UNIT: 5 SPRING, STRUTS

CS-25 Advanced Java Programming (J2EE)

- INTRODUCTION OF SPRING FRAMEWORK
- SPRING ARCHITECTURE
- SPRING FRAMEWORK DEFINITION
- SPRING & MVC
- SPRING CONTEXT DEFINITION
- INVERSION OF CONTROL (IOC) IN SPRING
- ASPECT ORIENTED PROGRAMMING IN SPRING (AOP)
- UNDERSTANDING STRUTS FRAMEWORK
- COMPARISION WITH MVC USING REQUESTDISPATCHER AND THE EL
- STRUTS FLOW OF CONTROL
- USING PROPERTIES FILES

: ASSIGNMENT 5:

- WHO IS DEVELOPER OF SPRING FRAMEWORK?
- WHO IS DEVELOPER OF STRUTS FRAMEWORK ?
- WHAT IS DI?
- WHAT ARE THE TYPES OF IOC CONTAINER IN SPRING ?
- WHAT IS SPRING CONTEXT ?
- EXPLAIN AOP.
- EXPLAIN STRUTS FRAMEWORK ARCHITECTURE.
- AOP STANDS FOR
- IOC STANDS FOR
- WHAT IS STRUTS ?
- EXPLAIN STRUTS BASIC FLOW.
- EXPLAIN SPRING FRAMEWORK ARCHITECTURE.
- OGNL STANDS FOR
- WHAT IS STRUTS ?
- EXPLAIN STRUTS BASIC FLOW.
- EXPLAIN SPRING FRAMEWORK ARCHITECTURE.
- WRITE NOTE ON INVERSION OF CONTROL.
- WRITE DOWN ADVANTAGE OF SHUTS PROPERTIES FILES.

