INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH & DEVELOPMENT

ISSN 2278 – 0211 (Online)

Struts Generator
A Review Paper on Generating Struts Framework from HTML as Input

Nilam Dhatrak
Student, Department of Computer Engineering, Sinhgad Academy of Engineering, Kondhwa, Savitribai Phule Pune University, Pune, Maharashtra, India
Jui Pote
Student, Department of Computer Engineering, Sinhgad Academy of Engineering, Kondhwa, Savitribai Phule Pune University, Pune, Maharashtra, India
Shraddha Rathod
Student, Department of Computer Engineering, Sinhgad Academy of Engineering, Kondhwa, Savitribai Phule Pune University, Pune, Maharashtra, India
Pratik Baldota
Student, Department of Computer Engineering, Sinhgad Academy of Engineering, Kondhwa, Savitribai Phule Pune University, Pune, Maharashtra, India

Abstract:
The time and effort that the developer would normally spend in trying to generate the Struts hierarchy and the Presentation layer, the application-independent task, is immensely reduced with the help of this code generator. He can now concentrate on the Business Logic, which is application-specific.
The Code Generator is a tool capable of converting HTML input to JSP output along with Action forms, Action classes and the struts-config.xml file. So in effect it creates the Presentation Layer for the web application, which can later be deployed on the server. This application is smart enough to handle all configuration required for struts application.

Keywords: *Code generator, MVC architecture, JSP, HTML, TOMCAT*

1. Introduction

For a long time now, extensive coding has been a cumbersome task for the developer in terms of both time and money. The software industry's thought-process has been driven by a need to reduce this effort. Several frameworks, such as Struts, Java Server Faces, FuseBox, etc have been derived to provide support for Web Application Development tasks. To allow the developer to get on with the business-logic implementation, a tool is needed to simplify the tedious programming tasks of Presentation Layer development. And a Code Generator is a step in this direction. Aim is to implement this Presentation Layer for the Struts framework.
The initial stages of any project involve immense analysis and research. Before moving on with design and implementation, the project manager with the help of HTML pages, conveys his perception of the project to the client. In the usual scenario, these would be rendered useless, and the developer would start his work from scratch. Now these HTML pages can be used as input to the "Code Generator". The Code Generator is a tool capable of converting HTML input to JSP output along with generating Action forms, Action classes and the struts-config.xml file. This output generated, along with the Business Logic, can be deployed on the server as a Web Application. The HTML input alone is not sufficient to generate all these files. HTML is used to display static content, whereas JSPs provide dynamic aspect to a web page. JSP supports the features necessary to complete a request-response cycle. HTML has certain inadequacies that need to be fulfilled in the form of extra input provided by the user. Since the actions differ for every application, the action-mapping present in the struts-config.xml file is unique and is generated independently each time. The project aims to be application-independent.

2. System Scope

The scope of the system is restricted to the HTML tag library of the Struts framework. There are two other libraries provided by the Struts framework with tags that involve business logic. Since the project deals with the presentation layer alone, handling of tags related to business logic is not required. Hence, the scope of the system is limited to the HTML tag library. Certain tags in this library

don’t have corresponding tags in HTML with the same functionality. Those tags cannot be dealt with because of insufficient information. There are a few other tags that are indirectly related to the tags in the Bean and Logic tag libraries available in Struts.

3. Background

A code generator reads the project meta-data (for example a database model) and churns out well-formed source code to a specific set of design patterns. Think of a generator as being a bonus member of a project team - one who embodies the experience of many developers, produces ultra-consistent code in minutes instead of months, and should (in theory, at least) have a much, much lower bug rate than the rest of the team.

It should be stressed, however, that code generation - whilst incredibly useful - is no silver bullet. It won't magically guarantee success for the project. It still needs quality staff to evaluate, to design, and to code all the areas not covered by the generator. What a generator can give is a massive head start in any new project.

Code generators can also make a project very agile. A change in implementation is simply a different rendering of the same meta-data. Anything that is repetitive can be automated. This is an approach that is already gathering speed in the agile community, for example with automated unit tests and functional tests. Nowadays it is rare to see a project that doesn't use an automated build script (e.g. using the Ant build tool). Automation is the key to agility.

Similarly, given the correct conditions, a lot of source code can be automatically generated - the programmer is then free to "fill in the gaps". The gain in development speed via code generation (under the right conditions) cannot be emphasized enough.

