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
```

.

Table Of Contents

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 **M**odel, **V**iew, and **C**ontroller 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

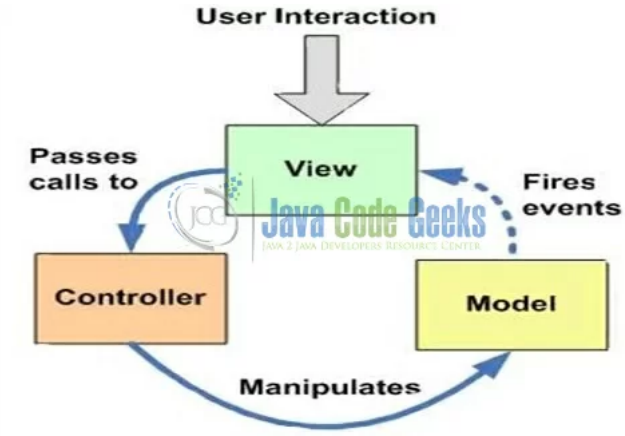


Fig. 1: Model View Controller (MVC) Overview

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:

```
HttpServletRespone
```

- : 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.

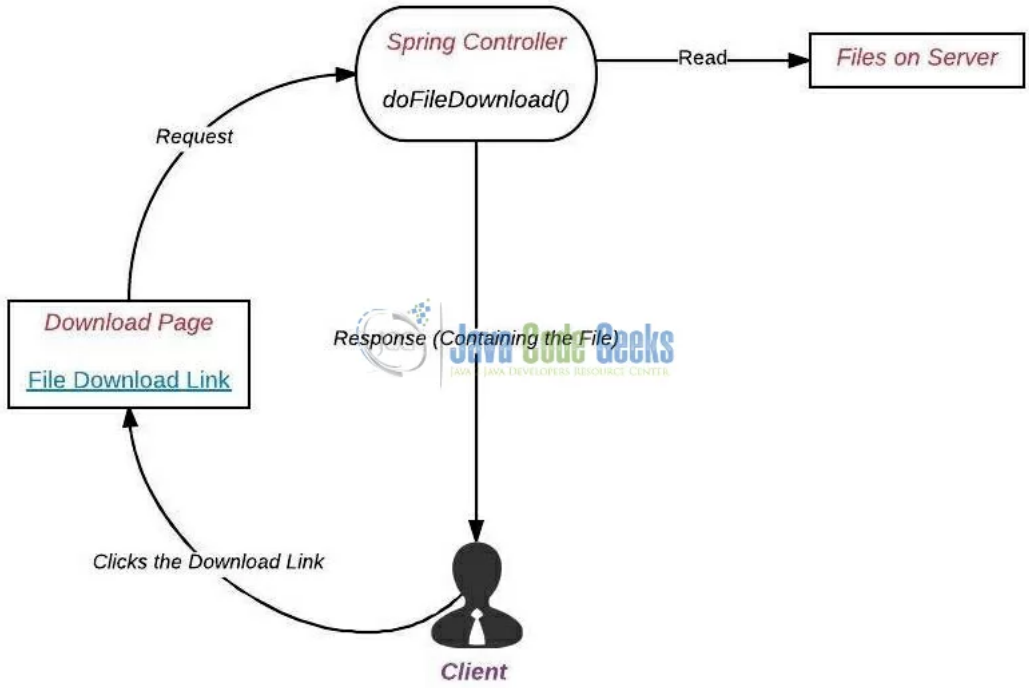


Fig. 2: Spring Mvc File Download Application Workflow

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!

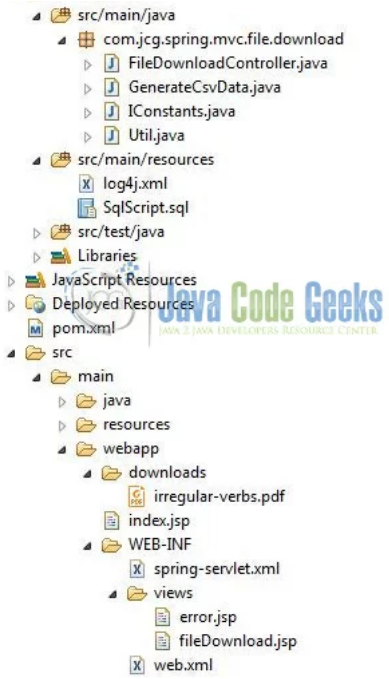


Fig. 3: Application Project Structure

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
```

.

