


[ANDROID](#)
[CORE JAVA](#)
[DESKTOP JAVA](#)
[ENTERPRISE JAVA](#)
[JAVA BASICS](#)
[JVM LANGUAGES](#)
[SOFTWARE DEVELOPMENT](#)
[DEVOPS](#)
[Home](#) » [Enterprise Java](#) » [spring](#) » [MVC](#) » [Spring MVC File Download Example](#)

## ABOUT YATIN BATRA



Yatin has graduated in Electronics & Telecommunication. During his studies, he has been involved with a large number of projects ranging from programming and software engineering to telecommunications analysis. He works as a software developer in the information technology sector where he is mainly involved with projects based on Java and J2EE technologies platform.



## Spring MVC File Download Example

Posted by: Yatin Batra in MVC November 28th, 2017

Hello readers. Spring framework provides an out of box support for the file download functionality from the server to a local machine. In this tutorial, we will show you how to implement the file download functionality with the Spring Mvc framework. To handle the file download capability in a web application, we will use the

```
HttpServletResponse
```

to directly write a file to the

```
ServletOutputStream
```

## Want to master Spring Framework ?

Subscribe to our newsletter and download the Spring Framework Cookbook [right now!](#)

In order to help you master the leading and innovative Java framework, we have compiled a kick-ass guide with all its major features and use cases! Besides studying them online you may download the eBook in PDF format!

**Email address:**

[Sign up](#)

## Table Of Contents

1. Introduction
  - 1.1 Spring Framework
  - 1.2 Spring Framework's Support for File Download
2. Spring Mvc File Download Example
  - 2.1 Tools Used
  - 2.2 Project Structure
  - 2.3 Project Creation
  3. Application Building
    - 3.1 Database & Table Creation
    - 3.2 Maven Dependencies
    - 3.3 Java Class Creation
    - 3.4 Configuration Files
    - 3.5 Creating JSP Views
  4. Run the Application
  5. Project Demo
  6. Conclusion
  7. Download the Eclipse Project

## NEWSLETTER

**184,005** insiders are already receiving weekly updates and complimentary whitepapers!

**Join them now** to gain [access](#) to the latest news in Java as well as insights about Android, Groovy and other related technologies!

**Email address:**

☒ Receive Java & Developer Area updates

[Sign up](#)

## JOIN US



With **1,500** unique and interesting articles placed at your fingertips, you can stay updated on the latest in Java development. So if you are looking for unique and interesting content to check out, our **JCG** partners provide a **guest writer** for Java Code Geeks to showcase your writing skills!

## 1.1 Spring Framework

- Spring is an open-source framework created to address the complexity of an enterprise application development
- One of the chief advantages of the Spring framework is its layered architecture, which allows developer to be selective about which of its components they can use while providing a cohesive framework for

J2EE

application development

- Spring framework provides support and integration to various technologies for e.g.:
  - Support for Transaction Management
  - Support for interaction with the different databases
  - Integration with the Object Relationship frameworks for e.g. Hibernate, iBatis etc
  - Support for Dependency Injection which means all the required dependencies will be resolved with the help of containers
  - Support for

REST

style web-services

### 1.1.1 Spring Mvc Framework

Model-View-Controller (MVC) is a well-known design pattern for designing the GUI based applications. It mainly decouples the business logic from UI by separating the roles of **Model**, **View**, and **Controller** in an application. This pattern divides the application into three components to separate the internal representation of the information from the way it is being presented to the user. The three components are:

- Model (M): Model's responsibility is to manage the application's data, business logic, and the business rules. It is a

POJO

class which encapsulates the application data given by the controller

- View (V): A view is an output representation of the information, such as displaying information or reports to the user either as a text-form or as charts. Views are usually the

JSP

templates written with Java Standard Tag Library (

JSTL

)

- Controller (C): Controller's responsibility is to invoke the Models to perform the business logic and then update the view based on the model's output. In spring framework, the controller part is played by the Dispatcher Servlet

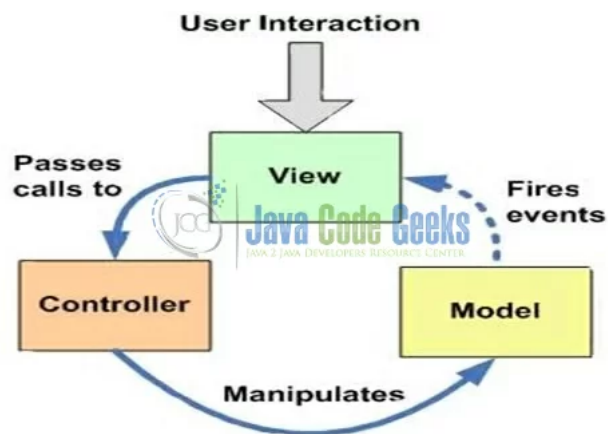


Fig. 1: Model View Controller (MVC) Overview

## 1.2 Spring Framework's Support for File Download

Spring Mvc framework provides several approaches for downloading a file in the Spring Mvc web-application. These are:

HttpServletResponse

- : Developers can use the

HttpServletResponse

object to directly write a file to the

ServletOutputStream

for the downloading purpose

ResponseEntity<InputStreamResource>

object which will be wrapped in a

ResponseEntity

from the Spring controller handler's method

ResponseEntity<ByteArrayResource>

- : Developers can also return a file as a

ByteArrayResource

object wrapped in the

ResponseEntity

The following picture depicts the workflow of the sample application we are going to build in this tutorial.