Code generators are distinct creatures from the "wizards" seen in IDEs such as NetBeans and Eclipse. Wizards are great time-savers for quickly generating one or two classes, but that does tend to be as far as they go. The difference is that code generators churn out entire APIs, sometimes even entire applications.

Currently, there is a proliferation of code generators on the market - some free, open source or shareware. More seem to be appearing all the time. The diversity of the generators' approaches and varying technologies (of both the code being generated and the tool used to do the donkey work) are indicative of a burgeoning software market, awash with fresh ideas - but noticeably lacking in formal standards at this stage.

Following is the diagram which shows the actual flow of the proposed project.

It starts by first accepting the input HTML file from the user. Now the transition forks into two parallel activities, i.e. generating the struts hierarchy and parsing of the input HML file. On parsing the input file, the HTML tags are obtained, again the transition forks to give the required outputs, i.e. the action forms and action classes, struts config.xml and the corresponding JSP page.

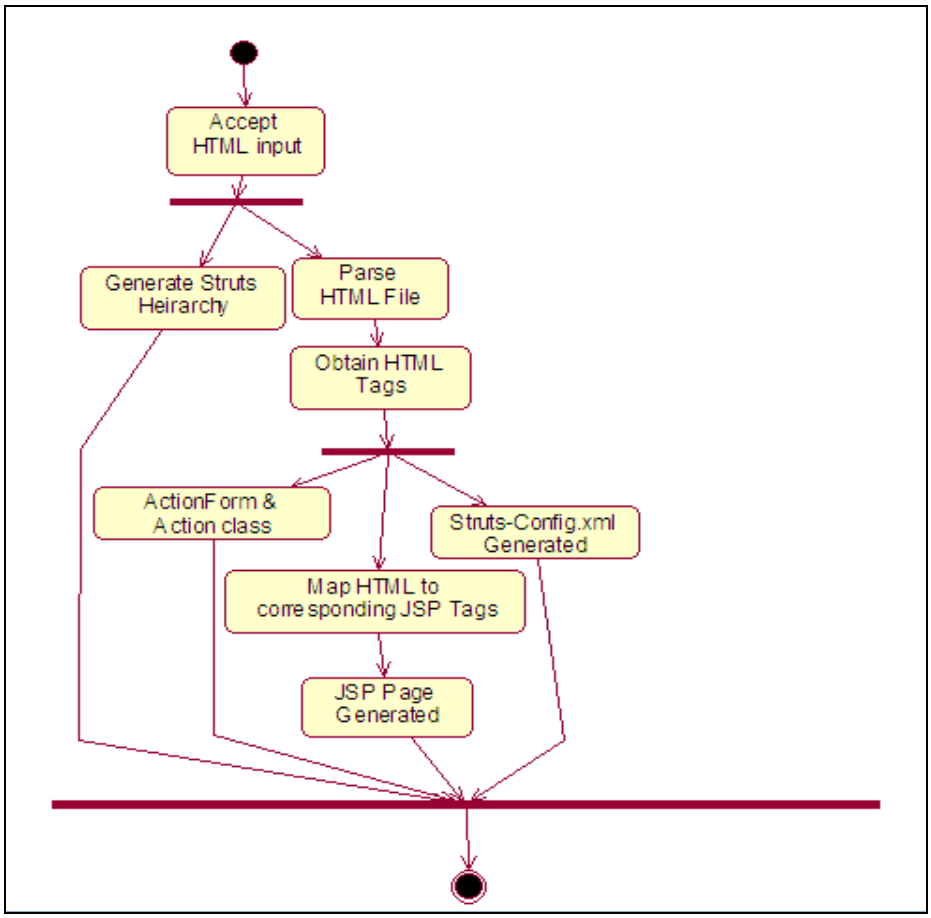

Figure 1: Activity Diagram

4. Technologies

HTML: A Web Server responds to a browser's request by returning as HTML page. The returned page can be static HTML page or a dynamic HTML page. User requests the page by typing an URL or by clicking the link returning the page. URL syntax is a specific sequence of protocol, domain name and the path to the requested information Protocol is the communication method used to gain access to information, e.g. http, ftp. Domain name is Domain Name System name of server that contains the information.

