Struts Spring Hibernate (SSH)

Ronghong Li
You Li
Paul Filipow
Miao Xu
Chao Rui

1.Introduction

Struts, Spring and Hibernate (SSH) make up a Framework for development of Web application. It uses the Model-View-Controler design pattern and JavaBean as the basic technologies. Java and JSP is used as the implementing language in SSH development. Tomcat is the web server.

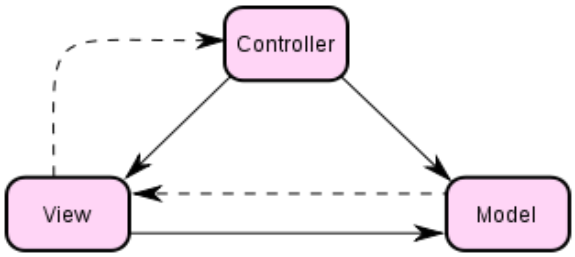
1.1 Model-View-Controler Design Pattern

Model–view–controller (MVC) is a software architecture, currently considered anarchitectural pattern used in software engineering. The pattern isolates "domain logic" (the application logic for the user) from the user interface (input and presentation), permitting independent development, testing and maintenance of each (separation of concerns).

1.1.1 Three layers

- The model
  - Represents the underlying data and business logic in one place
  - Contains no information about the user interface
- The view
  - The user interface – things the user can see and respond to
  - Represent a window into the model – there can be many of these
- The controller
  - Connects the model and the view

Used to communicate between the model and view



1.1.2 Advantages

Promotes code reuse
- The purpose of the model is to provide business logic and data access in one place
- This logic can be reused in many applications at the same time without the need for any extra coding

Reduces development time
- The model, view, and controller can be developed in parallel

More maintainable
- The view can be changed without affecting the model
  A Web page view can be changed to display a chart instead of a table with no change to the model
- The model can be changed without affecting the view
  For example, the way in which an insurance premium is calculated may change, but the interface to the business method remains the same
- Data can be moved without affecting the view or model
  The layering concept allows for more flexibility

1.2 JavaBean

JavaBeans are reusable software components for Java that can be manipulated visually in a builder tool.

1.2.1 Introduction

Practically, they are classes written in the Java programming language conforming to a particular convention. They are used to encapsulate many objects

into a single object (the bean), so that they can be passed around as a single bean object instead of as multiple individual objects. A JavaBean is a Java Object that is serializable, has a nullary constructor, and allows access to properties using getter and setter methods.

1.2.2 JavaBean conventions

- The class must have a public default constructor (no-argument). This allows easy instantiation within editing and activation frameworks.
- The class properties must be accessible using *get*, *set*, *is* (used for boolean properties instead of get) and other methods (so-called accessor methodsand mutator methods), following a standard naming-convention. This allows easy automated inspection and updating of bean state within frameworks, many of which include custom editors for various types of properties.
- The class should be serializable. It allows applications and frameworks to reliably save, store, and restore the bean's state in a fashion independent of the VM and of the platform.

2. Spring

2.1 Introduction

The **Spring Framework** is an open source application framework for the Java platform.

The first version was written by Rod Johnson who released the framework with the publication of his book*Expert One-on-One J2EE Design and Development* in October 2002. The framework was first released under the Apache 2.0 license in June 2003. The first milestone release, 1.0, was released in March 2004, with further milestone releases in September 2004 and March 2005. The Spring 1.2.6 framework won a Jolt productivity award and a JAX Innovation Award in 2006. Spring 2.0 was released in October 2006, and Spring 2.5 in November 2007. In December 2009 version 3.0 GA was released. The current version is 3.0.5.[4]

The core features of the Spring Framework can be used by any Java application, but there are extensions for building web applications on top of the Java EE platform. Although the Spring Framework does not impose any specific programming model, it has become popular in the Java community as an alternative to, replacement for, or even addition to the Enterprise JavaBean (EJB) model.

2.2 Basic Goal

Basically Spring Framework helps us manage the relationships between different parts of the project (bean management) to loosen the coupling between them with modules that will be mentioned after. Loose coupling will make the whole project easy to test, easy to change and easy to management.

2.3 Modules

2.3.1 Inversion of Control Container

Central to the Spring Framework is its Inversion of Control container, which provides a consistent means of configuring and managing Java objects using callbacks. The container is responsible for managing object lifecycles: creating objects, calling initialization methods, and configuring objects by wiring them together.

Spring use dependency injection to achieve IoC which means that we don't have to deal with the dependency in the code. We only need to define the interface and set the detail in Spring's configure file, Spring will inject the dependency (like create the instance we need) during the runtime.