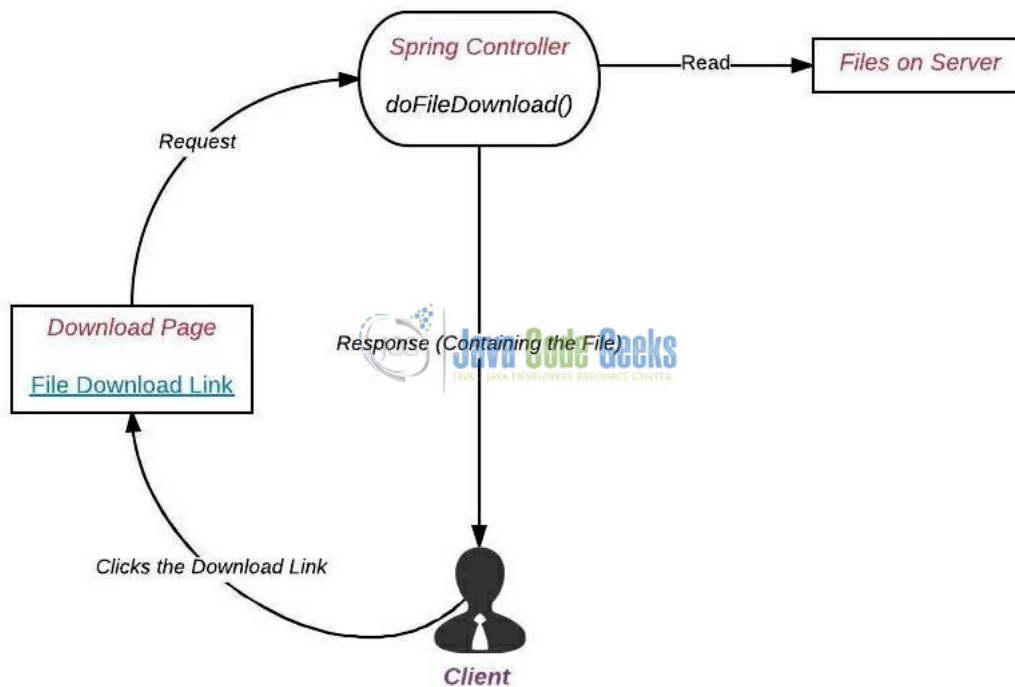


Fig. 2: Spring Mvc File Download Application Workflow

Now, open up the Eclipse Ide and let's start building the application!

## 2. Spring Mvc File Download Example

Below are the steps involved in developing this application.

### 2.1 Tools Used

We are using Eclipse Kepler SR2, JDK 8 and Maven. Having said that, we have tested the code against JDK 1.7 and it works well.

### 2.2 Project Structure

Firstly, let's review the final project structure, in case you are confused about where you should create the corresponding files or folder later!

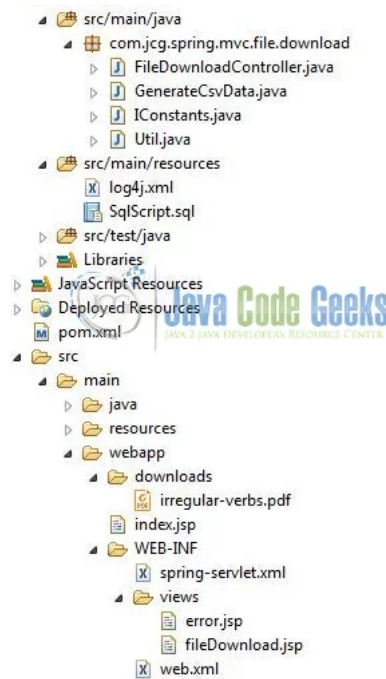


Fig. 3: Application Project Structure

## 2.3 Project Creation

This section will demonstrate on how to create a Java-based Maven project with Eclipse. In Eclipse IDE, go to

File -> New -> Maven Project

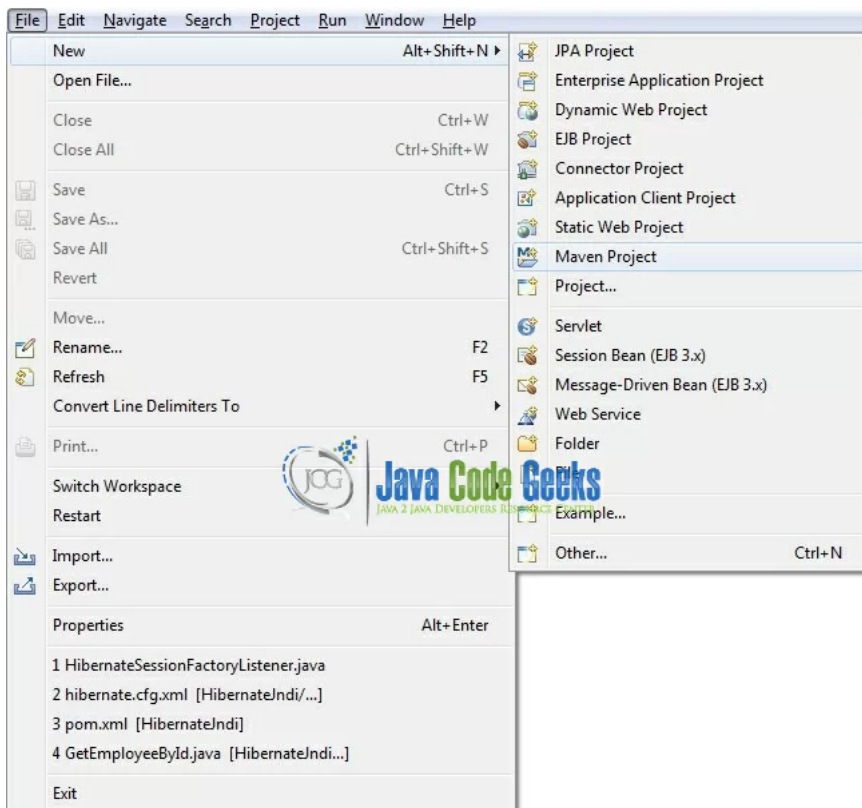


Fig. 4: Create Maven Project

In the New Maven Project window, it will ask you to select project location. By default, 'Use default workspace location' will be selected. Just click on next button to proceed.