JAVA: Primary motivation was the need for a platform-independent (that is, architecture-neutral) language that could be used to create software to be embedded in various consumer electronic devices such as microwave ovens and remote controls etc. The trouble with C and C++ (and most other languages) is that they are designed to be compiled for a specific target. Although it is possible to compile a C++ program for just about any type of CPU to do so requires a full C++ compiler targeted for that CPU. The problem is that compilers are expensive and time consuming to create. An easier- and more cost-efficient – solution was required. In an attempt to find such a solution, Gosling and others began work on a portable, platform –independent language that could be used to produce code that would run on a variety of CPUs under differing environments. This effort ultimately leads to the development of Java.

The second force was of course, the World Wide Web. With the emergence of the World Wide Web, Java was propelled to the fore-front of computer language design, because the Web too, demanded portable programs.

The Internet helped catapult Java to this forefront of programming, and Java, in turn, has had a profound effect on the Internet. The reason for this is quite simple: Java expanded the universe of objects that can move freely in cyberspace. In a network two very broad categories of objects are transmitted the server and the personal computer: passive information, and dynamic, active programs. For example, when you read your e-mail, you are viewing passive data. Even when you download a program, the program's code is still only passive data until you execute it. However, a second type of object can be transmitted to your computer: a dynamic self-executing program. Such a program is an active agent on the client computer, yet is initiated by the server. For example the server to display properly the data that the server is sending might provide a program.

4.1. Model-View-Controller

Issues with validation, flow control, and updating the state of the application need to be taken care of. This is where MVC comes to the rescue. MVC helps resolve some of the issues with the single module approach by dividing the problem into three categories:

4.2. Model

The model contains the core of the application's functionality. The model encapsulates the state of the application. Sometimes the only functionality it contains is state. It knows nothing about the view or controller.

4.3. View

The view provides the presentation of the model. It is the look of the application. The view can access the model getters, but it has no knowledge of the setters. In addition, it knows nothing about the controller. The view should be notified when changes to the model occur.

4.4. Controller

The controller reacts to the user input. It creates and sets the model

4.5. MVC Model 2

The Web brought some unique challenges to software developers, most notably the stateless connection between the client and the server. This stateless behavior made it difficult for the model to notify the view of changes. On the Web, the browser has to re-query the server to discover modification to the state of the application.

Another noticeable change is that the view uses different technology for implementation than the model or controller. Of course, Java could be used (or PERL, C/C++ or whatever) to generate HTML. There are several disadvantages to that approach:

Java programmers should develop services, not HTML.

Changes to layout would require changes to code.

Customers of the service should be able to create pages to meet their specific needs.

The page designer isn't able to have direct involvement in page development.

- HTML embedded into code is ugly.

For the Web, the classical form of MVC needed to change.

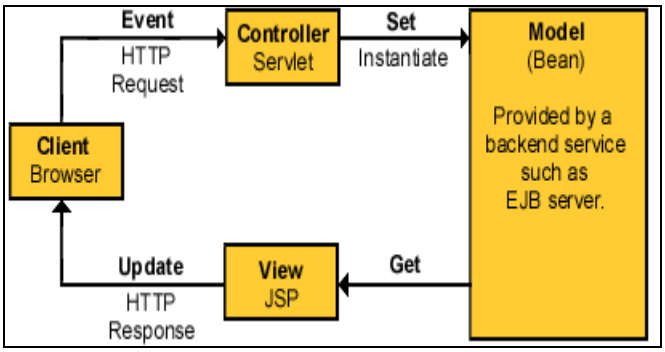

Figure 2: MVC Model 2

4.6. MVC2 Implementation

Struts is a set of cooperating classes, servlets, and JSP tags that make up a reusable MVC 2 design. This definition implies that Struts is a framework, rather than a library, but Struts also contains an extensive tag library and utility classes that work independently of the framework. Figure displays an overview of Struts. The Struts framework uses many design patterns. The main design pattern behind the struts framework is MVC which separates business logic from controller and view/display logic. Controller is based on command and Front Controller design pattern. Action classes use Adapter Design Pattern and process() method of the Request Processor uses the Template Method Design Pattern[1].

4.7. Benefits of Struts MVC

1. Configuration is centralized- The mapping that has to be done is present in the XML configuration file. This allows loose coupling so changes can be made and effect of the changes can be found by just editing a single xml file.
2. Form Validation- Struts provide strong validation of the form which can be applied to the registration form of the students who wish to enroll for a course. This feature is present by default in the struts framework [3].