Introduction to Spring Framework

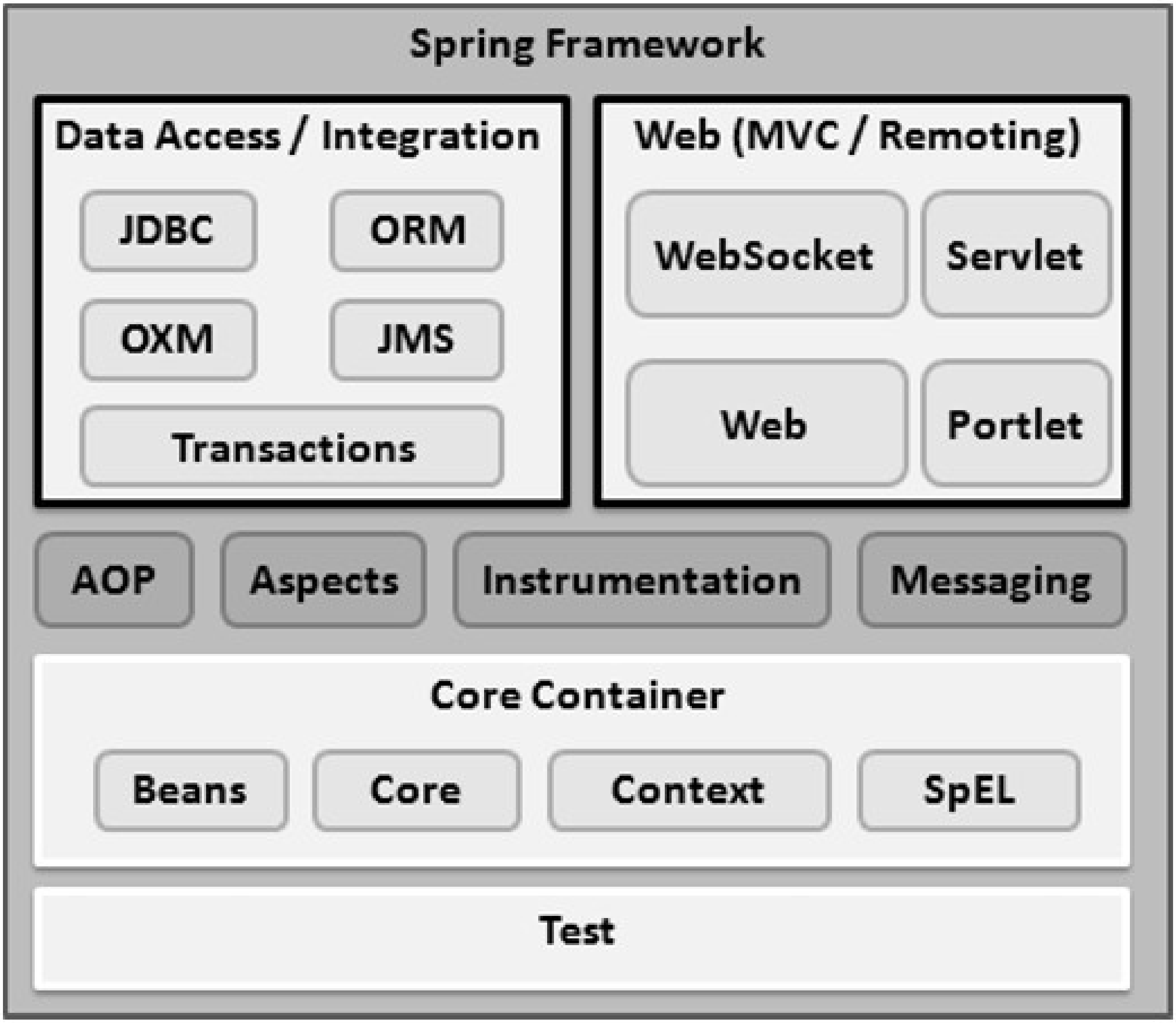
Prior to the advent of Enterprise Java Beans (EJB), Java developers needed to use JavaBeans to create Web applications. Although JavaBeans helped in the development of user interface (UI) components, they were not able to provide services, such as transaction management and security, which were required for developing robust and secure enterprise applications. The advent of EJB was seen as a solution to this problem EJB extends the Java components, such as Web and enterprise components, and provides services that help in enterprise application development. However, developing an enterprise application with EJB was not easy, as the developer needed to perform various tasks, such as creating Home and Remote interfaces and implementing lifecycle callback methods which lead to the complexity of providing code for EJBs Due to this complication, developers started looking for an easier way to develop enterprise applications.

The Spring framework(which is commonly known as Spring) has emerged as a solution to all these complications This framework uses various new techniques such as Aspect-Oriented Programming (AOP), Plain Old Java Object (POJO), and dependency injection (DI), to develop enterprise applications, thereby removing the complexities involved while developing enterprise applications using EJB, Spring is an open source lightweight framework that allows Java EE 7 developers to build simple, reliable, and scalable enterprise applications. This framework mainly focuses on providing various ways to help you manage your business objects. It made the development of Web applications much easier as compared to classic Java frameworks and Application Programming Interfaces (APIs), such as Java database connectivity(JDBC), JavaServer Pages(JSP), and Java Servlet.

The Spring framework can be considered as a collection of sub-frameworks, also called layers, such as Spring AOP. Spring Object-Relational Mapping (Spring ORM). Spring Web Flow, and Spring Web MVC. It is a lightweight application framework used for developing enterprise applications. You can use any of these modules separately while constructing a Web application. The modules may also be grouped together to provide better functionalities in a Web application. Spring framework is loosely coupled because of dependency Injection.

Spring Architecture

Spring could potentially be a one-stop shop for all your enterprise applications. However, Spring is modular, allowing you to pick and choose which modules are applicable to you, without having to bring in the rest. The following section provides details about all the modules available in Spring Framework. The Spring Framework provides about 20 modules which can be used based on an application requirement.