A good example: http://www.vaannila.com/spring/spring-ioc-1.html

2.3.2 Aspect-Oriented Programming

The Spring Framework has its own AOP framework which modularizes cross-cutting concerns in aspects. The motivation for creating a separate AOP framework comes from the belief that it would be possible to provide basic AOP features without too much complexity in either design, implementation, or configuration. The SpAOP framework also takes full advantage of the Spring Container.

AOP is used in the Spring Framework to:

- provide declarative enterprise services, especially as a replacement for EJB declarative services. The most important such service is *declarative transaction management*.
- allow users to implement custom aspects, complementing their use of OOP with AOP.
- More info: http://static.springsource.org/spring/docs/2.5.x/reference/aop.html

2.3.3 Some other modules:

1. Data access framework

2. Transaction management framework
3. Model-view-controller framework
4. Remote access framework
5. Convention-Over-Configuration Rapid Application Development
6. Batch Framework
7. For all those modules above, you can get more information about them on wiki and Spring's website.

2.4 Some cons

a) May make the code harder to read and harder to maintain.

As we abstract the dependency from the code to Spring's configure file and let Spring Framework take care all of them, our code is no longer contiguous. It is sometimes hard for us to read the code, especially when we are not the one who wrote it.

b) Cannot take advantages of IDE.

As we all know, IDE such like eclipse can help us a lot when will deal with the relationships or in other words dependency of the objects. But now IDE does not support Spring so well as the row java code.

c) Resources consuming

Even Spring Framework is a lightweight container, it still definitely need resources.

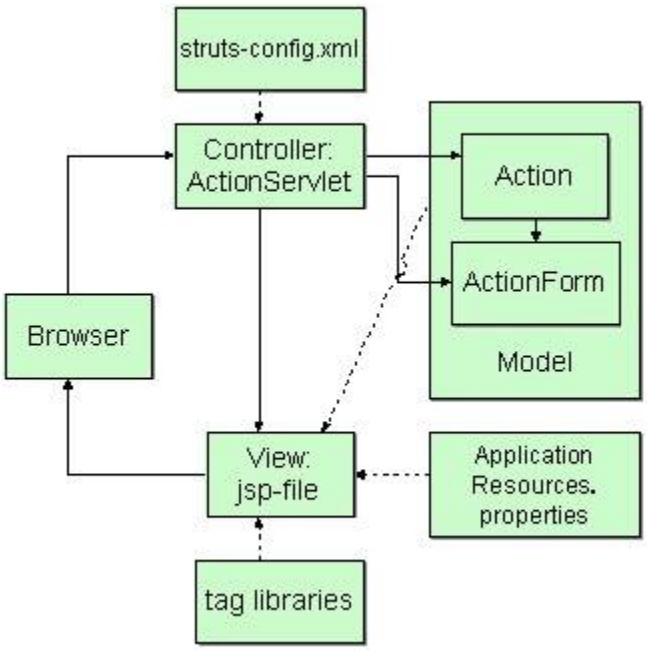
3. Struts

3.1 About Struts

Apache Struts is an open-source web application framework for developing Java EE web applications. It uses and extends the Java Servlet API to encourage developers to adopt a model-view-controller (MVC) architecture. It is a flexible control layer based on standard technologies like Java Servlets, JavaBeans, ResourceBundles, and XML, as well as various Apache Commons packages, like BeanUtils and Chain of Responsibility. The framework helps you create an extensible development environment for your application, based on published standards and proven design patterns.

3.2 Struts' Role

The goal of Struts is to separate the *model* (application logic that interacts with a database) from the *view* (HTML pages presented to the client) and the *controller* (instance that passes information between view and model). Struts provides the controller (a servlet known as ActionServlet) and facilitates the writing of templates for the view or presentation layer (typically in JSP, but XML/XSLT and Velocity are also supported). The web application programmer is responsible for writing the model code, and for creating a central configuration filestruts-config.xml that binds together model, view and controller.