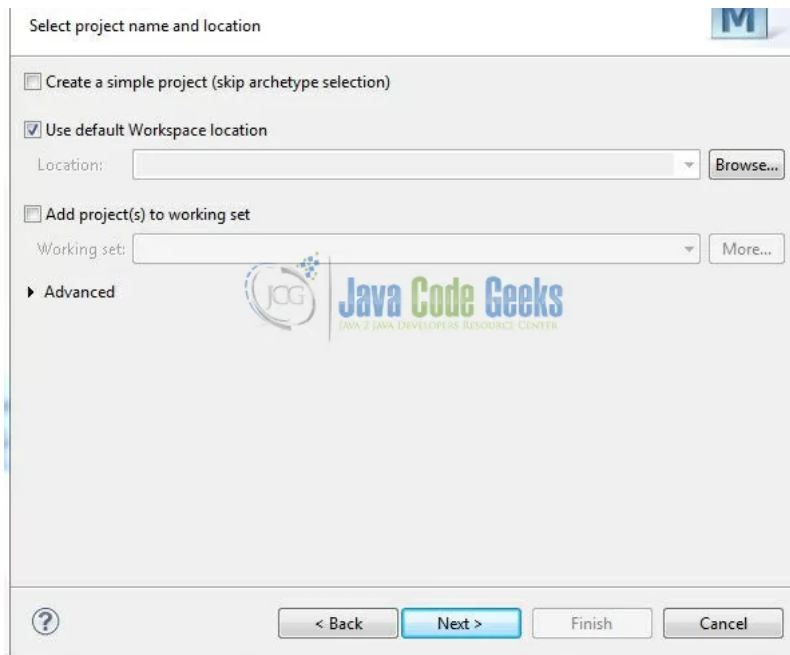


Fig. 5: Project Details

Select the *Maven Web App* Archetype from the list of options and click next.

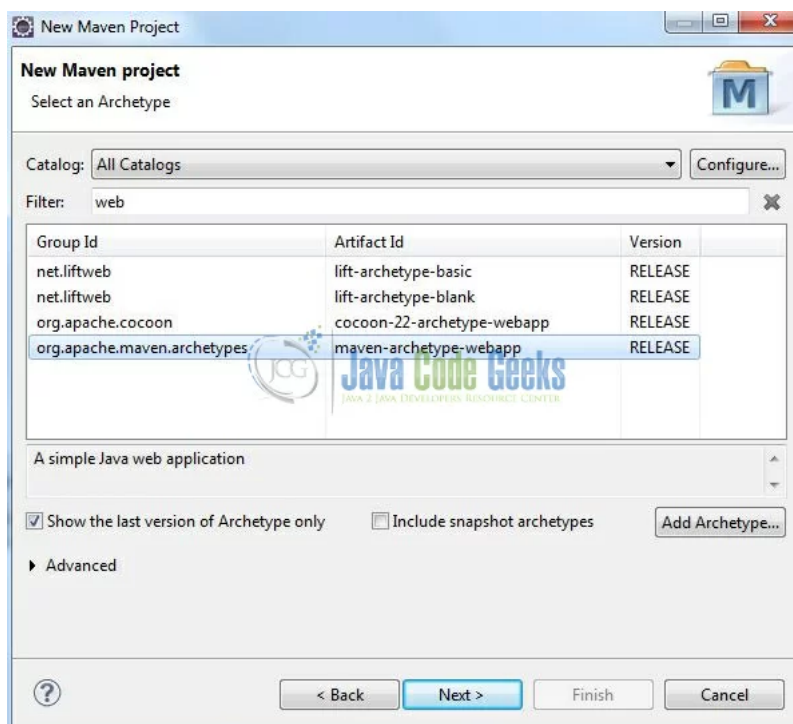


Fig. 6: Archetype Selection

It will ask you to 'Enter the group and the artifact id for the project'. We will input the details as shown in the below image. The version number will be by default:

0.0.1-SNAPSHOT

Fig. 7: Archetype Parameters

Click on Finish and the creation of a maven project is completed. If you observe, it has downloaded the maven dependencies and a

pom.xml

file will be created. It will have the following code:

pom.xml

```

1 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xs
2   <modelVersion>4.0.0</modelVersion>
3   <groupId>SpringMvcDownloadFile</groupId>
4   <artifactId>SpringMvcDownloadFile</artifactId>
5   <version>0.0.1-SNAPSHOT</version>
6   <packaging>war</packaging>
7 </project>

```

We can start adding the dependencies that developers want like Spring Mvc, Servlet Api, MySQL, and

Log4j

etc. Let's start building the application!

## 3. Application Building

Below are the steps involved in developing this application.

