

Spring Framework

The Spring Framework provides a comprehensive programming and configuration model for modern Java-based enterprise applications - on any kind of deployment platform.

A key element of **Spring** is infrastructural support at the application level: Spring focuses on the "plumbing" of enterprise applications so that teams can focus on application-level business logic, without unnecessary ties to specific deployment environments.

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.

Features

- **Lightweight**

The Spring Framework is very lightweight with respect to its size and functionality. This is due to its POJO implementation, which doesn't force it to inherit any class or implement any interfaces.

- **Aspect-Oriented Programming (AOP)**

This is an important part of the Spring Framework. Aspect-Oriented Programming is used for separating cross-cutting concerns (for example, logging, security, etc.) from the business logic of the application. In the coming articles, you will be learning about this in greater detail.

- **Transaction Management**

This is used to unify several transaction management APIs and is used to coordinate transactions for Java objects. Also, it is not tied to the J2EE environment and is used with **containerless** environments.

- **Container**

The Spring framework designs and manages the lifecycle and configurations of application objects.

- **Dependency Injection**

This feature of the Spring Framework allows you to develop loosely coupled applications. Therefore, the unit testing of these loosely coupled applications becomes easier. This also allows the developer to swap out some of the modules according to its need.

- **Integration With Other Frameworks**

A great thing about this framework is that it doesn't try to solve the problems that have already been solved. It just tries to integrate them with its framework, which provides a solution to greater problems. For example, this could include IBATIS, Hibernate, Toplink, etc.

Features of Spring 5.0

Now, major enhancements are done since the introduction of Spring framework by Rod Johnson in 2003. Several versions have been released after its first. As of now, Spring 5.0x versions are on the market. So, let's see the major upgraded features and enhancements of Spring 5.0 -:

1. **JDK 8 Plus 9 and Java EE 7 Baseline**

2. Removal of Classes, Methods, and Packages
3. Core Container Enhancements
4. General Web Improvements

Advantages of Spring Framework:

1. Solving difficulties of Enterprise application development

Spring is solving the difficulties of development of complex applications, it provides Spring Core, Spring IoC and Spring AOP for integrating various components of business applications.

2. Support Enterprise application development through POJOs

Spring supports development of Enterprise application development using the POJO classes which removes the need of importing heavy Enterprise container during development. This makes application testing much easier.

3. Easy integration other frameworks

Spring designed to be used with all other frameworks of Java, you can use ORM, Struts, Hibernate and other frameworks of Java together. Spring framework do not impose any restriction on the frameworks to be used together.

4. Application Testing

Spring Container can be used to develop and run test cases outside enterprise container which makes testing much easier.

5. Modularity

Spring framework is modular framework and it comes with many modules such as Spring MVC, Spring ORM, Spring JDBC, Spring Transactions etc. which can be used as per application requirement in modular fashion.

6. Spring Transaction Management

Spring Transaction Management interface is very flexible it can be configured to use local transactions in small application which can be scaled to JTA for global transactions.

Setup

Step 1 - Setup Java Development Kit (JDK)

You can download the latest version of SDK from Oracle's Java site – [Java SE Downloads](#). You will find instructions for installing JDK in downloaded files, follow the given instructions to install and configure the setup. Finally set PATH and JAVA_HOME environment variables to refer to the directory that contains java and javac, typically java_install_dir/bin and java_install_dir respectively.

If you are running Windows and have installed the JDK in C:\jdk1.6.0_15, you would have to put the following line in your C:\autoexec.bat file.

```
set PATH=C:\jdk1.6.0_15\bin;%PATH%
```

```
set JAVA_HOME=C:\jdk1.6.0_15
```

Alternatively, on Windows NT/2000/XP, you will have to right-click on My Computer, select Properties → Advanced → Environment Variables. Then, you will have to update the PATH value and click the OK button.