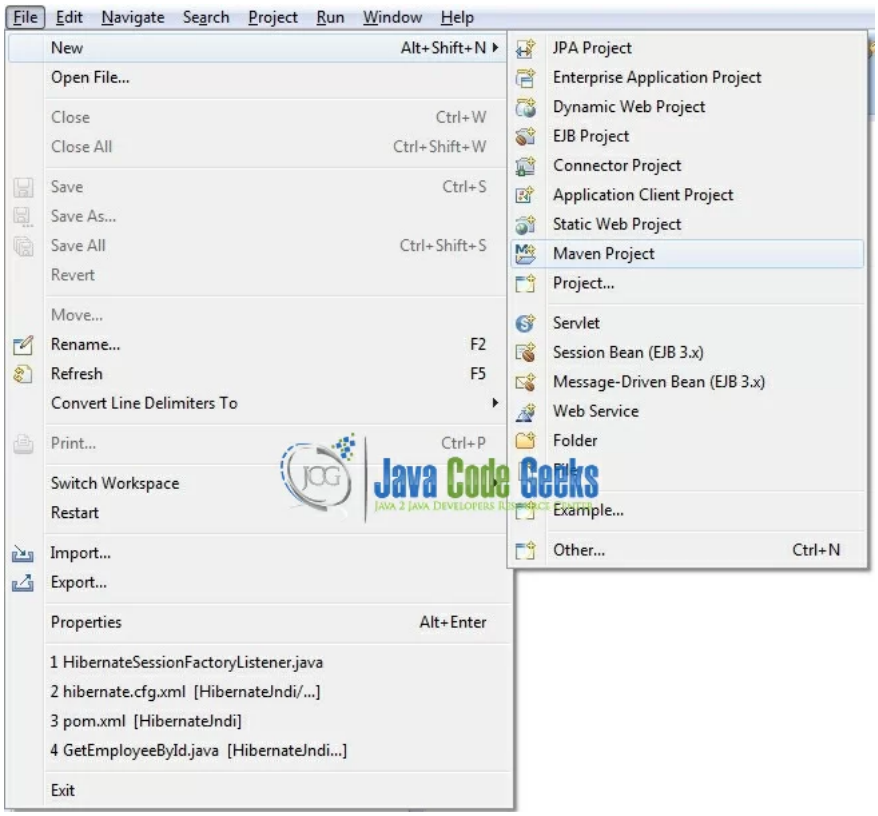


Fig. 4: Create Maven Project

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.