Core Container

- The Core Container consists of the Core, Beans, Context, and Expression Language modules the details of which are as follows −
- The Core module provides the fundamental parts of the framework, including the IoC and Dependency Injection features.
- The Bean module provides BeanFactory, which is a sophisticated implementation of the factory pattern.
- The Context module builds on the solid base provided by the Core and Beans modules and it is a medium to access any objects defined and configured. The ApplicationContext interface is the focal point of the Context module.

- The SpEL module provides a powerful expression language for querying and manipulating an object graph at runtime.

Data Access/Integration

- The Data Access/Integration layer consists of the JDBC, ORM, OXM, JMS and Transaction modules whose detail is as follows −
- The JDBC module provides a JDBC-abstraction layer that removes the need for tedious JDBC related coding.
- The ORM module provides integration layers for popular object-relational mapping APIs, including JPA, JDO, Hibernate, and iBatis.
- The OXM module provides an abstraction layer that supports Object/XML mapping implementations for JAXB, Castor, XMLBeans, JiBX and XStream.
- The Java Messaging Service JMS module contains features for producing and consuming messages.
- The Transaction module supports programmatic and declarative transaction management for classes that implement special interfaces and for all your POJOs.

Web

- The Web layer consists of the Web, Web-MVC, Web-Socket, and Web-Portlet modules the details of which are as follows −
- The Web module provides basic web-oriented integration features such as multipart file-upload functionality and the initialization of the IoC container using servlet listeners and a web-oriented application context.
- The Web-MVC module contains Spring's Model-View-Controller (MVC) implementation for web applications.
- The Web-Socket module provides support for WebSocket-based, two-way communication between the client and the server in web applications.
- The Web-Portlet module provides the MVC implementation to be used in a portlet environment and mirrors the functionality of Web-Servlet module.

Miscellaneous

There are few other important modules like AOP, Aspects, Instrumentation, Web and Test modules the details of which are as follows −

- The AOP module provides an aspect-oriented programming implementation allowing you to define method-interceptors and pointcuts to cleanly decouple code that implements functionality that should be separated.
- The Aspects module provides integration with AspectJ, which is again a powerful and mature AOP framework.
- The Instrumentation module provides class instrumentation support and class loader implementations to be used in certain application servers.
- The Messaging module provides support for STOMP as the WebSocket sub-protocol to use in applications. It also supports an annotation programming model for routing and processing STOMP messages from WebSocket clients.
- The Test module supports the testing of Spring components with JUnit or TestNG frameworks.

Spring Framework definition

The Spring Framework (Spring) is an open-source application framework that provides infrastructure support for developing Java applications. One of the most popular Java Enterprise Edition (Java EE) frameworks, Spring helps developers create high performing applications using plain old Java objects (POJOs).

A framework is a large body of predefined code to which developers can add code to solve a problem in a specific domain. There are many popular Java frameworks including Java Server Faces (JSF), Maven, Hibernate, Struts, and Spring.

Released in June 2003 by Rod Johnson under the Apache 2.0 license, the Spring Framework is hosted by SourceForge.

Why Spring?

Java programs are complex and feature many heavyweight components. Heavyweight means the components are dependent on the underlying operating system (OS) for their appearance and properties.

Spring is considered to be a secure, low-cost and flexible framework. Spring improves coding efficiency and reduces overall application development time because it is lightweight -- efficient at utilizing system resources -- and has a lot of support.

Spring removes tedious configuration work so that developers can focus on writing business logic. Spring handles the infrastructure so developers can focus on the application.

How Spring works

A web application (layered architecture) commonly includes three layers:

Presentation/view layer (UI) - This is the outermost layer which handles the presentation of content and interaction with the user.

Business logic layer - The central layer that deals with the logic of a program.

Data access layer - The deep layer that deals with data retrieval from sources.