### 3.1 Database & Table Creation

The following MySQL script is used to create a database called

filedownload

with table:

exam\_result

. Open the MySQL or the workbench terminal and execute the

SQL

script:

```

01 CREATE DATABASE IF NOT EXISTS filedownload;
02
03 USE filedownload;
04
05 CREATE TABLE exam_result (
06   student_id INTEGER NOT NULL,
07   student_name VARCHAR(30) NOT NULL,
08   student_dob DATE NOT NULL,
09   student_percentage double NOT NULL
10 );
11
12 INSERT INTO exam_result (student_id, student_name, student_dob, student_percentage) VALUES (101, 'Harry Pot
13 INSERT INTO exam_result (student_id, student_name, student_dob, student_percentage) VALUES (102, 'Java Code
14 INSERT INTO exam_result (student_id, student_name, student_dob, student_percentage) VALUES (103, 'Hermione

```

```
19 | SELECT * FROM exam_result;
```

If everything goes well, the database and the table will be shown in the MySQL Workbench.

```
mysql> DESC exam_result;
```

Field	Type	Null	Key	Default	Extra
student_id	int(11)	NO		NULL	
student_name	varchar(30)	NO		NULL	
student_dob	date	NO		NULL	
student_percentage	double	NO		NULL	

4 rows in set (0.05 sec)

```
mysql> SELECT * FROM exam_result;
```

student_id	student_name	student_dob	student_percentage
101	Harry Potter	1993-02-01	92
102	Java Code Geek	1987-02-03	62
103	Hermione Granger	1985-02-01	76
104	Lucifer Morningstar	1965-02-01	83

4 rows in set (0.00 sec)

Fig. 8: Database & Table Creation

## 3.2 Maven Dependencies

In this example, we are using the most stable Spring web-mvc, MySQL, and

```
Log4j
```

version in order to set-up the file download functionality. The **updated** file will have the following code:

pom.xml

```
01 <?xml version="1.0" encoding="UTF-8"?>
02 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
03   <modelVersion>4.0.0</modelVersion>
04   <groupId>SpringMvcDownloadFile</groupId>
05   <artifactId>SpringMvcDownloadFile</artifactId>
06   <packaging>war</packaging>
07   <version>0.0.1-SNAPSHOT</version>
08   <name>SpringMvcDownloadFile Maven Webapp</name>
09   <url>http://maven.apache.org</url>
10   <dependencies>
11     <!-- Spring Framework Dependencies -->
12     <dependency>
13       <groupId>org.springframework</groupId>
14       <artifactId>spring-webmvc</artifactId>
15       <version>4.3.11.RELEASE</version>
16     </dependency>
17     <!-- Servlet API Dependency -->
18     <dependency>
19       <groupId>javax.servlet</groupId>
20       <artifactId>servlet-api</artifactId>
21       <version>3.0-alpha-1</version>
22     </dependency>
23     <dependency>
24       <groupId>javax.servlet</groupId>
25       <artifactId>jstl</artifactId>
26       <version>1.2</version>
27     </dependency>
28     <dependency>
29       <groupId>javax.servlet.jsp</groupId>
30       <artifactId>jsp-api</artifactId>
31       <version>2.1</version>
32     </dependency>
33     <!-- MySQL Connector Java Dependency -->
34     <dependency>
35       <groupId>mysql</groupId>
36       <artifactId>mysql-connector-java</artifactId>
37       <version>5.1.40</version>
38     </dependency>
39     <!-- Log4J Dependency -->
40     <dependency>
41       <groupId>log4j</groupId>
42       <artifactId>log4j</artifactId>
43       <version>1.2.17</version>
44     </dependency>
45   </dependencies>
46   <build>
47     <finalName>${project.artifactId}</finalName>
48   </build>
49 </project>
```

## 3.3 Java Class Creation

Let's create the required Java files. Right-click on

```
src/main/java
```

folder,

```
New -> Package
```

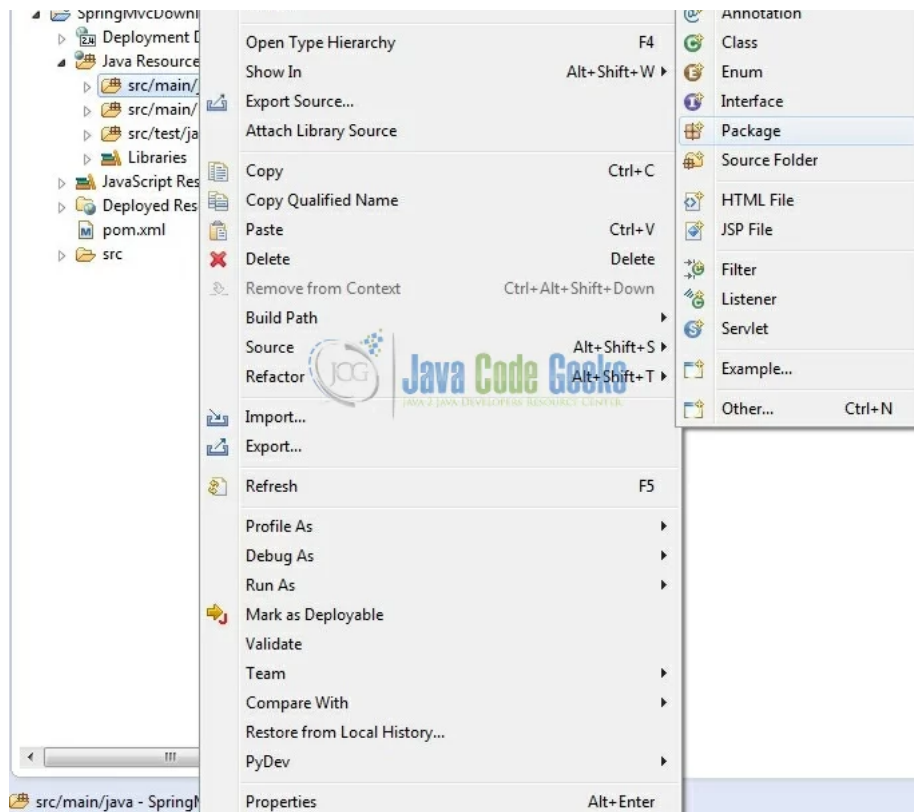


Fig. 9: Java Package Creation

A new pop window will open where we will enter the package name as:

```
com.jcg.spring.mvc.file.download
```