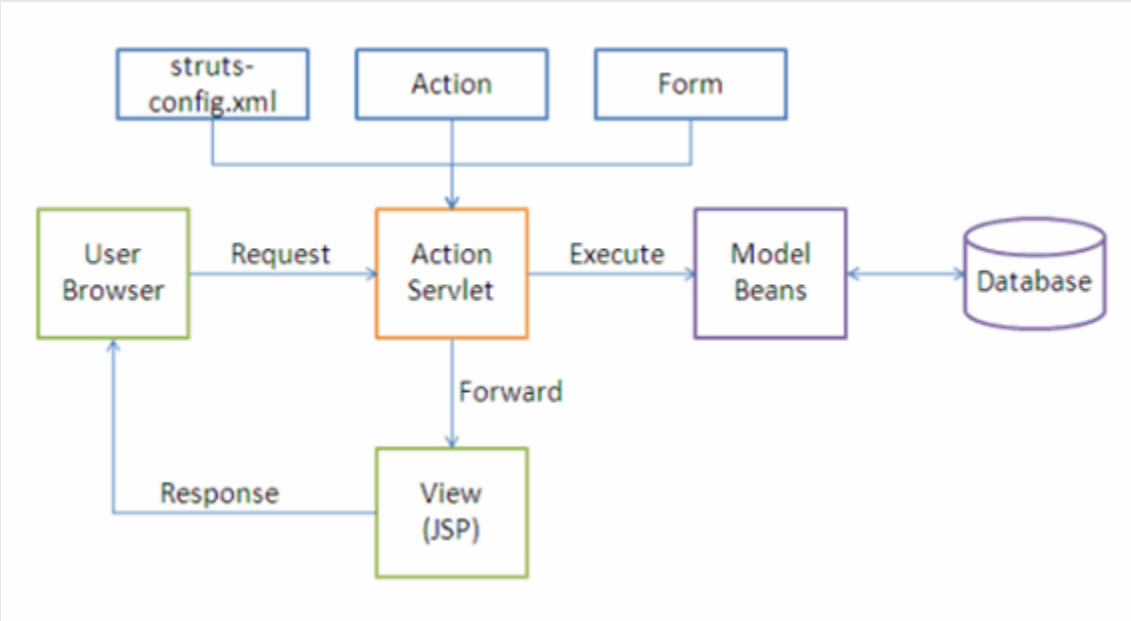


As shown in the graph, Java Servlets are designed to handle requests made by Web browsers. Java ServerPages are designed to create dynamic Web pages that can turn billboard sites into live applications. Struts uses a special Servlet as a switchboard to route requests from Web browsers to the appropriate ServerPage. This makes Web applications much easier to design, create, and maintain.

3.3 How Does Struts Work?

In general, requests from the client are sent to the controller in the form of "Actions" defined in the configuration file; if the controller receives such a request it calls the corresponding Action class that interacts with the application-specific model code. The model code returns an "ActionForward", a string telling the controller what output page to send to the client. Information is passed between model and view in the

form of special JavaBeans. A powerful custom tag library allows it to read and write the content of these beans from the presentation layer without the need for any embedded Java code.



The ActionServlet class

ActionServlet is the Command part of the MVC implementation and is the core of the Framework. ActionServlet(Command) creates and uses Action, an ActionForm, and ActionForward. As mentioned earlier, the struts-config.xmlfile configures the Command. During the creation of the Web project, Action and ActionForm are extended to solve the specific problem space. The file struts-config.xml instructs ActionServlet on how to use the extended classes.

The ActionForm class

ActionForm maintains the session state for the Web application. ActionForm is an abstract class that is sub-classed for each input form model.

The Action class

The Action class is a wrapper around the business logic. The purpose of Action class is to translate theHttpServletRequest to the business logic.

3.4 Struts Applications

If you need to write a very simple application, with a handful of pages, then a solution that uses only server pages will be good enough.

But, if you are writing a more complicated application, with dozens of pages,

that need to be maintained over time, then Struts can help.

4. Hibernate

4.1 Introduction

Hibernate is an object-relational mapping (ORM) library for the Java language, providing a framework for mapping an object-oriented domain model to a traditional relational database. It allows application developers high-level access to database technology by introducing a level of abstraction between Java objects and relational database tables.

4.2 Benefits

Hibernate offers a number of benefits to application developers. It provides for full object-oriented programming even when saving/retrieving data from database, and completely removes the developer's need for SQL and database queries, and relational table construction.

In addition to accelerating development time, the abstraction Hibernate provides also creates a higher level of data independence than does the traditional SQL/ relational model. By removing database structure and queries from the developer, applications and databases can change independently of each other, without any refactoring on either end.

4.3 Object-relational mapping

The foundation of Hibernate is a process called "object-relational mapping," which is a programming technique for converting otherwise incompatible types from different object oriented programming languages. Hibernate uses object-relational mapping to map Java classes to relational database tables and vice-versa. Object-relational mapping does this by creating a "virtual object database" that can be accessed from Java. This virtual object database provides a compromise between two common types of database management systems: object oriented database management systems (OODBMS), and relational database management systems (RDBMS).

RDBMS is probably the most common database management system, but involves a relatively low-level of operation, especially when used in high-level object oriented languages such as Java. OODBMS was designed to provide this abstraction to database systems, but suffers a lack of data independence—objects and data created by one programming language can only be written to and read from one OODBMS.

Object-relational mapping offers a "happy medium" between RDBMS and

OODBMS: a level of abstraction is added between the object oriented application and the RDBMS, but it is much more open than OODBMS, and allows for varying languages and incompatibly types. Another advantage of this approach is that Hibernate is used on top of existing RDBMS. As an example, a developer currently using Java and MySQL (a popular combination) would simply need to install and configure Hibernate, and start using it immediately, without the need to change database management systems.