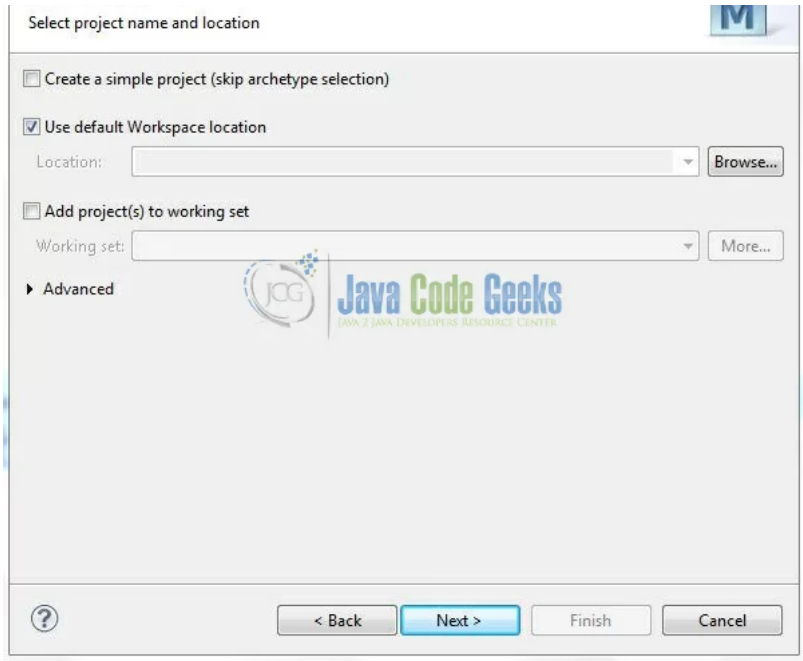


Fig. 5: Project Details

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

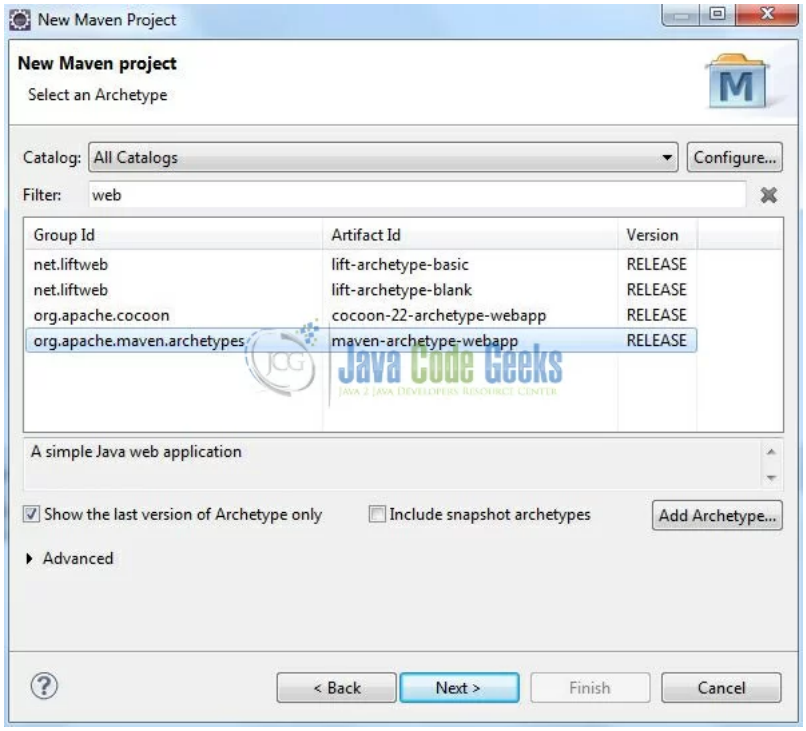


Fig. 6: Archetype Selection

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
```

.

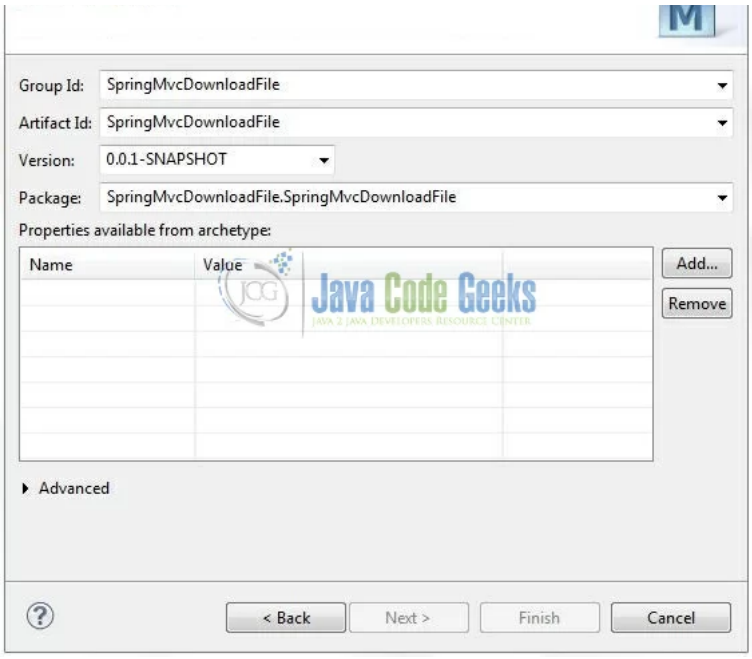


Fig. 7: Archetype Parameters

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

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

```
CREATE DATABASE IF NOT EXISTS filedownload;

USE filedownload;

CREATE TABLE exam_result (
    student_id INTEGER NOT NULL,
    student_name VARCHAR(30) NOT NULL,
    student_dob DATE NOT NULL,
    student_percentage  double NOT NULL
);

INSERT INTO exam_result (student_id, student_name, student_dob, student_percentage) VALUES (101, 'Harry Pot
INSERT INTO exam_result (student_id, student_name, student_dob, student_percentage) VALUES (102, 'Java Code
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.