Spring & MVC

The Spring Web MVC framework provides Model-View-Controller (MVC) architecture and ready components that can be used to develop flexible and loosely coupled web applications. The MVC pattern results in separating the different aspects of the application (input logic, business logic, and UI logic), while providing a loose coupling between these elements.

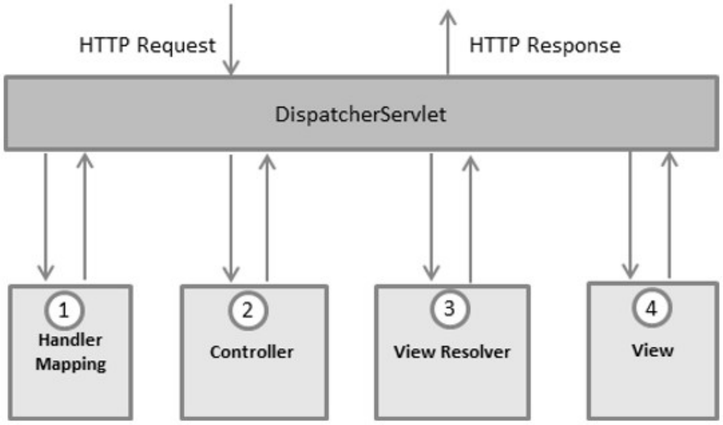
The Model encapsulates the application data and in general they will consist of POJO.

The View is responsible for rendering the model data and in general it generates HTML output that the client's browser can interpret.

The Controller is responsible for processing user requests and building an appropriate model and passes it to the view for rendering.

The DispatcherServlet

The Spring Web model-view-controller (MVC) framework is designed around a DispatcherServlet that handles all the HTTP requests and responses. The request processing workflow of the Spring Web MVC DispatcherServlet is illustrated in the following diagram



Following is the sequence of events corresponding to an incoming HTTP request to DispatcherServlet

After receiving an HTTP request, DispatcherServlet consults the HandlerMapping to call the appropriate Controller.

The Controller takes the request and calls the appropriate service methods based on used GET or POST method. The service method will set model data based on defined business logic and returns view name to the DispatcherServlet.

The DispatcherServlet will take help from ViewResolver to pickup the defined view for the request.

Once view is finalized, The DispatcherServlet passes the model data to the view which is finally rendered on the browser.

All the above-mentioned components, i.e. HandlerMapping, Controller, and ViewResolver are parts of WebApplicationContext which is an extension of the plainApplicationContext with some extra features necessary for web applications.

Spring Context definition

Spring contexts are also called Spring IoC containers, which are responsible for instantiating, configuring, and assembling beans by reading configuration metadata from XML, Java annotations, and/or Java code in the configuration files.

Inversion of Control (IoC) in Spring

Inversion of Control is a principle in software engineering which transfers the control of objects or portions of a program to a container or framework. We most often use it in the context of object-oriented programming. In contrast with traditional programming, in which our custom code makes calls to a library, IoC enables a framework to take control of the flow of a program and make calls to our custom code. To enable this, frameworks use abstractions with additional behaviour built in. If we want to add our own behaviour, we need to extend the classes of the framework or plugin our own classes.

The advantages of this architecture are:

- Decoupling the execution of a task from its implementation.
- Making it easier to switch between different implementations.
- Greater modularity of a program.
- Greater ease in testing a program by isolating a component or mocking its dependencies, and allowing components to communicate through contracts.
- We can achieve Inversion of Control through various mechanisms such as: Strategy design pattern, Service Locator pattern, Factory pattern, and Dependency Injection (DI).

Spring Dependency Injection

Dependency Injection is the main functionality provided by Spring IOC(Inversion of Control). The Spring-Core module is responsible for injecting dependencies through either Constructor or Setter methods. The design principle of Inversion of Control emphasizes keeping the Java classes independent of each other and the container frees them from object creation and maintenance. These classes, managed by Spring, must adhere to the standard definition of Java-Bean. Dependency Injection in Spring also ensures loose coupling between the classes.

