Contents

[SpringMVC 3](#_Toc22208052)

[SpringMVC Workflow 3](#_Toc22208053)

[Create Maven Project 4](#_Toc22208054)

[Add the Dependency for SpringMVC 8](#_Toc22208055)

[Basic Steps for SpringMVc 9](#_Toc22208056)

[Web.xml Configuration 9](#_Toc22208057)

[Create Servlet Config File 10](#_Toc22208058)

[Create Controller 14](#_Toc22208059)

[Create View 15](#_Toc22208060)

[View Resolver 16](#_Toc22208061)

[Run the Project 17](#_Toc22208062)

[2. Spring Forms Tag 18](#_Toc22208063)

[2.1 Create SpringMVC Forms 19](#_Toc22208064)

[2.1.1 Create Basic Html tag 20](#_Toc22208065)

[2.1.2 Run the Basic Html Page in server 20](#_Toc22208066)

[2.1.3 Converting Basic Html to SpringMvc Form 20](#_Toc22208067)

[2.2 Validating SpringMVC Form 25](#_Toc22208068)

[2.2.1 Add the dependency 25](#_Toc22208069)

[2.2.2 Add the validation in domain model 25](#_Toc22208070)

[2.2.3 Enable validation attribute in Controller class 26](#_Toc22208071)

[2.2.4 Binding error result in Controller class 27](#_Toc22208072)

[2.2.5 Display error in JSP page 28](#_Toc22208073)

[2.2.6 Run the Application 29](#_Toc22208074)

[2.3 Narrowing Request Mapping Using Http Request Method 30](#_Toc22208075)

[2.3.1 Add the Request Method in Request mapping in Controller 30](#_Toc22208076)

[2.3.2 Add the Request Method in Request mapping in UI 30](#_Toc22208077)

[2.3.3 Run the Application 31](#_Toc22208078)

[2.4 File Upload using Commons File Upload 32](#_Toc22208079)

[2.4.1 Add the Dependency 32](#_Toc22208080)

[2.4.2 Add the Multipart configuration in servletconfig xml file 32](#_Toc22208081)

[2.4.3 Add the file upload tag in JSP page 34](#_Toc22208082)

[2.4.4 Read the File from UI to Controller 34](#_Toc22208083)

[2.4.5 Run the Application 35](#_Toc22208084)

[3. Some more Feature of SpringMVC 37](#_Toc22208085)

[3.1 Serving Static Files from SpringMVC 38](#_Toc22208086)

[3.2 Redirecting in SpringMvC 40](#_Toc22208087)

[3.2.1 Redirect from UI Click on Submit button 40](#_Toc22208088)

[3.2.2 Redirect directly from Controller 42](#_Toc22208089)

[3.3 Enhance the Spring Model 44](#_Toc22208090)

[3.3.1 @ModelAttribute 46](#_Toc22208091)

[3.3.2 @SessionAttribute 48](#_Toc22208092)

[3.3.3 Working with cookies 53](#_Toc22208093)

[4. SpringMVC App with Java Based Configuration 55](#_Toc22208094)

[4.1 Create the Configuration Class 56](#_Toc22208095)

[4.2 Enable @WebMVC Annotation and extends WebMvcConfigureAdaptor 56](#_Toc22208096)

[4.3 Enable ComponentScan 57](#_Toc22208097)

[4.4 Enable Bean in Java Configuration 58](#_Toc22208098)

[4.5 Enable Resources 59](#_Toc22208099)

[4.6 Add the Context Class in web.xml 61](#_Toc22208100)

[4.7 Run the Application 62](#_Toc22208101)

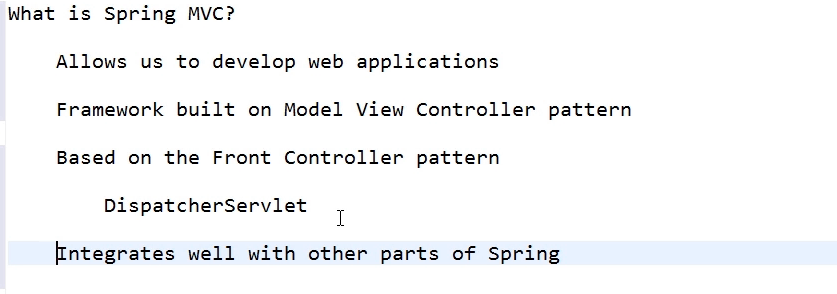
[5. Spring MVC Controller Handler Methods 63](#_Toc22208102)

[5.1 Matching Request Based on URL Parameters 63](#_Toc22208103)

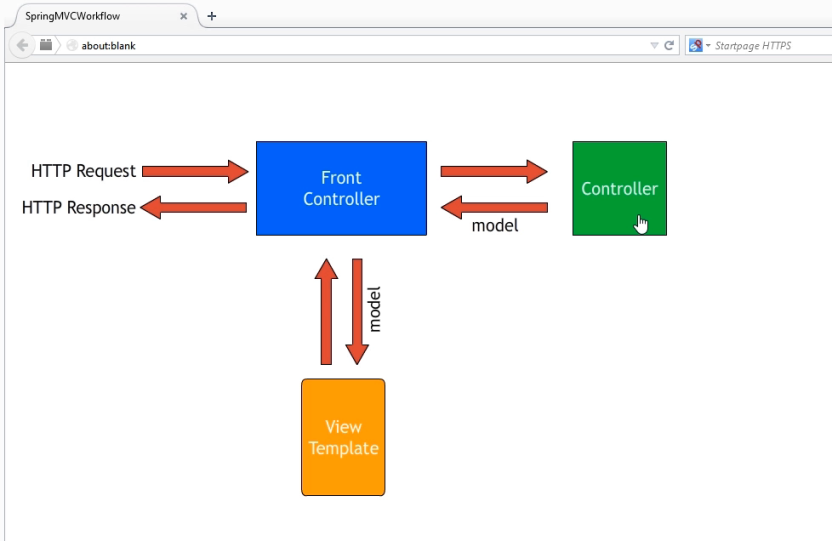
[5.2 Matching Request Based on Headers 67](#_Toc22208104)

[5.3 Accessing Request Parameters in Handler Methods 70](#_Toc22208105)