On Unix (Solaris, Linux, etc.), if the SDK is installed in /usr/local/jdk1.6.0_15 and you use the C shell, you will have to put the following into your .cshrc file.

```
setenv PATH /usr/local/jdk1.6.0_15/bin:$PATH
```

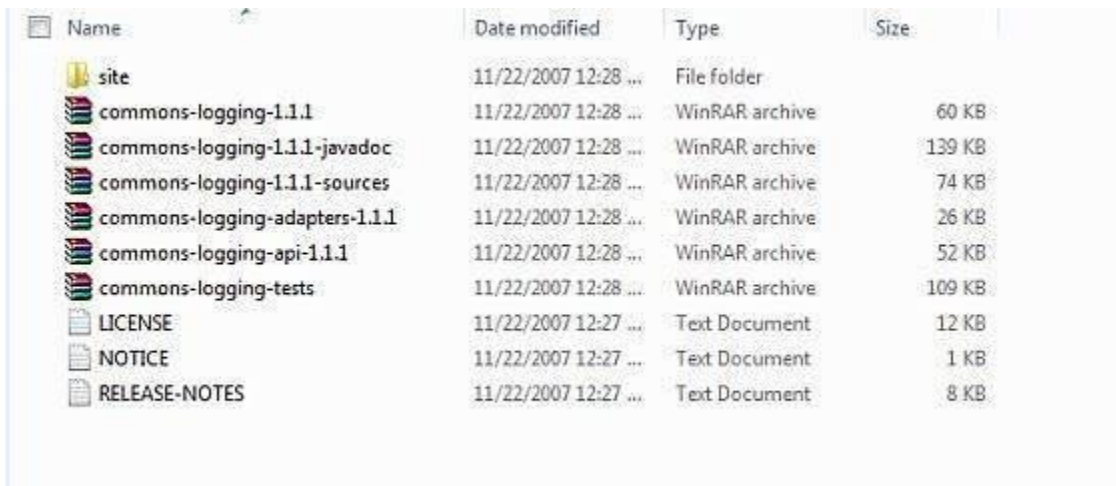
```
setenv JAVA_HOME /usr/local/jdk1.6.0_15
```

Alternatively, if you use an Integrated Development Environment (IDE) like Borland JBuilder, Eclipse, IntelliJ IDEA, or Sun ONE Studio, you will have to compile and run a simple program to confirm that the IDE knows where you have installed Java.

Otherwise, you will have to carry out a proper setup as given in the document of the IDE.

Step 2 - Install Apache Common Logging API

You can download the latest version of Apache Commons Logging API from <https://commons.apache.org/logging/>. Once you download the installation, unpack the binary distribution into a convenient location. For example, in C:\commons-logging-1.1.1 on Windows, or /usr/local/commons-logging-1.1.1 on Linux/Unix. This directory will have the following jar files and other supporting documents, etc.



Name	Date modified	Type	Size
site	11/22/2007 12:28 ...	File folder	
commons-logging-1.1.1	11/22/2007 12:28 ...	WinRAR archive	60 KB
commons-logging-1.1.1-javadoc	11/22/2007 12:28 ...	WinRAR archive	139 KB
commons-logging-1.1.1-sources	11/22/2007 12:28 ...	WinRAR archive	74 KB
commons-logging-adapters-1.1.1	11/22/2007 12:28 ...	WinRAR archive	26 KB
commons-logging-api-1.1.1	11/22/2007 12:28 ...	WinRAR archive	52 KB
commons-logging-tests	11/22/2007 12:28 ...	WinRAR archive	109 KB
LICENSE	11/22/2007 12:27 ...	Text Document	12 KB
NOTICE	11/22/2007 12:27 ...	Text Document	1 KB
RELEASE-NOTES	11/22/2007 12:27 ...	Text Document	8 KB

Make sure you set your CLASSPATH variable on this directory properly otherwise you will face a problem while running your application.

Step 3 - Setup Eclipse IDE

All the examples in this tutorial have been written using Eclipse IDE. So we would suggest you should have the latest version of Eclipse installed on your machine.

To install Eclipse IDE, download the latest Eclipse binaries from <https://www.eclipse.org/downloads/>. Once you download the installation, unpack the binary distribution into a convenient location. For example, in C:\eclipse on Windows, or /usr/local/eclipse on Linux/Unix and finally set PATH variable appropriately.

