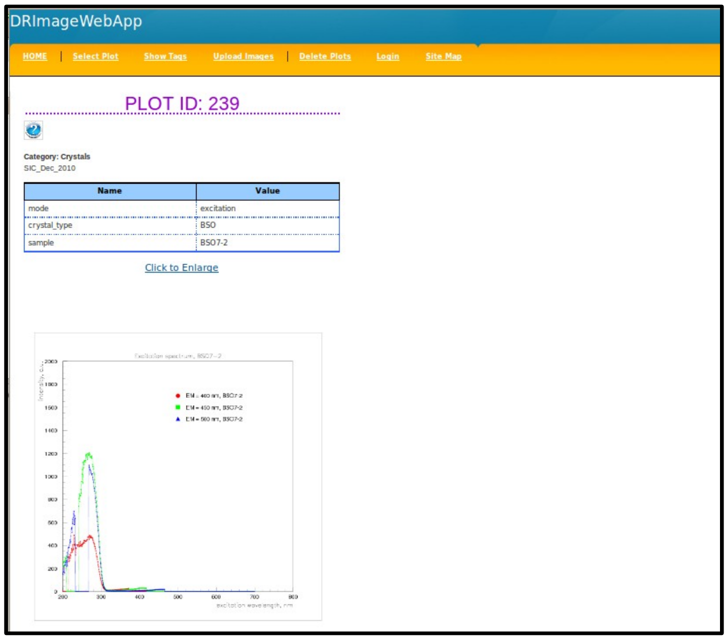


Figure 3.14: A screenshot of three plots opened as pop-ups in their own windows. JavaScript allows plots to be opened as pop-ups, so their enlarged versions to be viewed side-by-side and easily compared.

Once the new window appears, every plot can be enlarged and shown as a pop-up, displayed in a new window with a list of its tags and values, or zoomed in on in the current window. If all three plots from the above screenshot (figure 3.14) are opened as pop-ups, they can be viewed side-by-side and easily compared.

3.4 Viewing Single Plots


DRImageWebApp
HOME | Select Plot Show Tags Upload Images | Delete Plots Login Site Map
Excitation spectrum, BSO7-2

PLOT ID: 239

Category: Crystals
SIC_Dec_2010

Name	Value
mode	excitation
crystal_type	BSO
sample	BSO7-2

Click to Enlarge

Figure 3.15: A screenshot of single plot. Plots can be opened in their own windows with a table of their tags and values

Every plot can be viewed in its own window with the following information (figure 3.15):

- Help button with useful information about using the zooming and panning features
- Table of tags and values associated with the plot
- “Click to enlarge” link that will display the plot as a pop-up
- Full-size image of the plot with zooming and panning features

3.5 Zooming & Panning

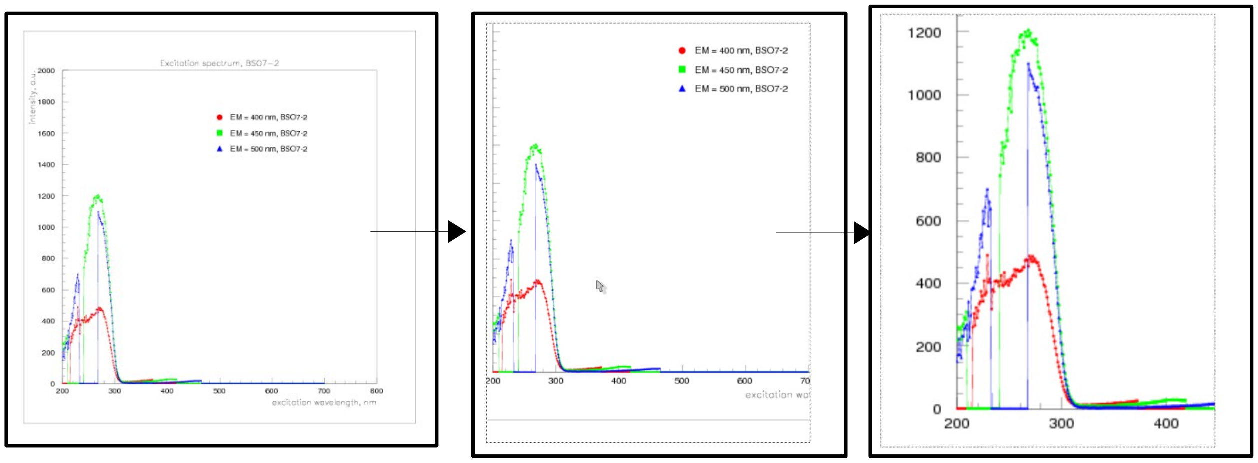


Figure 3.16: An example of a plot that has been zoomed and panned. The zooming and panning feature uses a JavaScript plug-in to allow the user to zoom and center on which part of the plot he or she desires.

The Web application also uses a JavaScript plug-in to allow the user to zoom in and pan on plots (figure 3.16). Instructions for zooming and panning can be found by clicking the question mark help button on the Web page, and are also listed below.