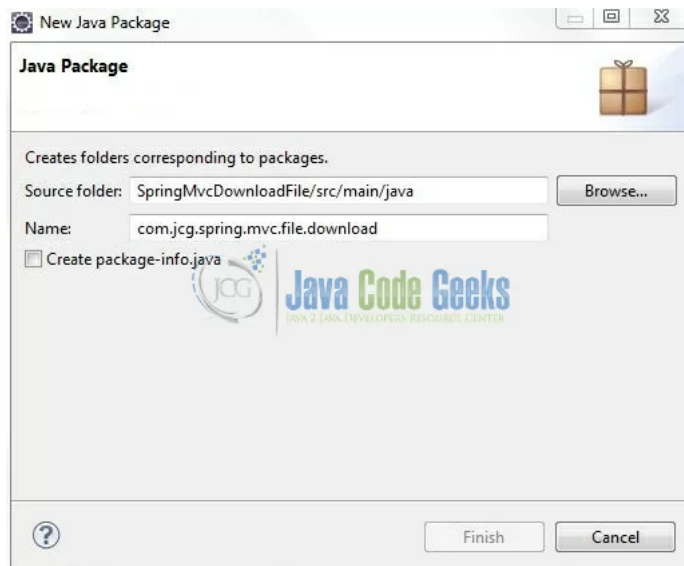


Fig. 10: Java Package Name (com.jcg.spring.mvc.file.download)

Once the package is created, we will need to create the implementation class. Right-click on the newly created package,

```
New -> Class
```