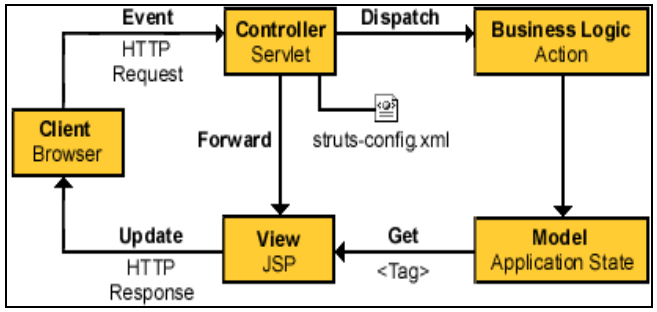

Figure 3: Struts overview

4.8. Struts Overview

4.8.1. Client Browser

An HTTP request from the client browser creates an event. The Web container will respond with an HTTP response.

4.8.2. Controller

The Controller receives the request from the browser, and makes the decision where to send the request. With Struts, the Controller is a command design pattern implemented as a servlet. The struts-config.xml file configures the Controller.

4.8.3. Business Logic

The business logic updates the state of the model and helps control the flow of the application. With Struts this is done with an Action class as a thin wrapper to the actual business logic.

4.8.4. Model State

The model represents the state of the application. The business objects update the application state. Action Form bean represents the Model state at a session or request level, and not at a persistent level. The JSP file reads information from the Action Form bean, using JSP tags.

4.8.5. View

The view is simply a JSP file. There is no flow logic, no business logic, and no model information -- just tags. Tags are one of the things that make Struts unique compared to other frameworks like Velocity.

4.9. JSP/Servlet

Java Server Pages technology allows web developers and designers to easily develop and maintain dynamic web pages that leverage existing business systems. Struts separates Java codes of JSP by Java Bean and Action class to be the MVC mode, transmits data among the three partitions of Model, View and Controller, demonstrates the connection between various classes and JSP pages by configuration files[2]. As part of the Java technology family, JSP enables rapid development of web-based applications that are platform-independent. JSP separates user interfaces from content generation, enabling designers to change the overall page layout without altering the underlying dynamic content.

In its basic form, a JSP page is simply an HTML web page that contains additional bits of code that execute application logic to generate dynamic content. This application logic may involve JavaBeans, JDBC objects, Enterprise Java Beans (EJB), and Remote Method Invocation (RMI) objects, all of which can be easily accessed from a JSP page. For example, a JSP page may contain HTML code that displays static text and graphics, as well as a method call to a JDBC object that accesses a database; when the page is displayed in a user's browser, it will contain both the static HTML content and dynamic information retrieved from the database.

The separation of user interface and program logic in a JSP page allows for a very convenient delegation of tasks between web content authors and developers. It also allows developers to create flexible code that can easily be updated and reused. Because JSP pages are automatically compiled as needed, web authors can make changes to presentation code without recompiling application logic. This makes JSP a more flexible method of generating dynamic web content than Java servlets, whose functionality Java Server Pages extend

4.10. Web Server-Tomcat

Apache Tomcat is the servlet container that is used in the official Reference Implementation for the Java Servlet and Java Server Pages technologies. Apache Tomcat is developed in an open and participatory environment.

5. Conclusion

The paper chronologically presents development of STRUTS framework. The Struts was designed with the intention of providing an open−source framework for creating Web applications that easily separate the presentation layer and allow it to be abstracted from the transaction/data layers. The system takes HTML pages as input and provides JSP pages along with Action Forms, Action Classes, and struts. Config.xml file. The system is based on MVC architecture which separates model, view and control part. Hence the work of developer is simplified and he can concentrate on business logic.

6. Future Work

Here concentration is on STRUTS 1.3; in future STRUTS 2.0 can be used.

7. Acknowledgement

We are highly indebted to Prof. D.K Joshi for her guidance and constant supervision as well as for providing necessary information.

We would like to express our gratitude towards Prof. B.B.Gite & Prof. Gandhali S. Gurjar for their kind co-operation and encouragement.

8. References

1. Using Factory Design Pattern for Database Connection and Daos (Data Access Objects) With Struts Framework International Journal of Engineering Research and Development e-ISSN: 2278-067X, p-ISSN : 2278-800X
2. Research of Structure Integration based on Struts and Hibernate 978-0-7695-3507-4/08 $25.00 © 2008 IEEE DOI 10.1109/CSIE.2009.561
3. Integration of Struts, Spring and Hibernate for an University Management System, International Journal of Emerging Technology and Advanced Engineering Website: www.ijetae.com (ISSN 2250-2459, Volume 2, Issue 6, June 2012)