Eclipse can be started by executing the following commands on Windows machine, or you can simply double-click on eclipse.exe

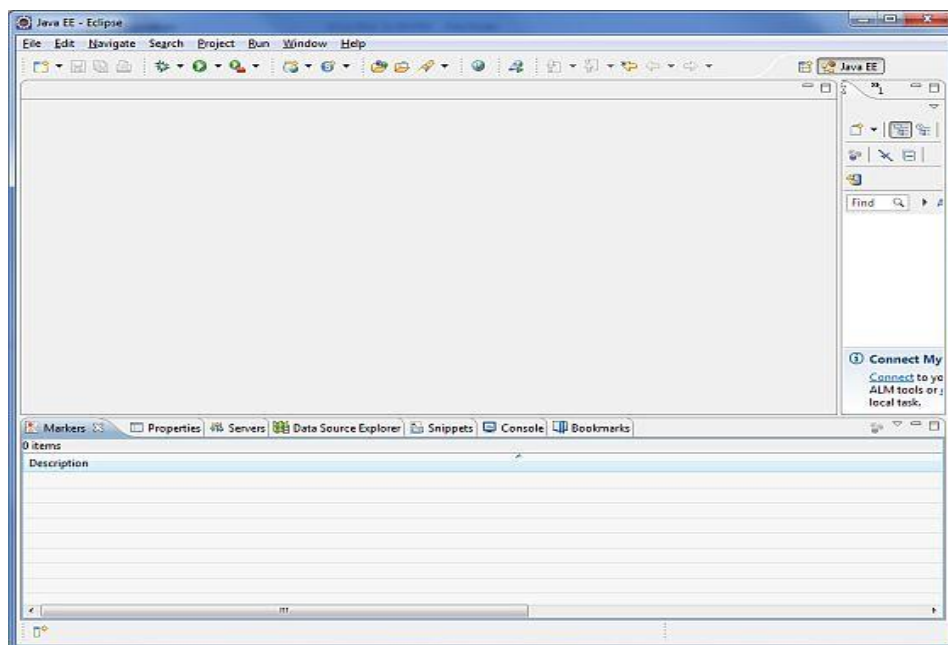
%C:\eclipse\eclipse.exe

Eclipse can be started by executing the following commands on Unix (Solaris, Linux, etc.) machine –

\$/usr/local/eclipse/eclipse

After a successful startup, if everything is fine then it should display the following result

–



Step 4 - Setup Spring Framework Libraries

Now if everything is fine, then you can proceed to set up your Spring framework.

Following are the simple steps to download and install the framework on your machine.

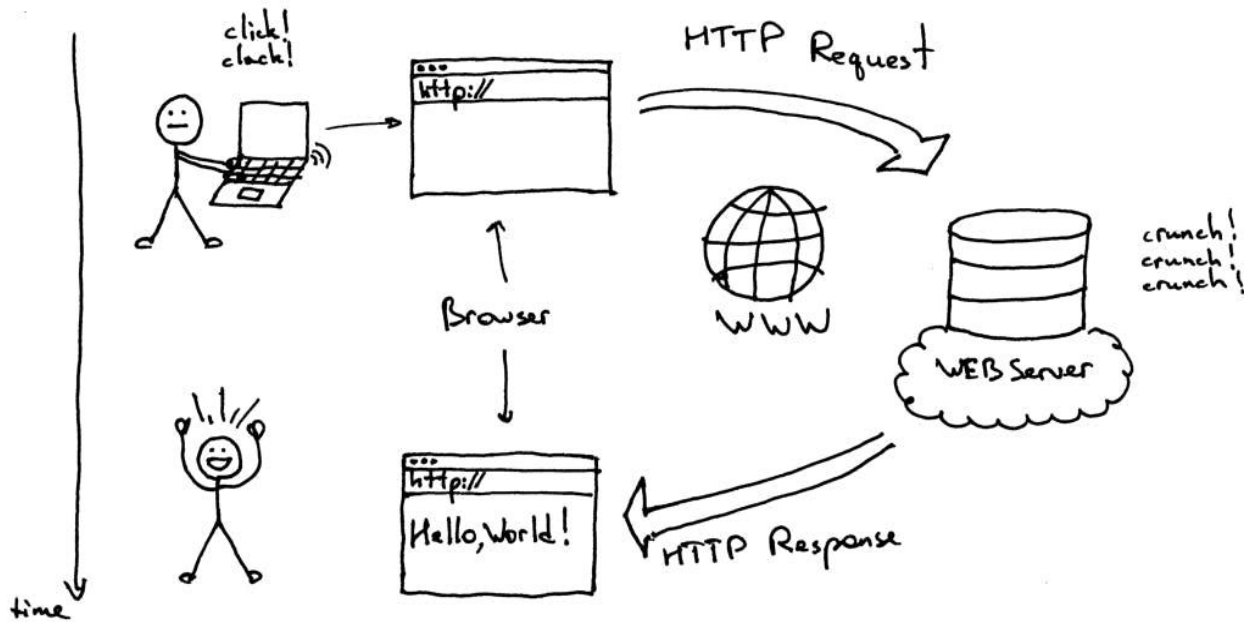
- Make a choice whether you want to install Spring on Windows or Unix, and then proceed to the next step to download .zip file for Windows and .tar.gz file for Unix.
- Download the latest version of Spring framework binaries from <https://repo.spring.io/release/org/springframework/spring>.
- At the time of developing this tutorial, **spring-framework-4.1.6.RELEASE-dist.zip** was downloaded on Windows machine. After the downloaded file was unzipped, it gives the following directory structure inside E:\spring.

