A Specialized Framework for Data Retrieval Web Applications

Jerzy M. NOGIEC, Kelley TROMBLY-FREYTAG, Dana WALBRIDGE
Fermi National Accelerator Laboratory
Batavia, IL 60510, USA

ABSTRACT

Although many general-purpose frameworks have been developed to aid in web application development, they typically tend to be both comprehensive and complex. To address this problem, a specialized server-side Java framework designed specifically for data retrieval and visualization has been developed. The framework's focus is on maintainability and data security. The functionality is rich with features necessary for simplifying data display design, deployment, user management and application debugging, yet the scope is deliberately kept limited to allow for easy comprehension and rapid application development. The system clearly decouples the application processing and visualization, which in turn allows for clean separation of layout and processing development. Duplication of standard web page features such as toolbars and navigational aids is therefore eliminated. The framework employs the popular Model-View-Controller (MVC) architecture, but it also uses the filter mechanism for several of its base functionalities, which permits easy extension of the provided core functionality of the system.

Keywords: Framework, Web Application, Data Retrieval, JSP, Servlet.

1.WEB APPLICATION FRAMEWORKS

Many web application frameworks have been developed and more are appearing every day. Web applications are fundamentally suited to the framework concept, because they deal with the presentation of data in a consistent logical manner for human viewing. The repetitive aspect of presentations (navigation bars, headers, footers, etc) requires a degree of structure that makes traditional presentation applications (HTML, JSP) difficult to maintain and change.

Frameworks can supply the basis for abstracting the human factors of the design out of the basic presentation of the data, and allow each unit of data presentation to stand on its own. Frameworks, such as Struts [1], provide partitioning and modularity, separation of tasks, improved code manageability, and more extensible and adaptable code.

The framework being described is specialized on numerical data retrieval and display. It uses JSP tags to provide a simple mechanism to query data from a database, and display the query results in several forms: tables, charts and Microsoft Excel spreadsheets. It also provides assorted administrative functionality, such as user authentication and page-granularity access control, embedded debugging mechanisms, and a coarse to fine-grained logging system.

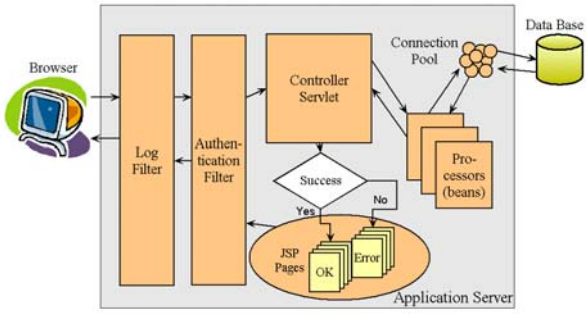

Figure 1. Framework architecture.

2. ARCHITECTURE

The framework is based on the MVC pattern and therefore uses one central servlet to handle all incoming requests (Figure 1). This means there is a single point of entry for the whole application and the maintenance of the application is simplified. The business logic of the application is supplied in a number of processor objects to which processing is delegated by the framework when the request is dispatched. This significantly reduces the complexity of the controller and removes the maintenance problems of the traditional front controller.

The principle feature of the framework is an action mapping mechanism that controls data processing, layout and page flow. The controller passes control to the processor, which accesses the data sources and performs the necessary data manipulations and processing. The mapping of requests to processors and output pages is configuration driven (see a processing specification for an example request in Figure 2.) and allows for independent specification of the page flow for both successful and erroneous outcomes.

```
<request
   id="DayView"
   processor="calendar.DateProcessor"
   success="/DayView.jsp"
   title="DayView"
   layout="/MainLayout.jsp"
   error="/Error.jsp"
   access="system"
/>
```

Figure 2. A request processing specification.

The configuration file provides a single resource that defines all of the site navigation. Selection of processing and displays are limited to the configuration and no re-programming of the controller servlet is needed. Since the controller handles view selection, it can consistently apply templates and security policies across all views. The JSP is simply responsible for retrieving any objects or beans that may have been previously created by the processor, and extracting the dynamic content for insertion within static templates.

Views use virtual links rather than hard coded ones to remove the dependencies between views so they can be reused or modified freely. Hence, the application scalability is greatly improved.

```
<login
   name="Jerry"
   password="mypassword"
   access="user|admin"
   attribute="reader"
/>
```

Figure 3. User authentication information.

2. AUTHENTICATION AND ACCESS CONTROL

The framework supplies an integrated authentication and access authorization mechanisms, which are based on groups and permissions. Access to various parts of the application or its functionality can be easily controlled, with the access granularity defined to the level of a single page. The authentication mechanism is implemented as a filter with access control policies are enforced by the controller. The controller will allow for serving a request if the user belongs to a group listed for this request. Each user can, of course, belong to several groups. In addition, one can specify a separate set of application specific attributes for each user, which can be then used to further specify various application access rights the user was granted (Figure 3). For instance, some users may be allowed to modify entries made by others, whereas other users may not. Either a database or an XML file can be used for storing authentication and authorization data.

The same filter mechanism is also used for logging and auditing and allows for tracking users of the web application and gathering their use profiles as well as allowing security monitoring and auditing. Various levels of logging detail are provided, initial values of which are set in the application parameters. These can be altered at run time. Logging output can be sent either to the console or a file or both.