Aspect Oriented programming in Spring (AOP)

Aspect Oriented Programming (AOP) compliments OOPs in the sense that it also provides modularity. But the key unit of modularity is aspect than class. AOP breaks the program logic into distinct parts (called concerns). It is used to increase modularity by cross-cutting concerns. A cross-cutting concern is a concern that can affect the whole application and should be centralized in one location in code as possible, such as transaction management, authentication, logging, security etc.

Understanding Struts Framework

Struts is used to create a web applications based on servlet and JSP, is developed by Apache Software Foundation in 2000. Struts depend on the MVC (Model View Controller) framework. Struts application is a genuine web application. Struts are thoroughly useful in building J2EE (Java 2 Platform, Enterprise Edition) applications because struts takes advantage of J2EE design patterns. Struts follows these J2EE design patterns including MVC.

In struts, the composite view manages the layout of its sub-views and can implement a template, making persistent look and feel easier to achieve and customize across the entire application. A composite view is made up by using other reusable sub views such that a small change happens in a sub-view is automatically updated in every composite view. Struts consists of a set of own custom tag libraries. Struts are based on MVC framework which is pattern oriented and includes JSP custom tag libraries. Struts also supports utility classes. The Object Graph Navigation Language (OGNL) is an expression language. It simplifies the accessibility of data stored in the ActionContext. The struts framework sets the ValueStack as the root object of OGNL. Notice that action object is pushed into the ValueStack. We can direct access the action property.

Features of Struts: Struts has the following features:

- Struts encourages good design practices and modelling because the framework is designed with “time-proven” design patterns.
- Struts is almost simple, so easy to learn and use.
- It supports many convenient features such as input validation and internationalization.
- It takes much of the complexity out as instead of building your own MVC framework, you can use struts.
- Struts is very well integrated with J2EE.
- Struts has large user community.
- It is flexible and extensible, it is easy for the existing web applications to adapt the struts framework.
- Struts provide good tag libraries.
- It allows capturing input form data into JavaBean objects called Action forms.
- It also hand over standard error handling both programmatically and declaratively.

Struts Architecture

Struts is famous for its robust Architecture and it is being used for developing small and big software projects. Struts is an open source framework used for developing JEE web applications using Model View Controller (MVC) design pattern. It uses and extends the Java Servlet API to encourage developers adopting MVC architecture. Struts framework provides three key components:

A request handler provided by the application developer that is used to be mapped to a particular URI.

A response handler which is used to transfer the control to another resource which will be responsible for completing the response.

A tag library which helps developers to create the interactive form based applications with server pages.

Overview of the Struts Framework

The Struts framework is composed of approximately 300 classes and interfaces which are organized in about 12 top level packages. Along with the utility and helper classes framework also provides the classes and interfaces for working with controller and presentation by the help of the custom tag libraries. It is entirely on to us which model we want to choose. The view of the Struts architecture is given below:

The Struts Controller Components:

Whenever a user request for something, then the request is handled by the Struts Action Servlet. When the ActionServlet receives the request, it intercepts the URL and based on the Struts Configuration files, it gives the handling of the request to the Action class. Action class is a part of the controller and is responsible for communicating with the model layer.