Name	Date modified	Type	Size
docs	4/22/2015 2:44 PM	File folder	
libs	4/22/2015 2:45 PM	File folder	
schema	4/22/2015 2:45 PM	File folder	
license	4/22/2015 2:42 PM	Text Document	15 KB
notice	4/22/2015 2:42 PM	Text Document	1 KB
readme	4/22/2015 2:42 PM	Text Document	1 KB

You will find all the Spring libraries in the directory **E:\spring\libs**. Make sure you set your CLASSPATH variable on this directory properly otherwise you will face a problem while running your application. If you are using Eclipse, then it is not required to set CLASSPATH because all the setting will be done through Eclipse.

Servlet

A servlet is a **Java Programming** language class that is used to extend the capabilities of servers that host applications accessed by means of a request-response programming model. Although servlets can respond to any type of request, they are commonly used to extend the applications hosted by web servers. It is also a web component that is deployed on the server to create a dynamic web page.

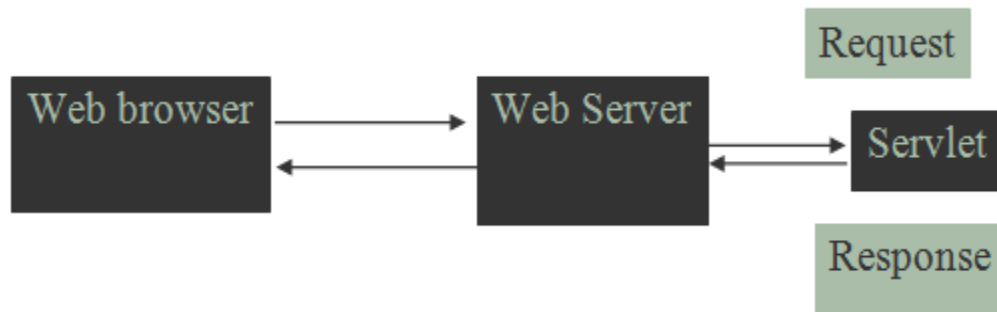


In this figure you can see, a client sends a request to the server and the server generates the response, analyses it and sends the response to the client.

Servlet Architecture

The **architecture**, here, discusses the communication interface, protocol used, requirements of client and server, the programming with the languages and software involved. Basically, it performs the below-mentioned tasks.

- First, it reads the explicit data sent by the clients (browsers). This data can include an HTML form on a Web page, an applet or a custom HTTP client program. It also reads implicit HTTP request data sent by the clients (browsers). This can include cookies, media types and compression schemes the browser understands, and so forth.
- After that, the servlet processes the data and generate the results. This process may require communicating to a database, executing an RMI, invoking a Web service, or computing the response directly.