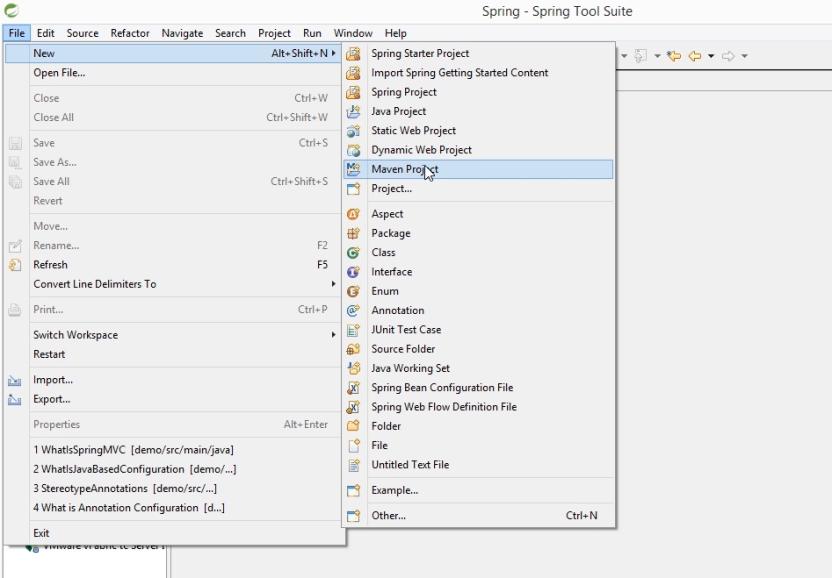
# SpringMVC

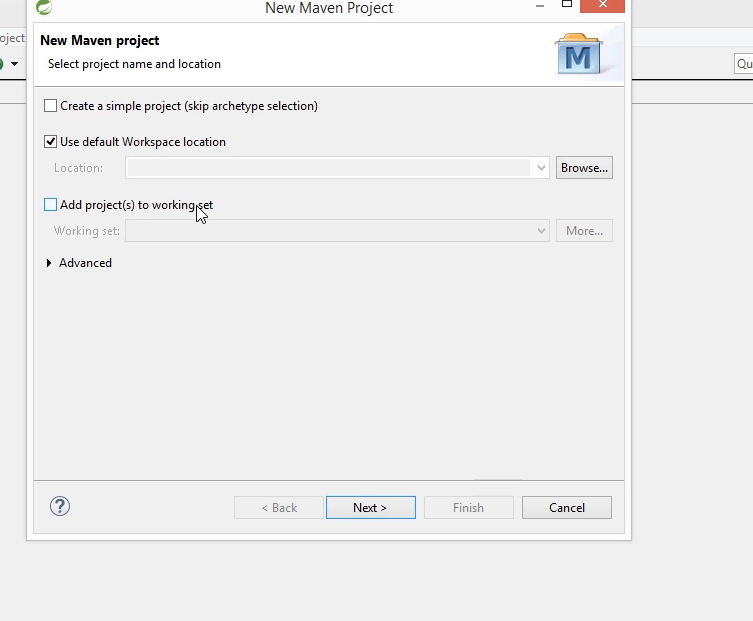


# SpringMVC Workflow

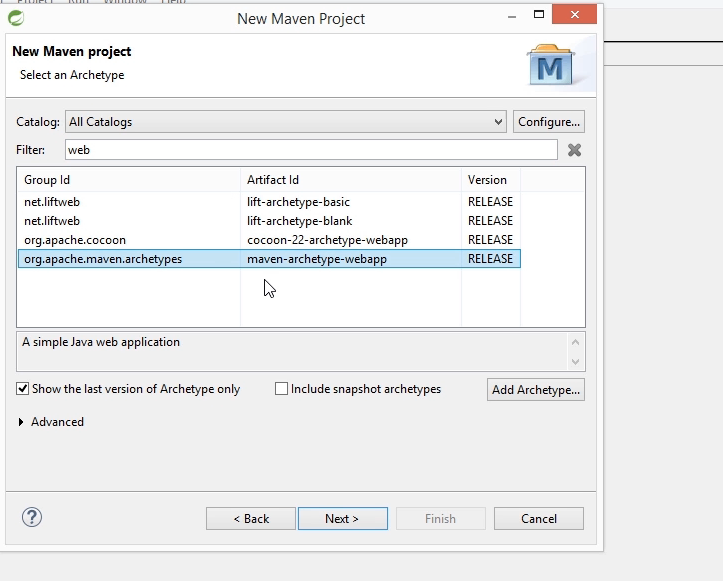


# Create Maven Project



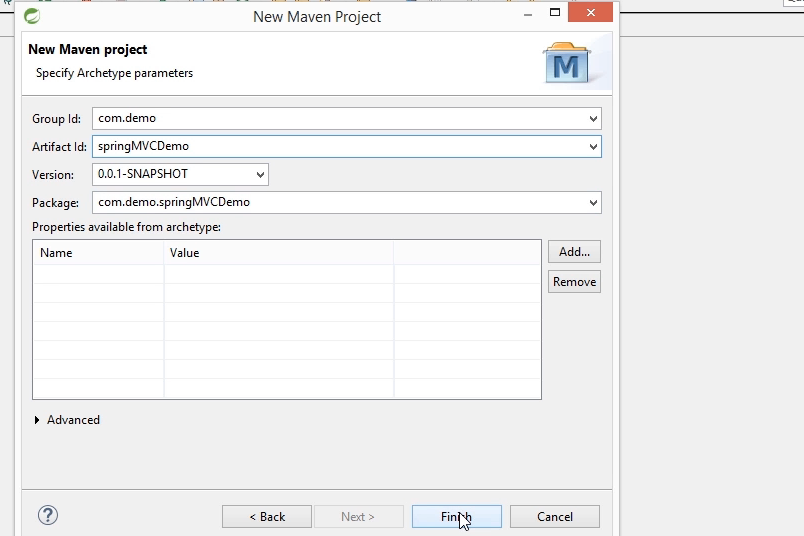


Click Next



Choose web archetypr

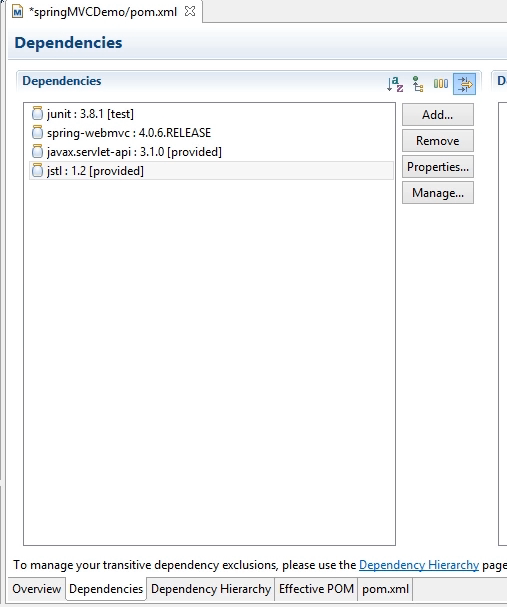