How to zoom:

- Use the mouse: the mouse wheel zooms into or out of the image.
- Use the keyboard: zoom in using '+', '=', or 'x' and zoom out using 'z' or '-'

How to pan:

- Use the mouse: click and drag the image to pan.
- Use the keyboard: 'w', 'a', 's', and 'd' pan up, left, down, and right respectively

4. Discussion and Conclusions

The DR Image Web Application now has much more functionality and user capability, allowing the user to choose plots based on categories, tags, and values, rather than an automatically generated ID number. This can be done quickly using appropriate prepared statements to query the SQL database. It is much faster for the program to query the database with a prepared statement than it is to loop through a list of plots. When I originally could not figure out how to send the list of current plots to the database to search through, the program looped through and queried every individual plot in the list. This took a great deal of time and was a very inefficient way of solving the problem. I was able to take advantage of the IN operator to deal with this problem so that one prepared statement could be used to query the list of all current plots.

After the database returns a list of plots that meet the user's specifications, the data is displayed in a dynamic table. At first, this table used only Java and JSP, which ended up creating very problematic and overly-complicated code. The table could be easily displayed using only Java and JSP, but in order to allow for user-interaction, JavaScript was introduced. A JavaScript plug-in allows the user to search, sort, and page through the various rows and columns of data, without needing to store and resend the list of plots from the server to the client on each change. A separate JavaScript plug-in also allows a user to open multiple plots at one time in their own pop-up window, allowing for easy plot comparison.

5. Future Work

The Web application for the Dual Readout Calorimeter Project can continue to be improved in the future, but there is one main aspect that is currently being worked on. Currently, a standalone Java application uploads the plots from a user's computer to the database. The hope is that users will be able to upload plots directly to the database with the Web application. This is a challenge because the data must be read locally from the client's computer, sent to the Web server, and then passed into the database, without the client having access to the database itself. Once this problem is solved, a user with the correct username and password will be able to login to the upload page and upload XML files directly to the database. This Web application is also a general tool that can be used as a framework for accessing other databases, based on their various tables and tags.

There are also larger goals for the Dual Readout Calorimeter Project in general. The number of Geant4 simulation tests might reach hundreds of thousands in the next couple years. This will accumulate a large quantity of data, and increase the number of categories, tags, and values in the DR Image Database. Comparison of these various tests will hopefully find the best way to construct a Dual Readout Calorimeter so the results are as accurate as possible. Additionally, the goal is for the database to become a knowledge base where all of this data will be stored. If, for example, a user knows what type of crystal or detector to use, the knowledge base will tell him or her the specific refractive index, mode, and other parameters that are needed to run the simulation in Geant4. This should help to increase the accuracy of the models and aid in the creation of a Dual Readout Calorimeter.

6. Acknowledgements

I would like to thank Linda Diepholz, Jamieson Olsen, Dianne Engram, my two mentors David Peterson and Elmie Peoples, and the entire SIST committee for providing me with this internship opportunity and helping me during the entirety of this summer. I would also like to thank Dr. James Davenport for providing me with resources for and feedback on my presentation and paper. I am very appreciative of my supervisor, Hans Wenzel, for guiding me, answering my questions, and teaching me a great deal about databases, Web applications, and everything in between. I would also like to thank Adam Para for his feedback and suggestions for the Web application every week, and for challenging me to think about how to go about solving different problems in new ways. I enjoyed getting to know the members of the Simulation of Optical Processes Group and learning more about their projects during our weekly meetings. Finally, I would like to thank my cubicle buddy and fellow SIST intern, Edgar Nandayapa, for his constant entertainment throughout the day.

7. References

[1] Bergsten, Hans. JavaServer Pages. 1st. Cambridge: O'Reilly, 2001. Print.

[2] "CSS Tutorial, SQL Tutorial, JavaScript Tutorial." Learn to Create Websites. W3Schools, 2011. Web. 1 Aug 2011. <http://www.w3schools.com/default.asp>.

[3] Fields, Duane, and Mark Kolb. Web Development with JavaServer Pages. Greewich, CT: Manning Publications Co., 2000. Print.

[4] Firoz, Munawwar. Javascript Image Viewer. Computer software. Spictrading.com Javascript Image Viewer. Spictrading, 6 Oct. 2010. Web. 21 July 2011. <http://www.spictrading.com/viewer/home.php>.

[5] Freedman, Alan. "Definition of: JSP." PCMag. The Computer Language Company , 2011. Web. 1 Aug 2011. <http://www.pcmag.com/encyclopedia_term/0,2542,t=JSP&i=45685,00.asp>.

[6] Grakalic, Alen. Easiest Tooltip and Image Preview Using JQuery. Computer software. Easiest Tooltip and Image Preview Using JQuery. CSS Globe, 8 May 2008. Web. 20 July 2011. <http://cssglobe.com/post/1695/easiest-tooltip-and-image-preview-using-jquery>.

[7] Hall, Marty. "Servlets and JSP: An Overview." Customized J2EE Training. Coreservlets.com, 2011. Web. 1 Aug 2011. <http://www.apl.jhu.edu/~hall/java/Servlet-Tutorial/Servlet-Tutorial-Overview.html>.

[8] Jardine, Allan. DataTables. Computer software. Datatables. Vers. 1.8.1. 25 June 2011. Web. 14 July 2011. <http://www.datatables.net/index>.

[9] "About." PostgreSQL. PostgreSQL Global Development Group , 2011. Web. 4 Aug 2011. <http://www.postgresql.org/about/>.

[10] "Using Prepared Statements." The Java Tutorials. Oracle, 2011. Web. 8 Aug 2011. <http://download.oracle.com/javase/tutorial/jdbc/basics/prepared.html>.

8. Appendix

8.1 List of Figures

8.2 List of Tables