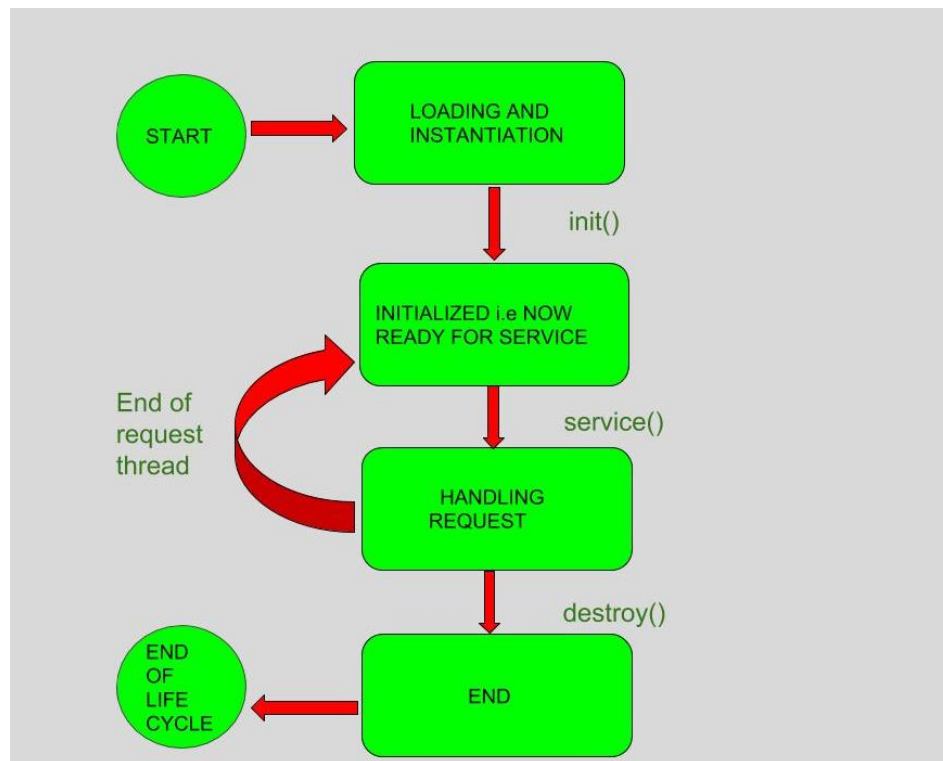


- After processing, it sends the explicit data (i.e., the document) to the clients (browsers). This document can be sent in a variety of formats, including text (HTML or XML), binary (GIF images), or Excel formats.
- Finally, it also sends the implicit HTTP response to the clients (browsers). This includes telling the browsers or other clients what type of document is being returned.

Servlet Life Cycle

The Servlet life cycle mainly includes the following four stages,

- Loading a Servlet
 - Initializing the Servlet
 - Request handling
 - Destroying the Servlet
1. When the web server (e.g. Apache Tomcat) starts up, the servlet container deploy and loads all the servlets.
 2. The servlet is initialized by calling the `init()` method. The `Servlet.init()` method is called by the Servlet container to indicate that this Servlet instance is instantiated successfully and is about to put into service.



3. The servlet then calls *service()* method to process a client's request. This method is invoked to inform the Servlet about the client requests.
4. The servlet is terminated by calling the *destroy()*.
5. The *destroy()* method runs only once during the lifetime of a Servlet and signals the end of the Servlet instance.