Then click Next



Enter Group Id, Artefact Id

Then click Finish

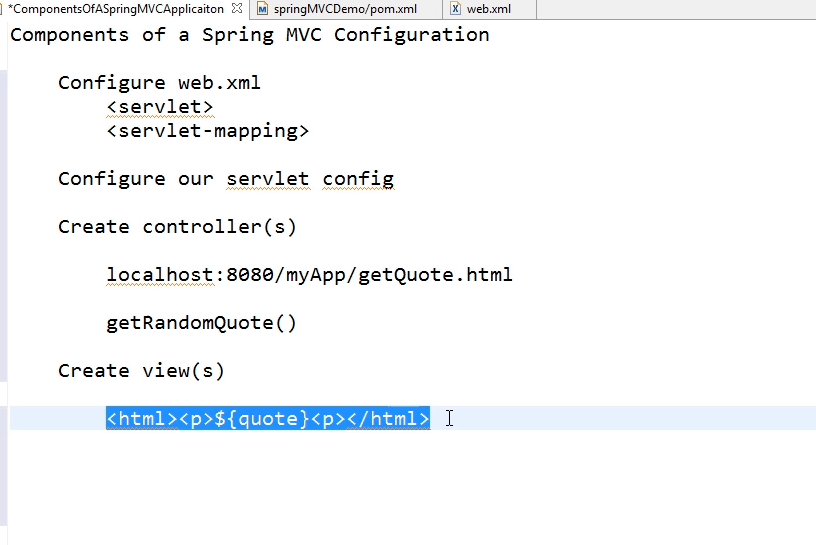
# Add the Dependency for SpringMVC



Open pom.xml

Add the above mentioned dependency

# Basic Steps for SpringMVc

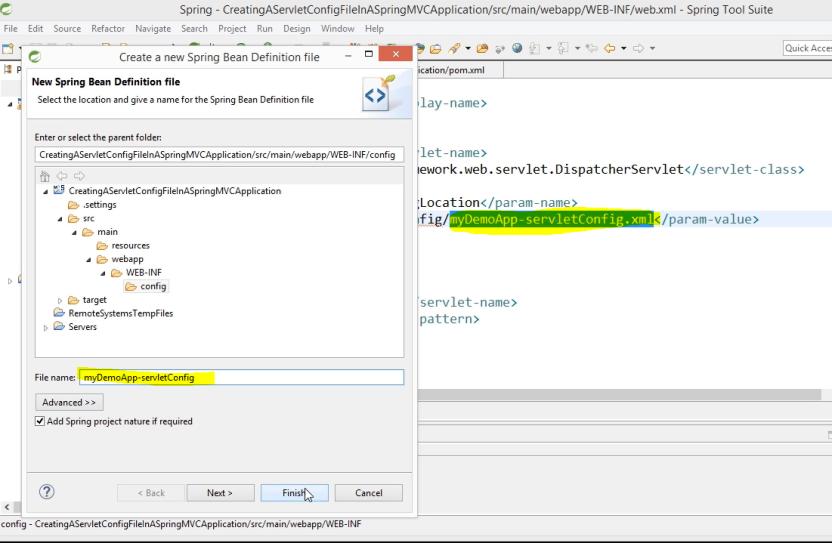


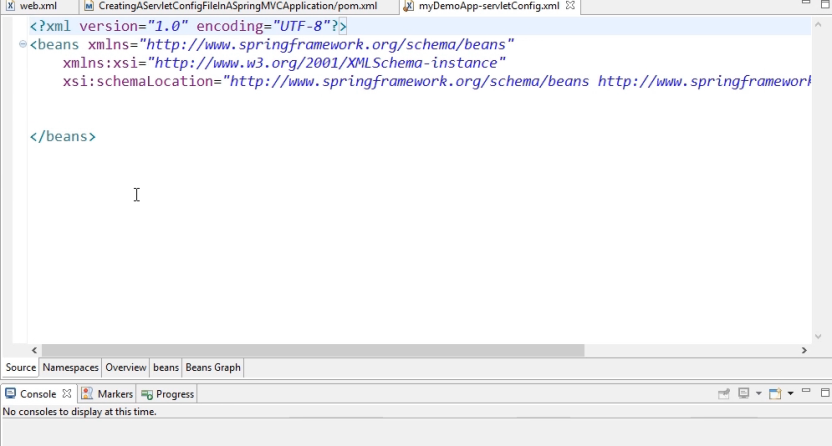