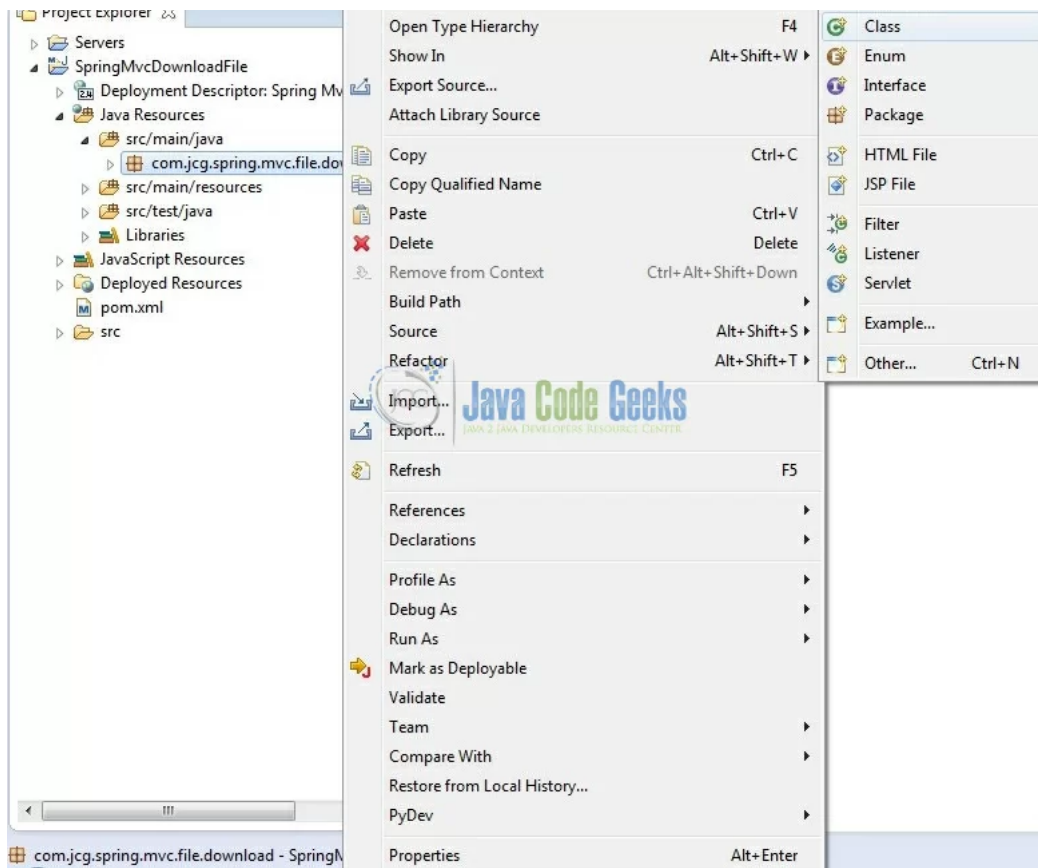


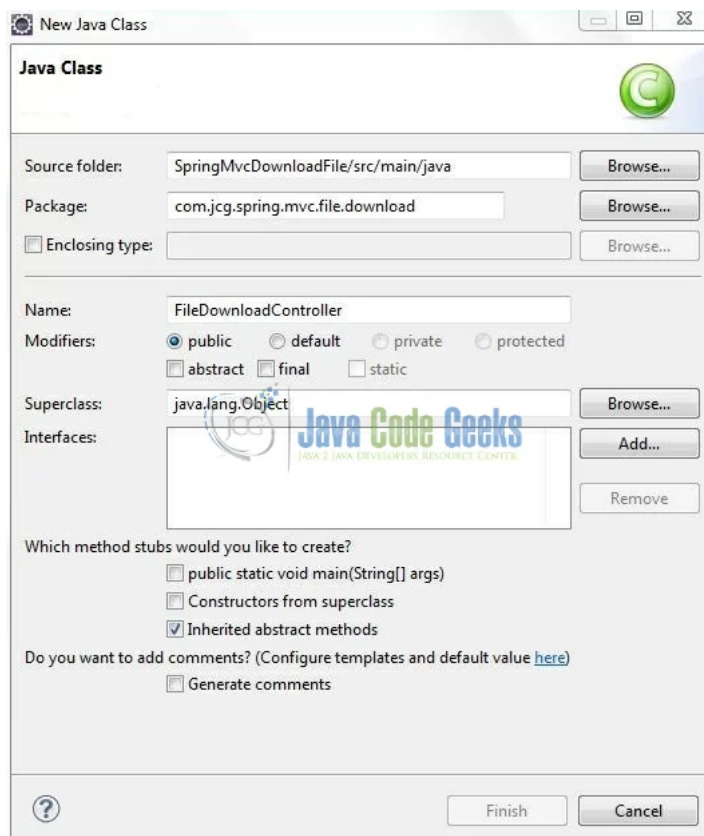
Fig. 11: Java Class Creation

A new pop window will open and enter the file name as:

FileDownloadController

. The spring controller class will be created inside the package:

com.jcg.spring.mvc.file.download



### 3.3.1 Implementation Of Controller Class

This is a typical spring controller class which is annotated by the Spring MVC annotation types. The methods

```
downloadPdf()
```

or the

```
downloadCsv()
```

will receive requests from the client. *These two methods will read the file on the server and send it back to the client for downloading.* Note that, unlike the traditional spring controller's methods, these methods do not return a view-name as the application's purpose is to send a file to the client. The method scope is completed as soon as the file is completely transferred to the client.

Let's write a quick Java program in the spring controller class to handle the file download requests. Add the following code to it.

*FileDownloadController.java*

```
01 package com.jcg.spring.mvc.file.download;
02
03 import java.io.File;
04 import java.io.IOException;
05
06 import javax.servlet.http.HttpServletRequest;
07 import javax.servlet.http.HttpServletResponse;
08
09 import org.apache.log4j.Logger;
10 import org.springframework.stereotype.Controller;
11 import org.springframework.ui.ModelMap;
12 import org.springframework.web.bind.annotation.RequestMapping;
13 import org.springframework.web.bind.annotation.RequestMethod;
14 import org.springframework.web.servlet.ModelAndView;
15
16 @Controller
17 public class FileDownloadController {
18
19     static ModelAndView modelAndViewObj;
20
21     private static Logger logger = Logger.getLogger(FileDownloadController.class);
22
23     @RequestMapping(value = {"/", "fileDownload"}, method = RequestMethod.GET)
24     public ModelAndView showUploadFileForm(ModelMap model) {
25         modelAndViewObj = new ModelAndView("fileDownload");
26         return modelAndViewObj;
27     }
28
29     @RequestMapping(value = "downloadFile/pdf", method = RequestMethod.GET)
30     public void downloadPdf(HttpServletRequest req, HttpServletResponse resp) throws IOException {
31         String pdfFilePath = "", pdfFileName = "irregular-verbs.pdf";
32         logger.info("Downloading A .PDF File From The Server ....!");
33
34         /**** Get The Absolute Path Of The File ****/
35         pdfFilePath = Util.getFilePath(req) + File.separator + pdfFileName;
36         logger.info("Absolute Path Of The .PDF File Is?= " + pdfFilePath);
37
38         File downloadFile = new File(pdfFilePath);
39         if(downloadFile.exists()) {
40             Util.downloadFileProperties(req, resp, pdfFilePath, downloadFile);
41         } else {
42             logger.info("Requested .PDF File Not Found At The Server ....!");
43         }
44     }
45
46     @RequestMapping(value = "downloadFile/csv", method = RequestMethod.GET)
47     public void downloadCsv(HttpServletRequest req, HttpServletResponse resp) throws IOException {
48         String csvFilePath = "";
49         logger.info("Downloading A .CSV File From The Server ....!");
50
51         /**** Get The Absolute Path Of The File ****/
52         csvFilePath = GenerateCsvData.writeDbDataToCsvFile(Util.getFilePath(req));
53         logger.info("Absolute Path Of The .CSV File Is?= " + csvFilePath);
54
55         File downloadFile = new File(csvFilePath);
56         if(downloadFile.exists()) {
57             Util.downloadFileProperties(req, resp, csvFilePath, downloadFile);
58         } else {
59             logger.info("Requested .CSV File Not Found At The Server ....!");
60         }
61     }
62 }
```

## 3.4 Configuration Files

Let's write all the configuration files involved in this application.