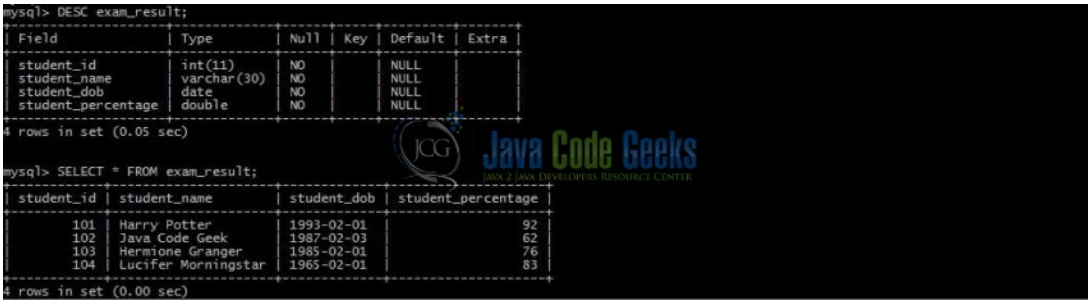


Fig. 8: Database & Table Creation

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" xs
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
```

.

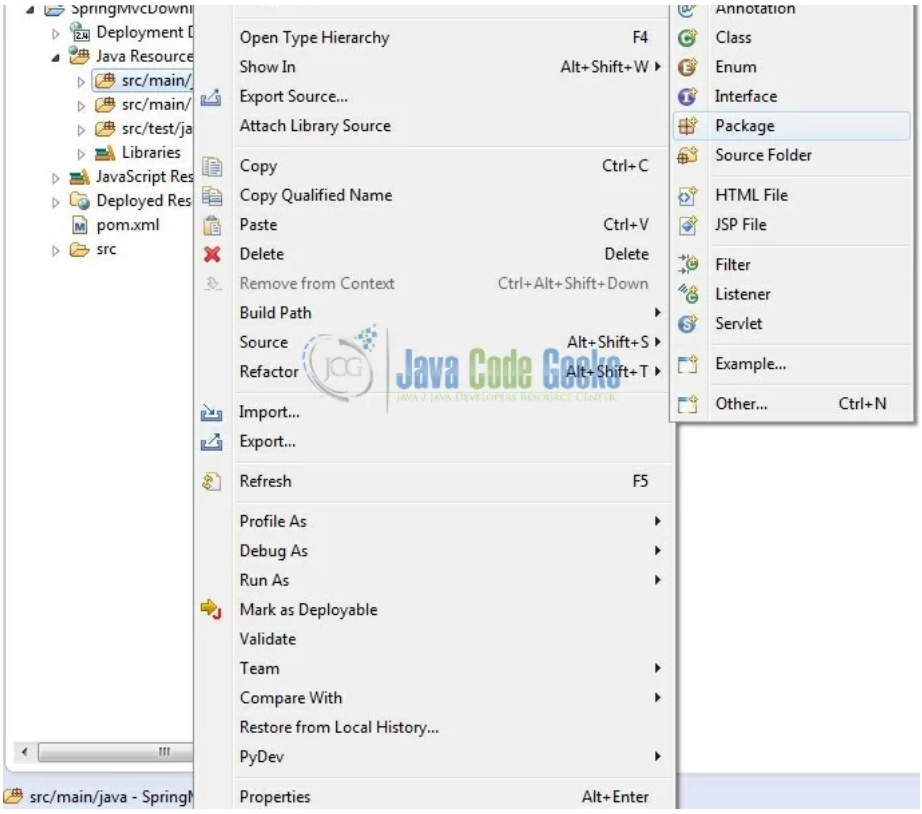


Fig. 9: Java Package Creation

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

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

.

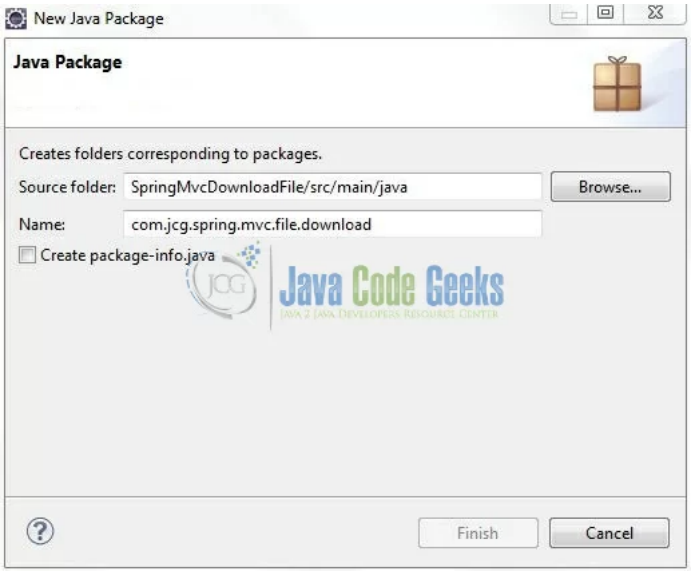


Fig. 10: Java Package Name (com.jcg.spring.mvc.file.download)

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