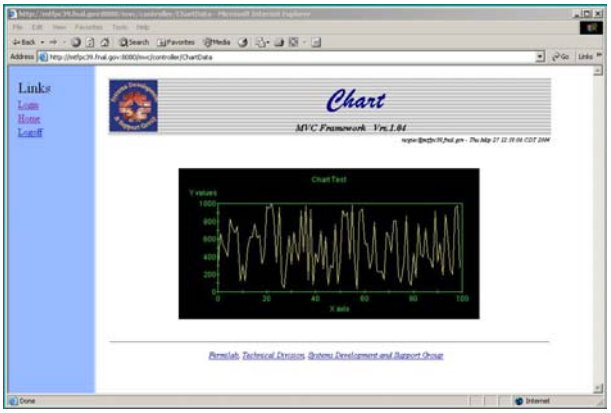

Figure 4. Example charting output.

3. DATA ACCESS AND VISULIZATION

The framework focuses on database access and data presentation with built-in support from a customized tag library. Database connection pooling allows for optimal use of resources. Tags allow for database access, graphic (Figure 4) and tabular data visualization and reformatting the data into Excel format. Data drill-down data capabilities are also supported.

The framework's core tag library provides tags to obtain and visualize data. It consists of several tags divided into three functional areas: debug, query and display. The debug tag gives a mechanism to provide debugging information to be displayed when the application's debug flag is set (this can be set on the fly). This allows for more convenient testing, with less of the JSP edit-refresh-edit cycling.

The query and substitute tags allow for flexible creation and execution of database queries. The substitute tag allows the simple substitution of request

variables within a string (a database query) by marking the word to be substituted with a ^. Should this character be needed as a literal within the query, the substitute tag has an attribute that allows you to choose the substitution character. The query tag sends the string in its "in" attribute to be executed by the database, and names the results with the "out" attribute. The data display tags use these results as input arguments.

The various database connection parameters are supplied in the application configuration file. This framework can use any database for which a JDBC support is available. At this time, connection to only one database per application is supported.

4. LAYOUT MECHANISM

The presentation of data is supported in the framework by a layout mechanism based on templates and implemented as a tag library. Layouts can be independently defined for single pages or can be shared among multiple pages.

The template mechanism is implemented in a tag library that consists of three tags: insert, put and get. The insert tag provides the mechanism for linking the various generic parts of a page layout (header, footer, etc.) with the JSP pages that will render their output. When used with context variables, this creates a powerful mechanism to provide a single layout for use within the entire web application. The insert tag takes as a parameter the JSP template page it is providing the values for. Within the insert tag, the put tag is used to define the actual values for the generic layout parts. The definition of these parts are totally within the developer control, and are not limited to "header", "footer" etc. When the parts have been defined, the specified layout JSP is displayed. The get tag is used within the specified layout JSP, to obtain the values set in the insert tag for those generic sections of the layout.

5. DEBUGGING AND ERROR HANDLING

Extensive, built-in debugging features allow for easy troubleshooting and aid in development of applications. The framework uses debugging levels to determine the required amount of generated debug information and permits independent control of debugging levels for different functional aspects of the application, with the debug output sent to either/or files as well as the console. In addition, all pages have an embedded debugging visualization, the display of which can be controlled on the fly, without restarting the application. The tag libraries included in the framework provide tracking parameters, as well as session and application states.

6. APPLICATIONS

The framework has been successfully employed in the development of such data-centered applications as the Accelerator Component Test Result Report System to access results of magnetic measurements [2][3] and the Operations Activity Calendar utility (Figure 5) and the Group Planning utility, where it proved to be both powerful and easy to use.

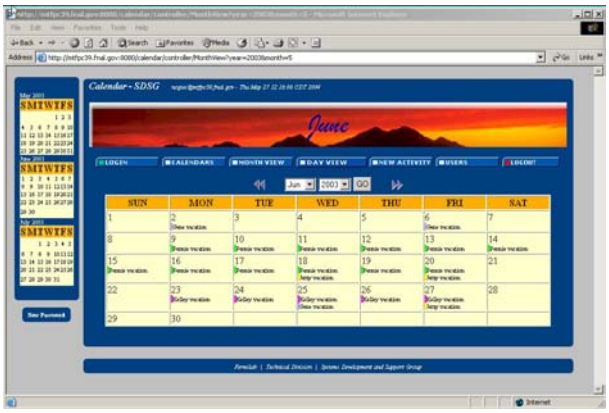

Figure 5. A calendar application.

7. CONCLUSIONS

The presented framework greatly simplifies development of data-centered web applications. It succeeded in reducing maintenance efforts of such applications by consolidating application data flow and processing in a single XML configuration file. This data-driven approach combined with a set of tag libraries allowed for practically eliminating Java coding from JSP and produced an easy to understand and maintain system. Although, the framework is targeted at data processing applications it can be, in its simplest form, also used for organizing static web site contents, allowing for use of logical pointers instead of hard-coded hyperlinks.

8. REFERENCES

[1] http://jakarta.apache.org/struts/
[2] J.M. Nogiec et al, "A Flexible and Configurable System to Test Accelerator Magnets", PAC'01, Chicago, 2001
[3] J.M. Nogiec et al, "Hierarchical Data Archival System for EMS", PCaPAC'02, Frascatti, Italy, 2002