### 3.4.1 Spring Configuration File

To configure the spring framework, we need to implement a bean configuration file i.e.

```
spring-servlet.xml
```

which provide an interface between the basic Java class and the outside world. Put this

```
XML
```

file in the

```
SpringMvcDownloadFile/src/main/webapp/WEB-INF
```

```

01 <?xml version="1.0" encoding="UTF-8"?>
02 <beans xmlns="http://www.springframework.org/schema/beans" xmlns:context="http://www.springframework.org/sc
03
04 <context:component-scan base-package="com.jcg.spring.mvc.file.download" />
05
06 <!-- Resolves Views Selected For Rendering by @Controllers to *.jsp Resources in the /WEB-INF/ Folder --
07 <bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
08 <property name="prefix" value="/WEB-INF/views/" />
09 <property name="suffix" value=".jsp" />
10 </bean>
11
12 <!-- File Download Exception Resolver i.e. In Case Of Exception The Controller Will Navigate To 'error.j
13 <bean class="org.springframework.web.servlet.handler.SimpleMappingExceptionResolver">
14 <property name="exceptionMappings">
15 <props>
16 <prop key="java.lang.Exception">error</prop>
17 </props>
18 </property>
19 </bean>
20 </beans>

```

### 3.4.2 Web Deployment Descriptor

The

web.xml

file declares one servlet (i.e. Dispatcher Servlet) to receive all kind of the requests and specifies the default page when accessing the application. Dispatcher servlet here acts as a front controller. Add the following code to it:

web.xml

```

01 <?xml version="1.0" encoding="UTF-8"?>
02 <web-app xmlns="http://java.sun.com/xml/ns/javaee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" ve
03 <display-name>Spring Mvc File Download Example</display-name>
04 <!-- Spring Configuration - Processes Application Requests -->
05 <servlet>
06 <servlet-name>SpringController</servlet-name>
07 <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
08 <init-param>
09 <param-name>contextConfigLocation</param-name>
10 <param-value>/WEB-INF/spring-servlet.xml</param-value>
11 </init-param>
12 <load-on-startup>1</load-on-startup>
13 </servlet>
14 <servlet-mapping>
15 <servlet-name>SpringController</servlet-name>
16 <url-pattern>/</url-pattern>
17 </servlet-mapping>
18 </web-app>

```

## 3.5 Creating JSP Views

Spring Mvc supports many types of views for different presentation technologies. These include –

JSP

HTML

XML

etc. So let us write a simple view in

SpringMvcDownloadFile /src/main/webapp/WEB-INF/views

folder. This page simply shows the download file links which are handled by the methods in the spring controller class (i.e.

FileDownloadController.java

). Add the following code to it:

fileDownload.jsp

```

01 <%@ page language="java" contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO-8859-1"%>
02 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
03 <html>
04 <head>
05 <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
06 <title>Spring Mvc File Download Example</title>
07 <style type="text/css">
08 .linkCSS {
09 cursor: pointer;
10 text-decoration: none;
11 }
12 .padding {
13 padding: 13px 0px 20px 145px;
14 }
15 </style>
16 </head>
17 <body>
18 <center><h2>Spring Mvc File Download Example</h2></center>

```

```

23         <a id="downloadCsvFileLink" target="_self" class="linkCSS" href="${pageContext.request.contextP
24     </div>
25 </body>
26 </html>