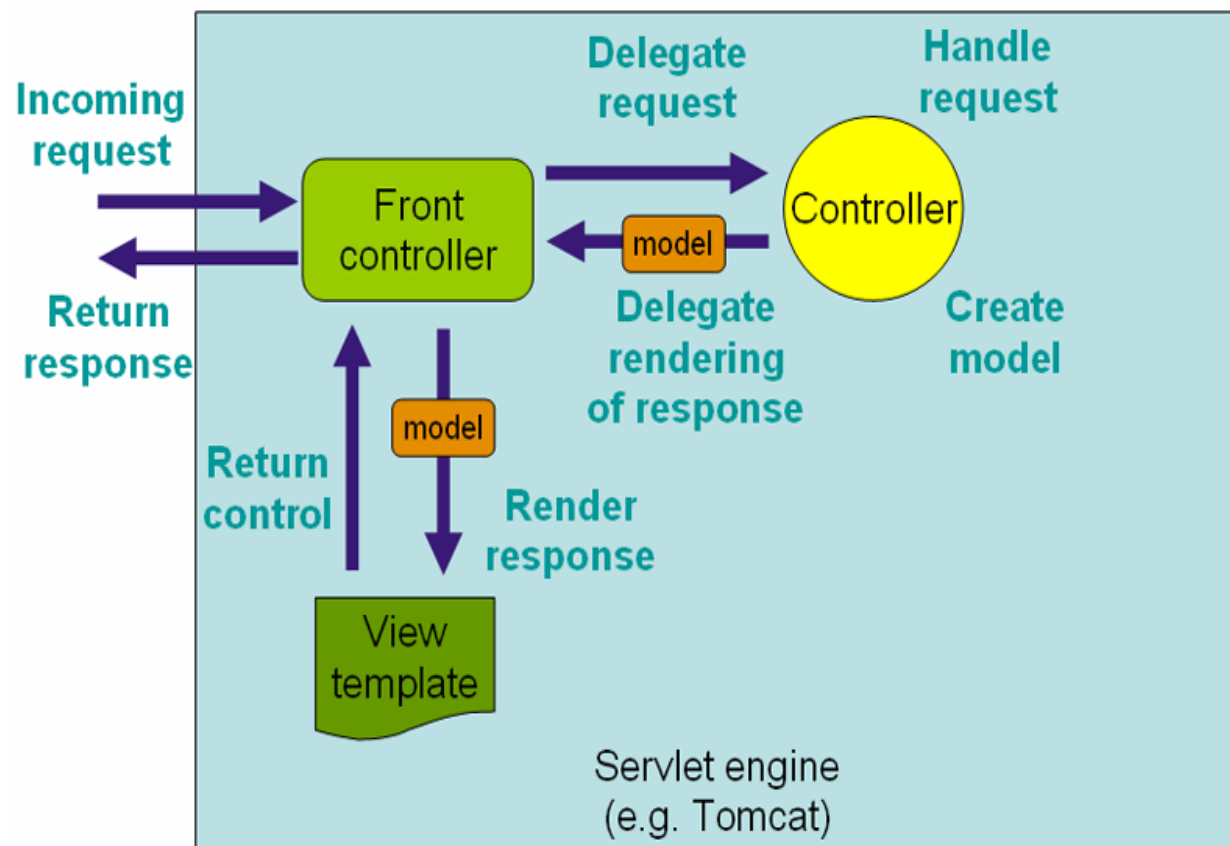
init() and *destroy()* methods are called only once. Finally, a servlet is garbage collected by the garbage collector of the JVM. So this concludes the life cycle of a servlet. Now, let me guide you through the steps of creating java servlets.

The DispatcherServlet

Spring's web MVC framework is, like many other web MVC frameworks, request-driven, designed around a central servlet that dispatches requests to controllers and offers other functionality that facilitates the development of web applications.

Spring's DispatcherServlet however, does more than just that. It is completely integrated with the Spring IoC container and as such allows you to use every other feature that Spring has.

The request processing workflow of the Spring Web MVC DispatcherServlet is illustrated in the following diagram. The pattern-savvy reader will recognize that the DispatcherServlet is an expression of the “Front Controller” design pattern (this is a pattern that Spring Web MVC shares with many other leading web frameworks).



The requesting processing workflow in Spring Web MVC (high level).

The DispatcherServlet is an actual Servlet (it inherits from the HttpServlet base class), and as such is declared in the web.xml of your web application. You need to map requests that you want the DispatcherServlet to handle, by using a URL mapping in the same web.xml file. This is standard J2EE servlet configuration; the following example shows such a DispatcherServlet declaration and mapping:

```
<web-app>

<servlet>
<servlet-name>example</servlet-name>
<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-
class>
<load-on-startup>1</load-on-startup>
</servlet>

<servlet-mapping>
<servlet-name>example</servlet-name>
<url-pattern>*.form</url-pattern>
</servlet-mapping>

</web-app>
```

In the preceding example, all requests ending with .form will be handled by the example DispatcherServlet.

Reference:

- Oracle
- Edureka.co
- Javatpoint