```
New -> Class
```

.

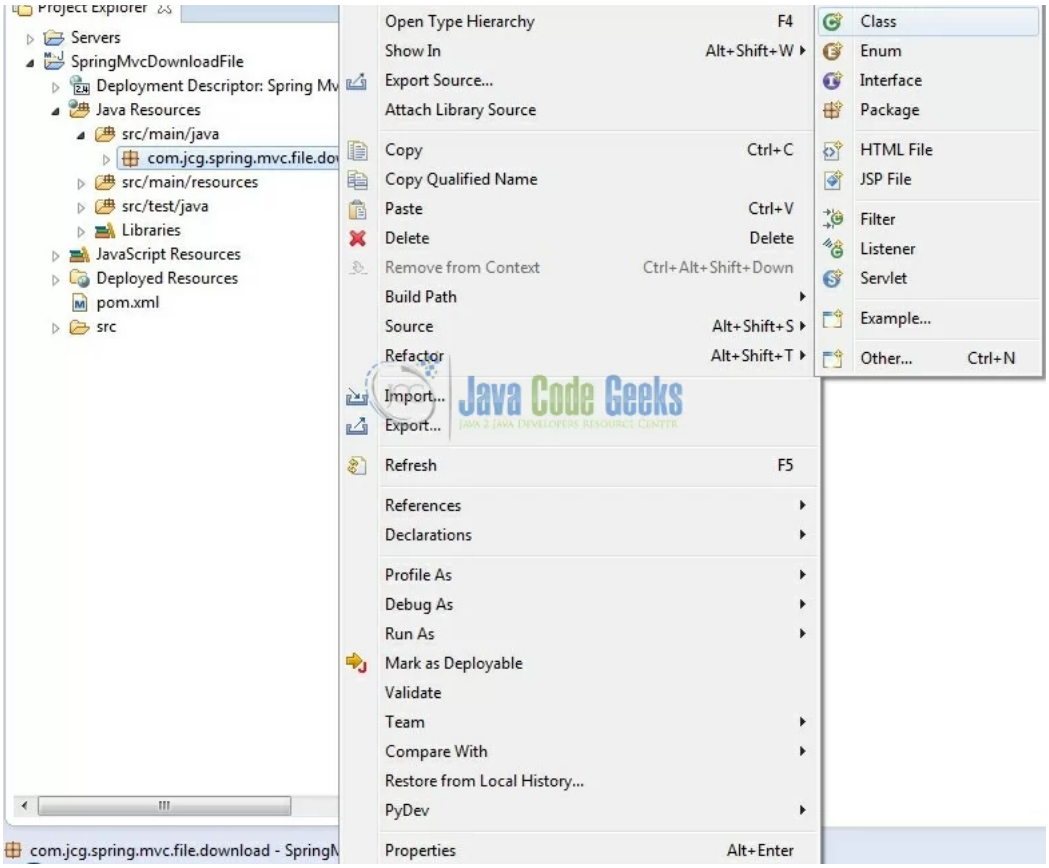


Fig. 11: Java Class Creation

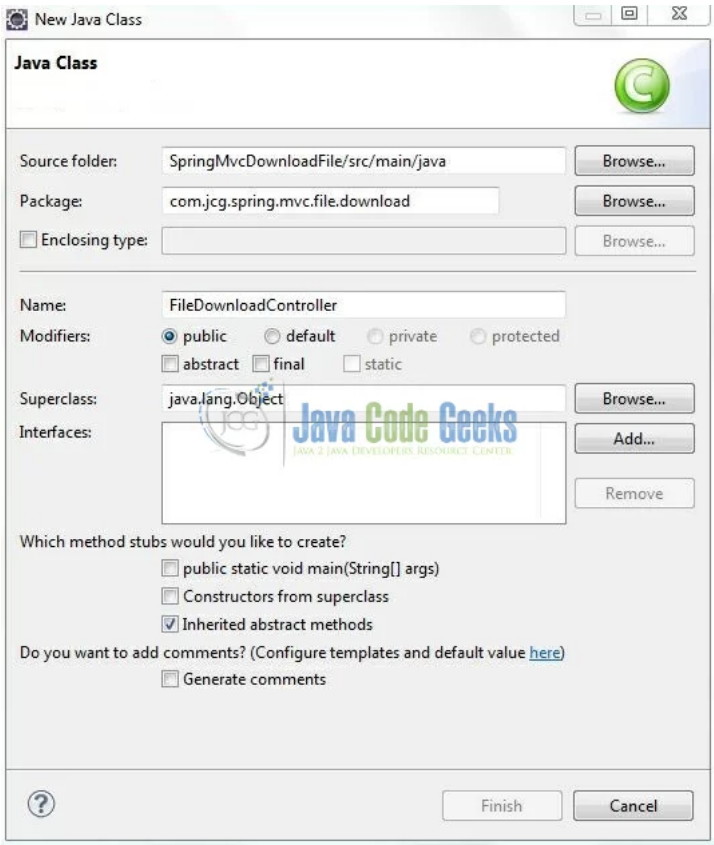
A new pop window will open and enter the file name as:

```
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 modelViewObj;
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         modelViewObj = new ModelAndView("fileDownload");
26         return  modelViewObj;
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
```

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans" xmlns:context="http://www.springframework.org/sc

   <context:component-scan base-package="com.jcg.spring.mvc.file.download" />

   <!-- Resolves Views Selected For Rendering by @Controllers to *.jsp Resources in the /WEB-INF/ Folder --
   <bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
      <property name="prefix" value="/WEB-INF/views/" />
      <property name="suffix" value=".jsp" />
   </bean>

   <!-- File Download Exception Resolver i.e. In Case Of Exception The Controller Will Navigate To 'error.j
   <bean class="org.springframework.web.servlet.handler.SimpleMappingExceptionResolver">
      <property name="exceptionMappings">
         <props>
            <prop key="java.lang.Exception">error</prop>
         </props>
      </property>
   </bean>
</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

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://java.sun.com/xml/ns/javaee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" ve
   <display-name>Spring Mvc File Download Example</display-name>
   <!-- Spring Configuration - Processes Application Requests -->
   <servlet>
      <servlet-name>SpringController</servlet-name>
      <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
      <init-param>
         <param-name>contextConfigLocation</param-name>
         <param-value>/WEB-INF/spring-servlet.xml</param-value>
      </init-param>
      <load-on-startup>1</load-on-startup>
   </servlet>
   <servlet-mapping>
      <servlet-name>SpringController</servlet-name>
      <url-pattern>/</url-pattern>
   </servlet-mapping>
</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

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
        <title>Spring Mvc File Download Example</title>
        <style type="text/css">
            .linkCSS {
                cursor: pointer;
                text-decoration: none;
            }
            .padding {
                padding: 13px 0px 20px 145px;
            }
        </style>
    </head>
    <body>
        <center><h2>Spring Mvc File Download Example</h2></center>
```

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

4. R

As we ar

Tomat7, right-click on the project and navigate to

```
Run as -> Run on Server
```

.