# Web.xml Configuration

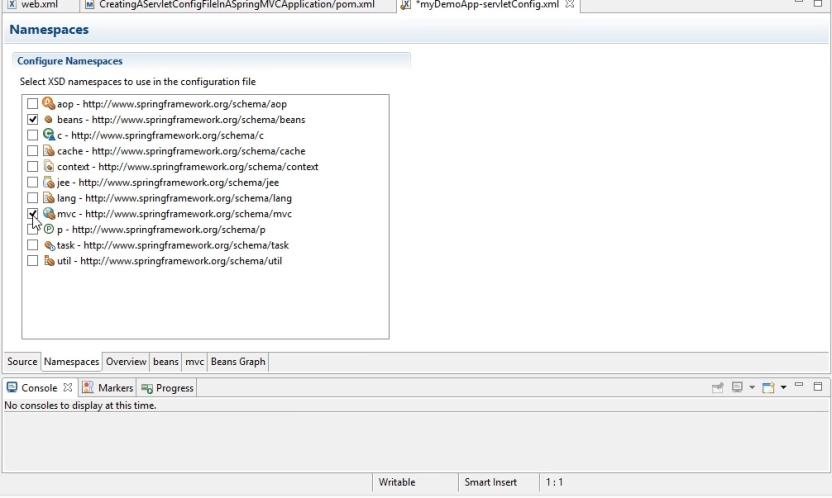


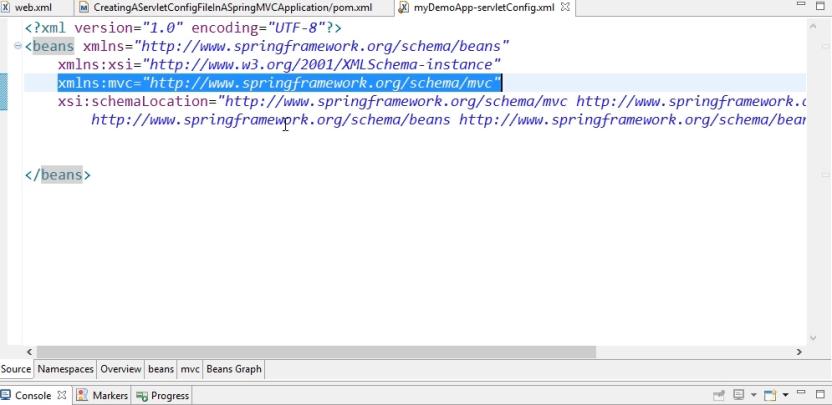
# Create Servlet Config File

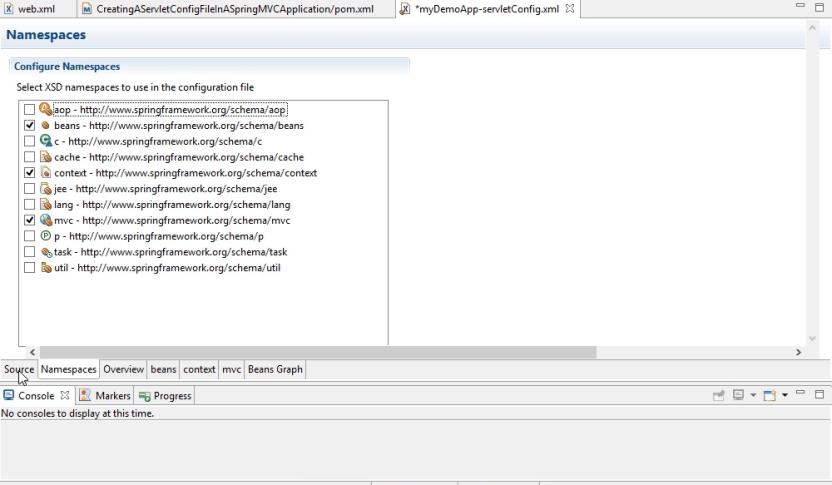


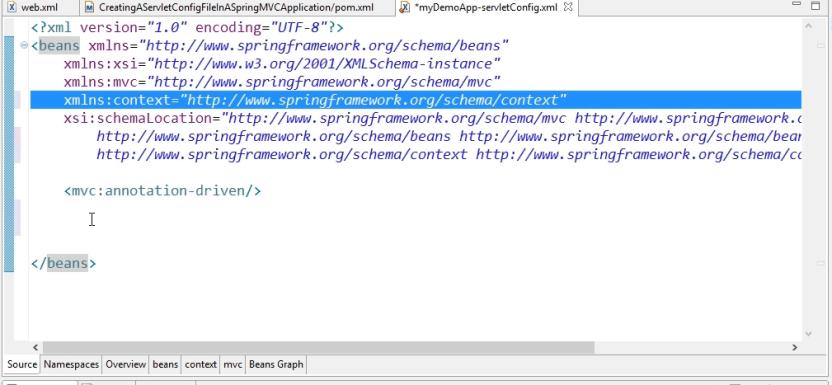


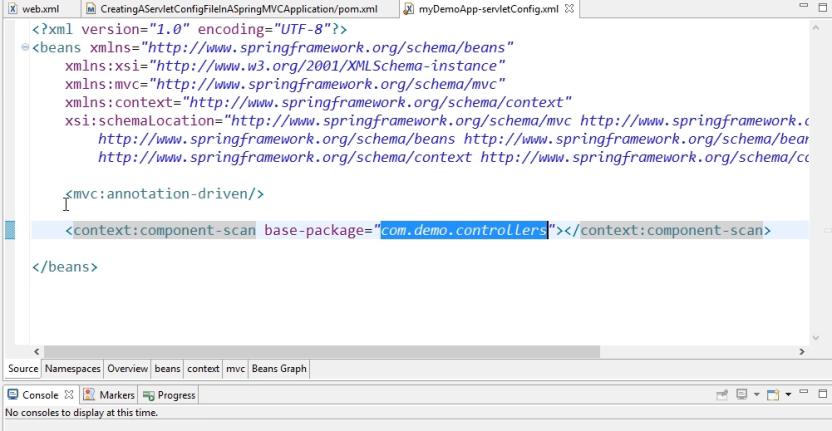




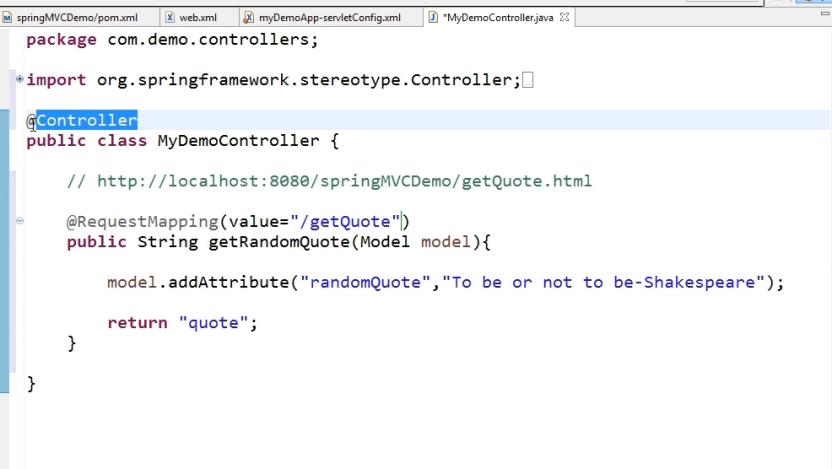