```

## 4. R

As we ar

Tomat7, right-click on the project and navigate to

Run as -> Run on Server

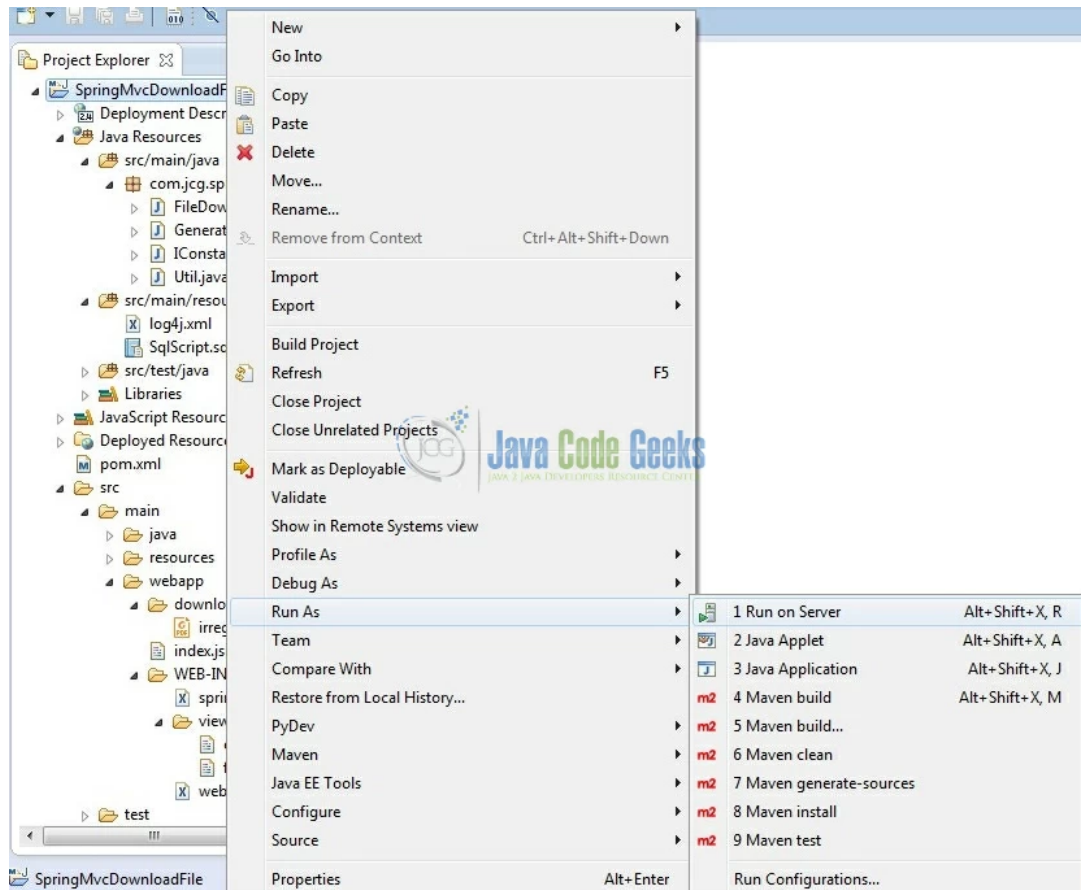


Fig. 13: How to Deploy Application on Tomcat

Tomcat will deploy the application in its web-apps folder and shall start its execution to deploy the project so that we can go ahead and test it in the browser.

## 5. Project Demo

Open your favorite browser and hit the following URL. The output page will be displayed.

<http://localhost:8085/SpringMvcDownloadFile/>

Server name (localhost) and port (8085) may vary as per your tomcat configuration. Developers can debug the example and see what happens after every step. Enjoy!



Fig. 14: File Download Page

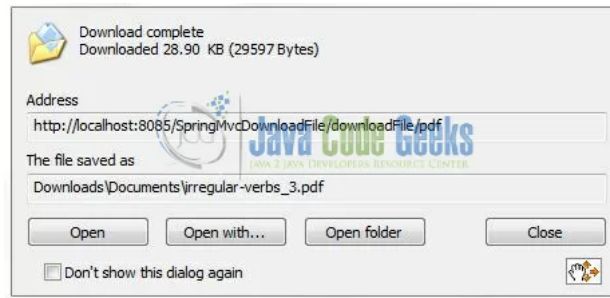


Fig. 15: Pdf File Download

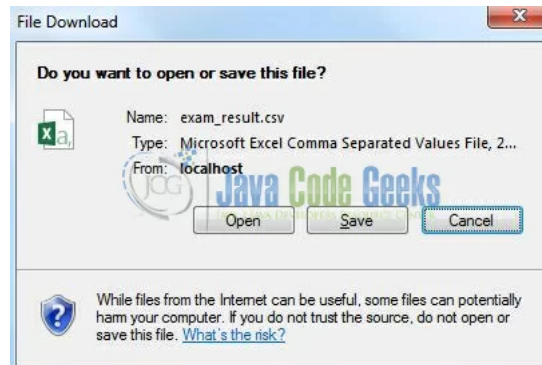


Fig. 16: CSV File Download

That's all for this post. Happy Learning!!

## 6. Conclusion

In this section, developers learned how to create a sample Spring Mvc application that allows the file download functionality. Developers can download the sample application as an Eclipse project in the Downloads section, and remember to *update* the database connection settings.

## 7. Download the Eclipse Project

This was an example of *File Download* with Spring Mvc.

### Download

You can download the full source code of this example here: **SpringMvcDownloadFile**

Tagged with: CORE JAVA CSV JAVA JAVA 8 JAVA CODE JDBC SPRING SPRING MVC

## Do you want to know how to develop your skillset to become a **Java Rockstar**?

Subscribe to our newsletter to start Rocking right now!

To get you started we give you our best selling eBooks for **FREE!**

1. JPA Mini Book
2. JVM Troubleshooting Guide
3. JUnit Tutorial for Unit Testing
4. Java Annotations Tutorial
5. Java Interview Questions
6. Spring Interview Questions
7. Android UI Design

and many more ....

### Email address:

☒ Receive Java & Developer job alerts in your Area

Sign up

Leave a Reply

## 2 Comments on "Spring MVC File Download Example"

Notify of



Join the discussion

Sort by: newest|oldest|most voted



Tahta Kafa



It could be better if you put

util

Guest

class that has downloading functionality, it was the core of the article. Thank for your article.

0

🕒 9 days 18 hours ago ^



Author

Yatin Batra



Tahta, hope you are doing good. The relevant files are already present in the code. Please download the project and let us know in case any further information is required from JCG end.

0

🕒 9 days 10 hours ago

### KNOWLEDGE BASE

Courses

Minibooks

News

Resources

Tutorials

### THE CODE GEEKS NETWORK

.NET Code Geeks

Java Code Geeks

System Code Geeks

Web Code Geeks

### HALL OF FAME

Android Alert Dialog Example

Android OnClickListener Example

How to convert Character to String and a String to Character Array in Java

Java Inheritance example

Java write to File Example

java.io.FileNotFoundException – How to solve File Not Found Exception

java.lang.ArrayIndexOutOfBoundsException – How to handle Array Index Out Of Bounds Exception

java.lang.NoClassDefFoundError – How to solve No Class Def Found Error

JSON Example With Jersey + Jackson

Spring JdbcTemplate Example

### ABOUT JAVA CODE GEEKS

JCGs (Java Code Geeks) is an independent online community focused on ultimate Java to Java developers resource center; targeted at the technical team lead (senior developer), project manager and junior dev. JCGs serve the Java, SOA, Agile and Telecom communities with daily in domain experts, articles, tutorials, reviews, announcements, code snippets, source projects.

### DISCLAIMER

All trademarks and registered trademarks appearing on Java Code Geeks are the property of their respective owners. Java is a trademark or registered trademark of Oracle Corporation in the United States and other countries. Examples are not connected to Oracle Corporation and is not sponsored by Oracle.