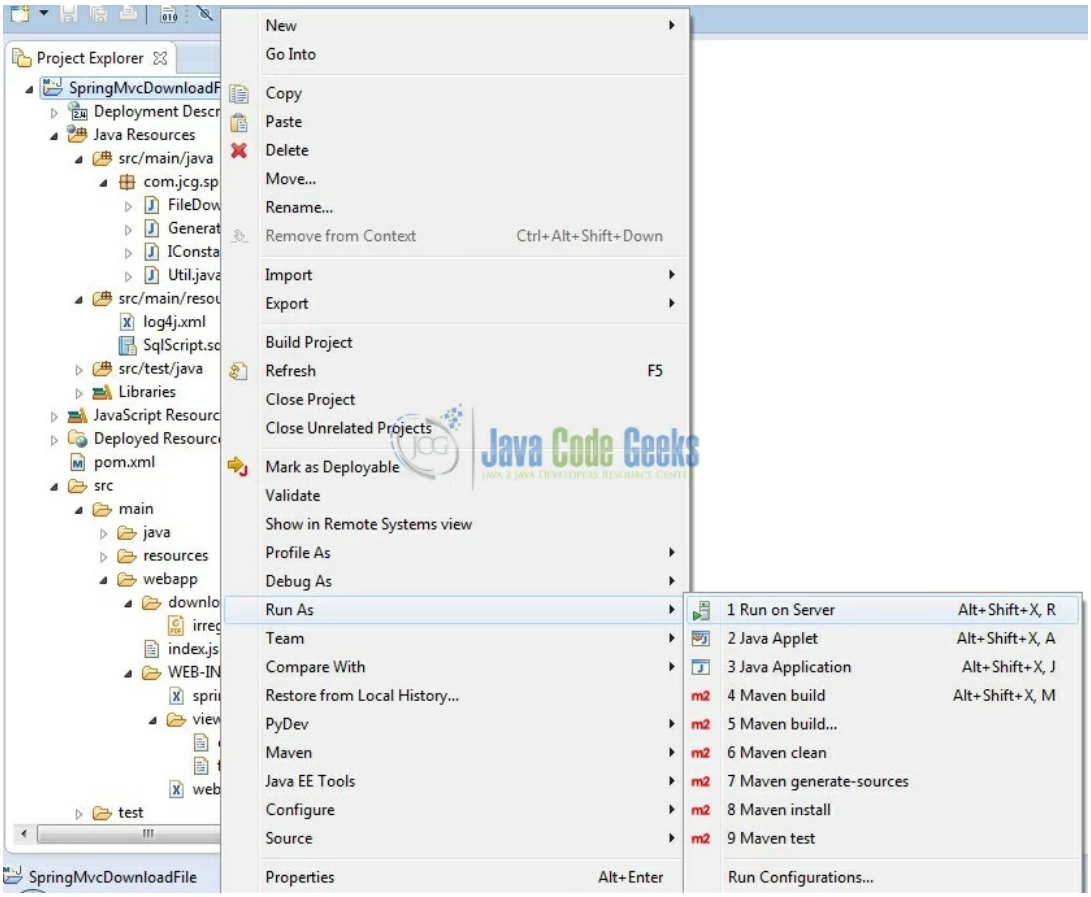


Fig. 13: How to Deploy Application on Tomcat

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

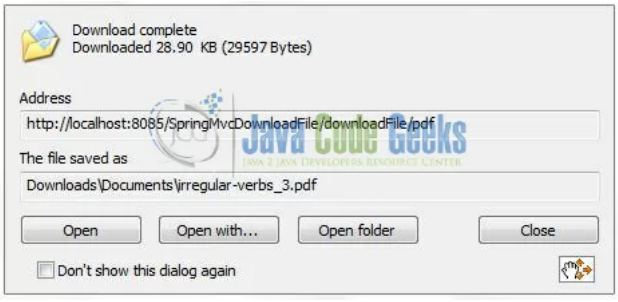


Fig. 15: Pdf File Download

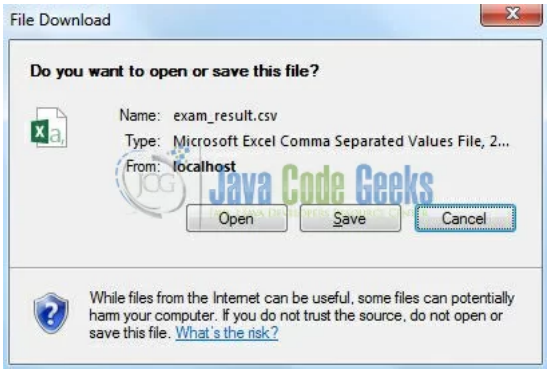


Fig. 16: CSV File Download

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

Leave a Reply

2 Comments on "Spring MVC File Download Example"

Notify of new follow-up comments Email ›

Join the discussion

Sort by: newest|oldest|most voted

Tahta Kafa

Guest

It could be better if you put

```
Util
```

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

0 REPLY 9 days 18 hours ago

Yatin Batra

Author

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 REPLY 9 days 10 hours ago