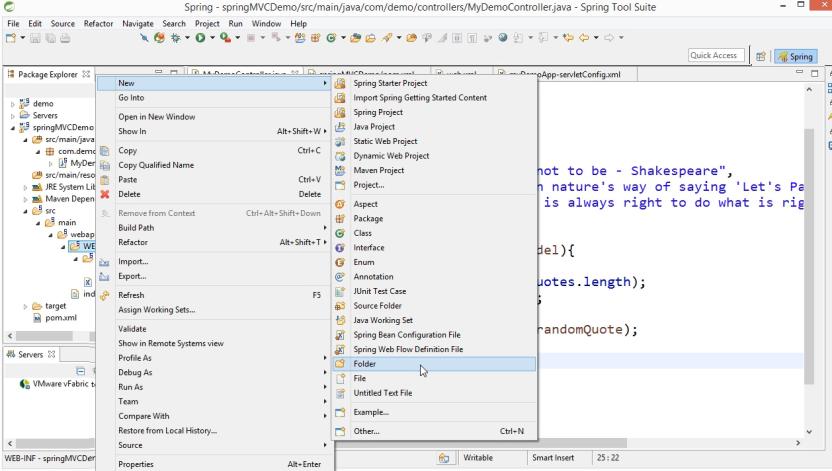


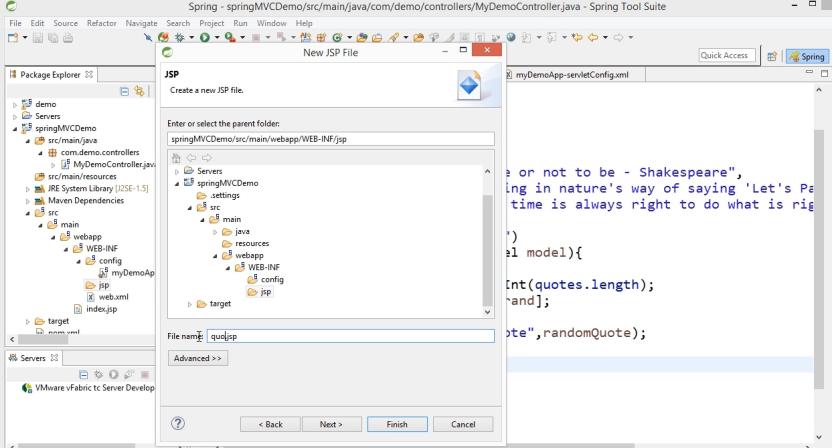


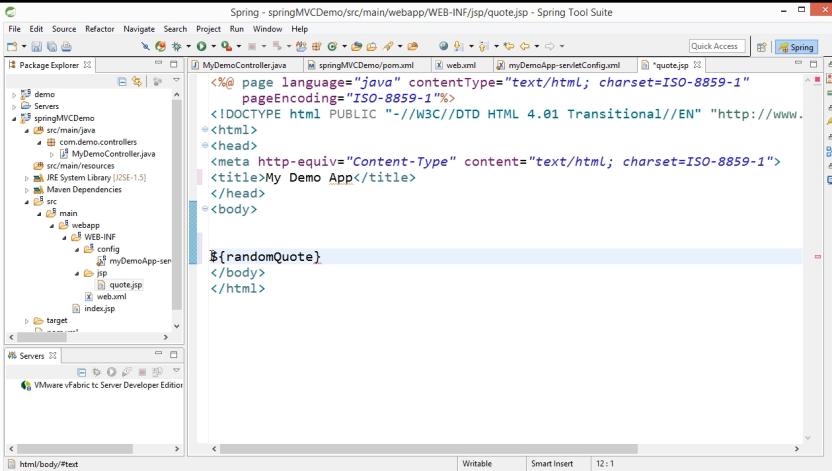
# Create Controller



# Create View



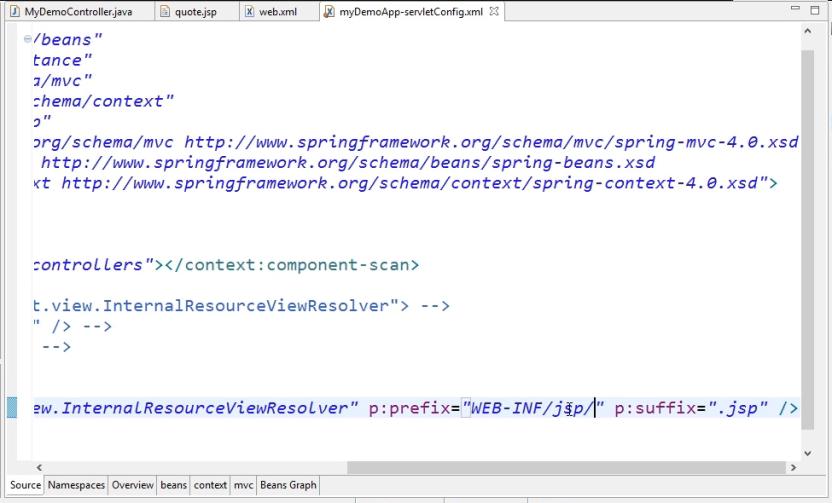




# View Resolver



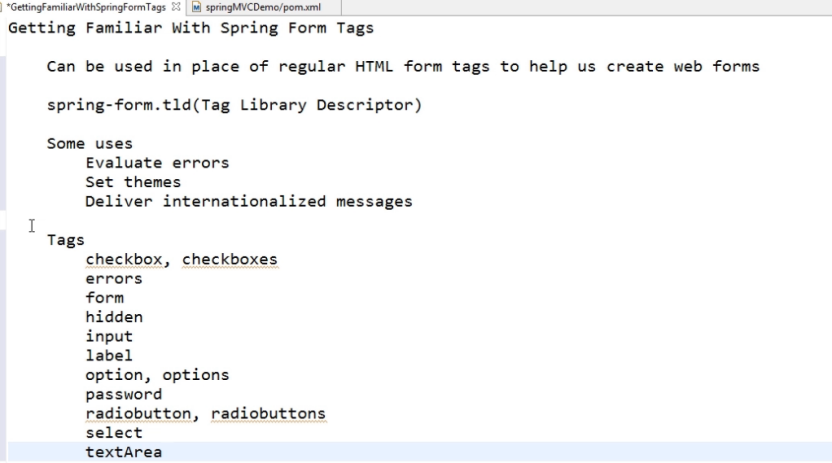
Or