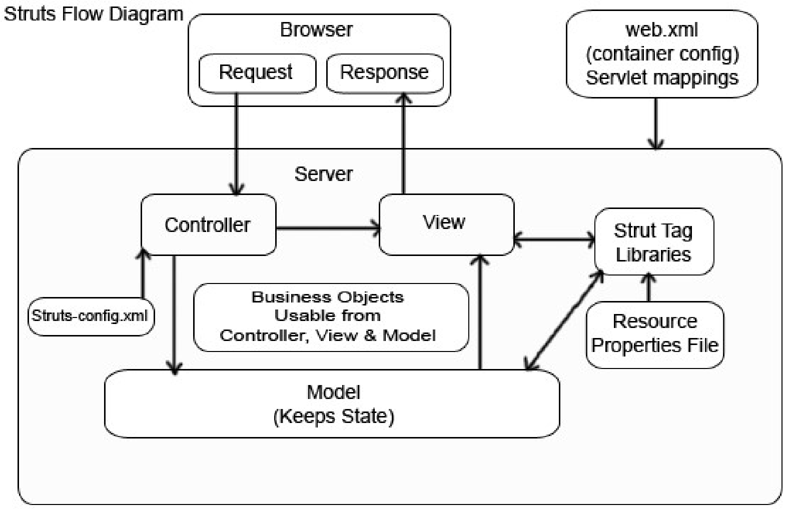
The Struts View Components:

The view components are responsible for presenting information to the users and accepting the input from them. They are responsible for displaying the information provided by the model components. Mostly we use the JavaServer Pages (JSP) for the view presentation. To extend the capability of the view we can use the Custom tags, java script etc.

The Struts model component:

The model component provides a model of the business logic behind a Struts program. It provides interfaces to databases or back- ends systems. Model components are generally a java class. There is not any such defined format for a Model component, so it is possible for us to reuse Java codes which are written for other projects. We should choose the model according to our client requirement.

Struts Flow of Control



The process flow is shown below.

web.xml: Whenever the container gets start up the first work it does is to check the web.xml file and determine what struts action Servlets exist. The container is responsible for mapping all the file requests to the correct action Servlet.
Request: This is the second step performed by the container after checking the web.xml file. In this the user submits a form within a browser and the request is intercepted by the controller.
Controller: This is the heart of the container. Most Struts application will have only one controller that is ActionServlet which is responsible for directing several Actions. The controller determines what action is required and sends the information to be processed by an action Bean. The key advantage of having a controller is its ability to control the flow of logic through the highly controlled, centralized points.
struts-config.xml: Struts has a configuration file to store mappings of actions. By using this file there is no need to hard code the module which will be called within a component. The one more responsibility of the controller is to check the struts-config.xml file to determine which module to be called upon an action request. Struts only reads the struts-config.xml file upon start up.
Model: The model is basically a business logic part which takes the response from the user and stores the result for the duration of the process. This is a great place to perform the pre-processing of the data

received from request. It is possible to reuse the same model for many page requests. Struts provides the ActionForm and the Action classes which can be extended to create the model objects.

View: The view in Struts framework is mainly a JSP page which is responsible for producing the output to the user.

Struts tag libraries: These are struts components helps us to integrate the struts framework within the project's logic. These struts tag libraries are used within the JSP page. This means that the controller and the model part can't make use of the tag library but instead use the struts class library for strut process control.

Property file: It is used to store the messages that an object or page can use. Properties files can be used to store the titles and other string data. We can create many property files to handle different languages.

Business objects: It is the place where the rules of the actual project exist. These are the modules which just regulate the day-to-day site activities.

Response: This is the output of the View JSP object.

The Struts.properties File

This configuration file provides a mechanism to change the default behavior of the framework. Actually, all the properties contained within the struts.properties configuration file can also be configured in the web.xml using the init-param, as well using the constant tag in the struts.xml configuration file. But, if you like to keep the things separate and more struts specific, then you can create this file under the folder WEB-INF/classes.

The values configured in this file will override the default values configured in default.properties which is contained in the struts2-core-x.y.z.jar distribution. There are a couple of properties that you might consider changing using the struts.properties file.

```
### When set to true, Struts will act much more friendly for developers
struts.devMode = true

### Enables reloading of internationalization files
struts.i18n.reload = true

### Enables reloading of XML configuration files
struts.configuration.xml.reload = true

### Sets the port that the server is run on
struts.url.http.port = 8080
```

Here any line starting with hash (#) will be assumed as a comment and it will be ignored by Struts.