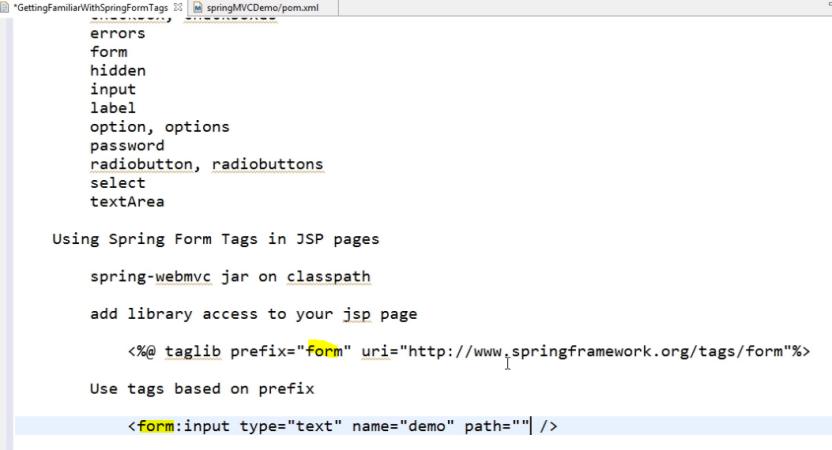


# Run the Project



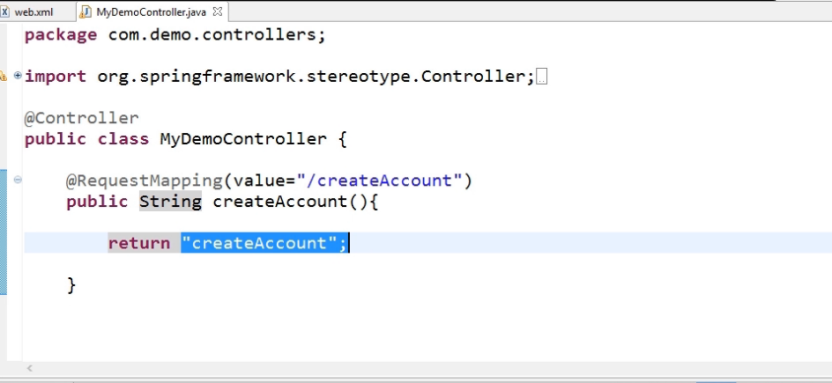
# 2. Spring Forms Tag



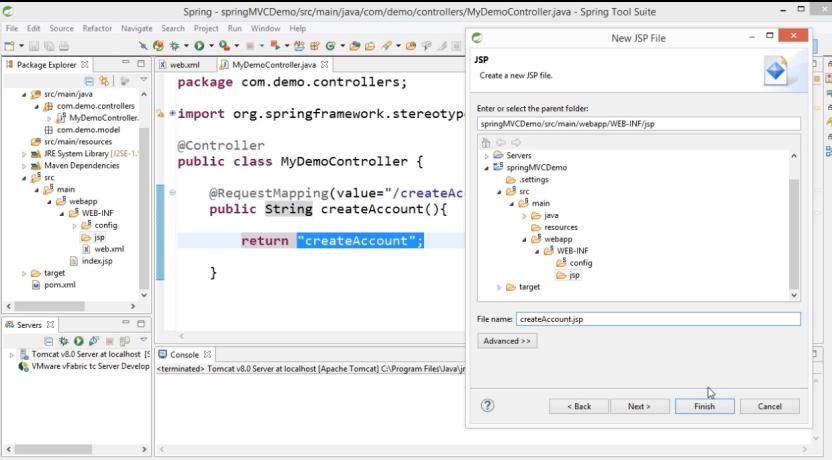


## 2.1 Create SpringMVC Forms

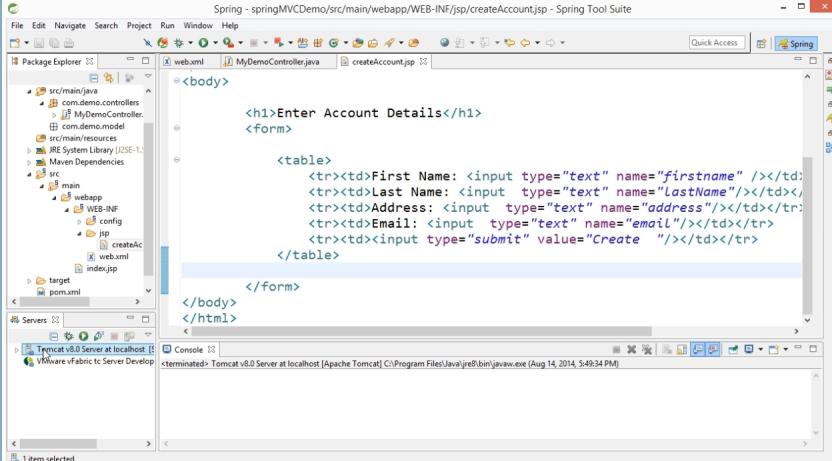
Create Controller



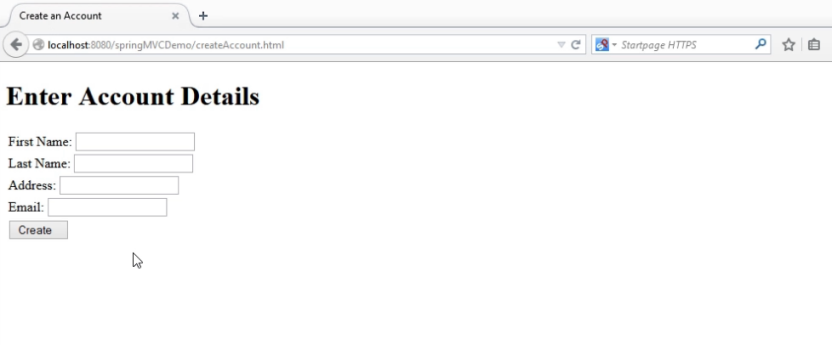
Create View:



### 2.1.1 Create Basic Html tag

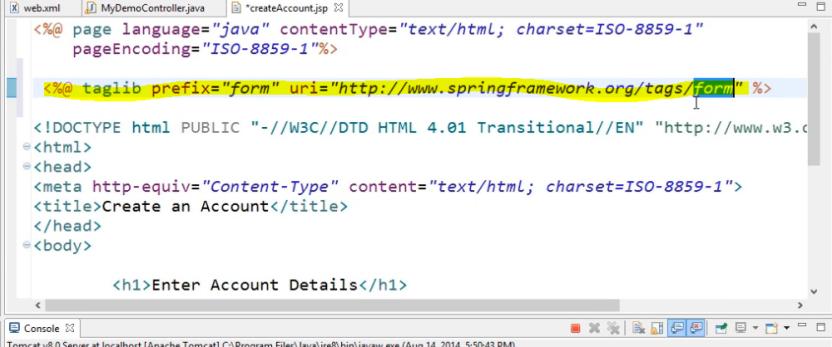


### 2.1.2 Run the Basic Html Page in server

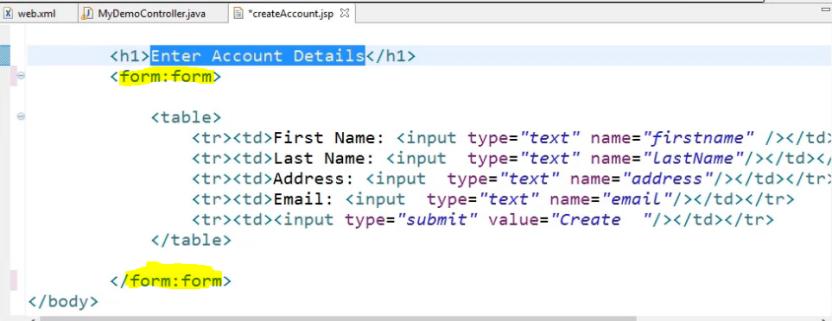


### 2.1.3 Converting Basic Html to SpringMvc Form

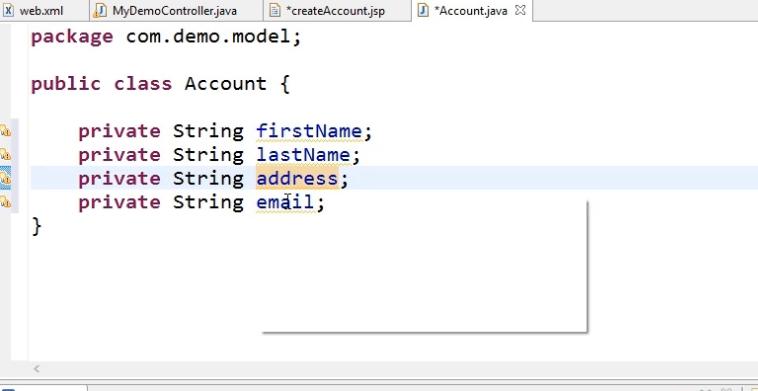
Declare taglib in the JSP page



Spring Mvc form tag is used for data binding feature

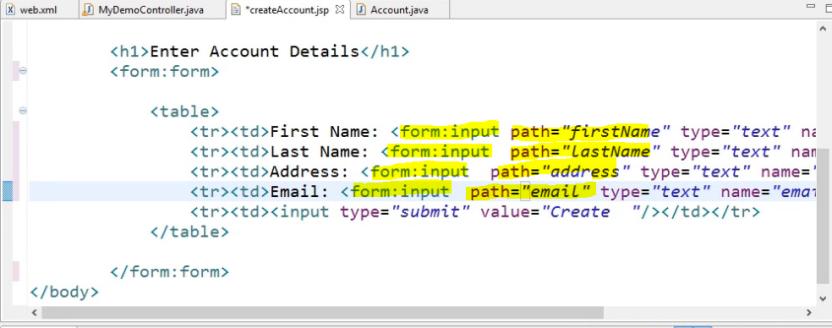


Create domain model for data binding

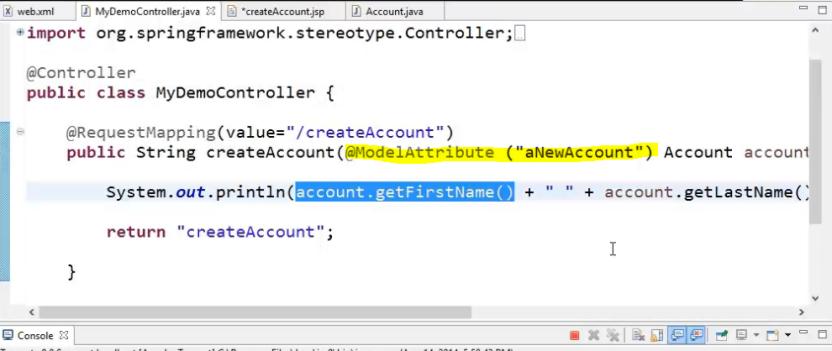


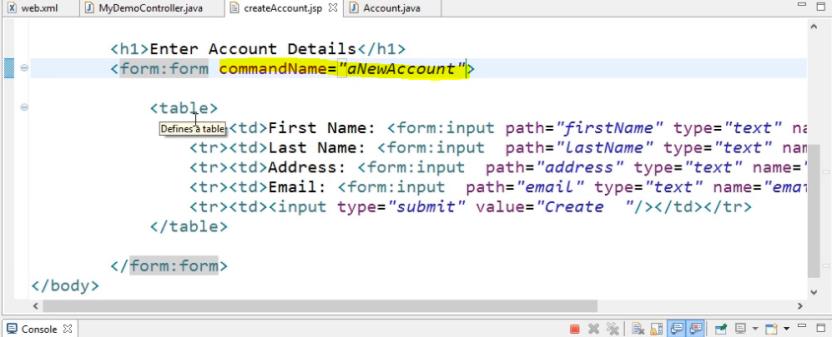
// generate getter and setter

Convert basic html input to spring tag. Path attribute will enable when we are declare spring form tag. Path attribute is used for bind the data attribute

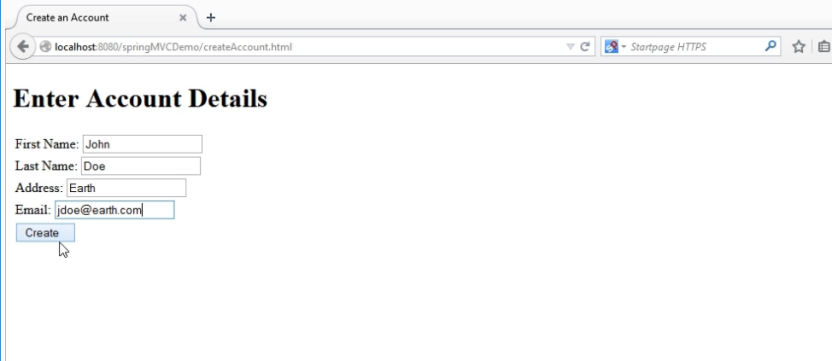


Bind the data from UI to domain model we have to use Model attribute





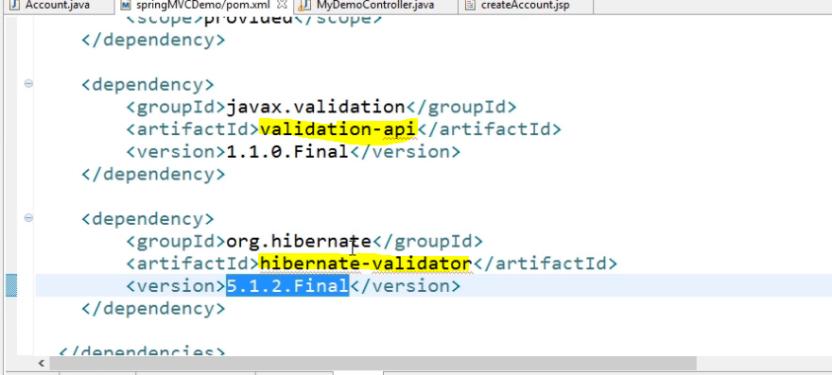
Run the Application



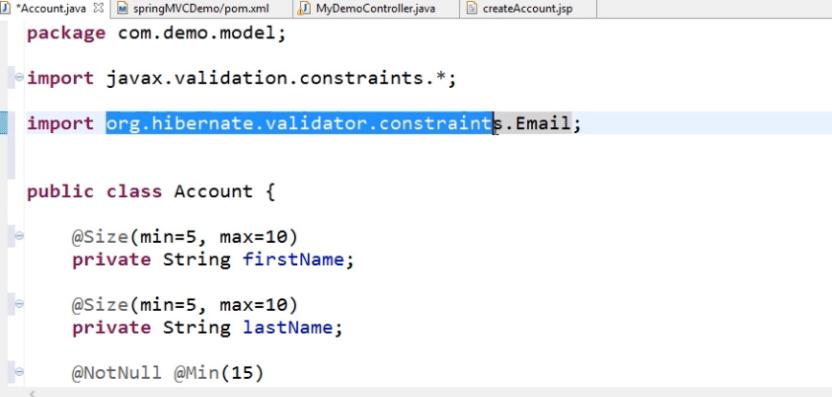


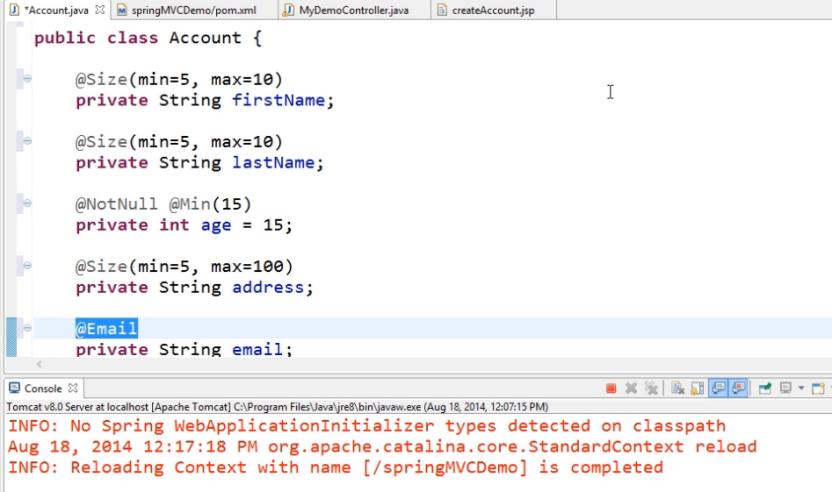
## 2.2 Validating SpringMVC Form

### 2.2.1 Add the dependency

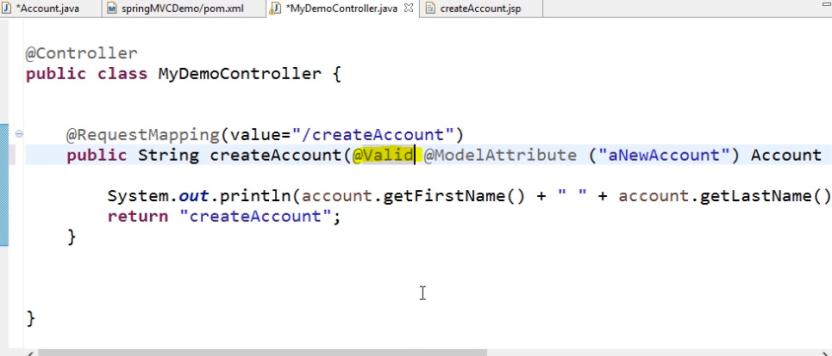


### 2.2.2 Add the validation in domain model

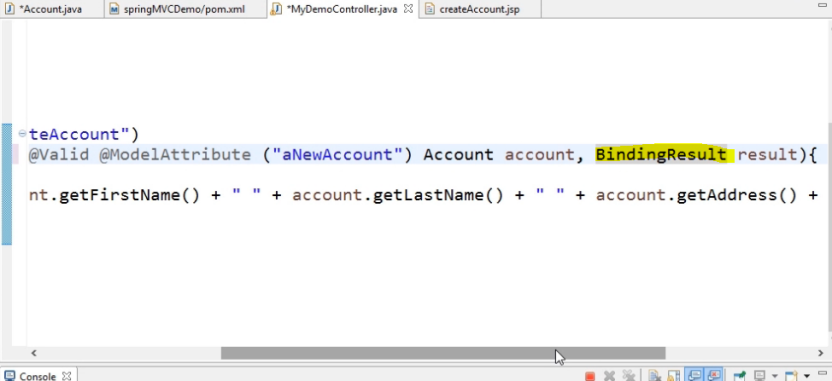




### 2.2.3 Enable validation attribute in Controller class

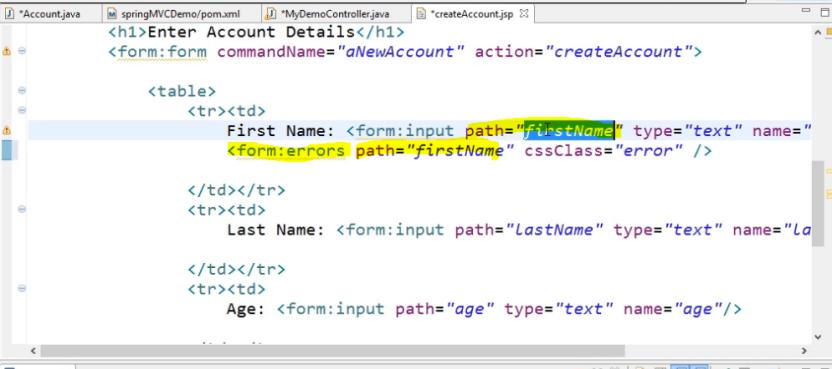


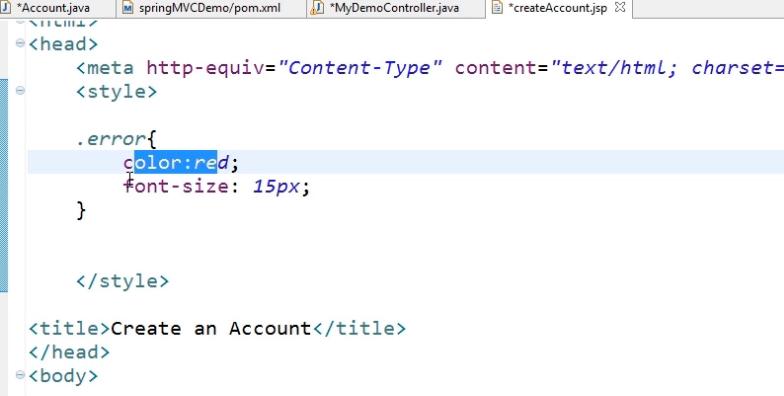
### 2.2.4 Binding error result in Controller class

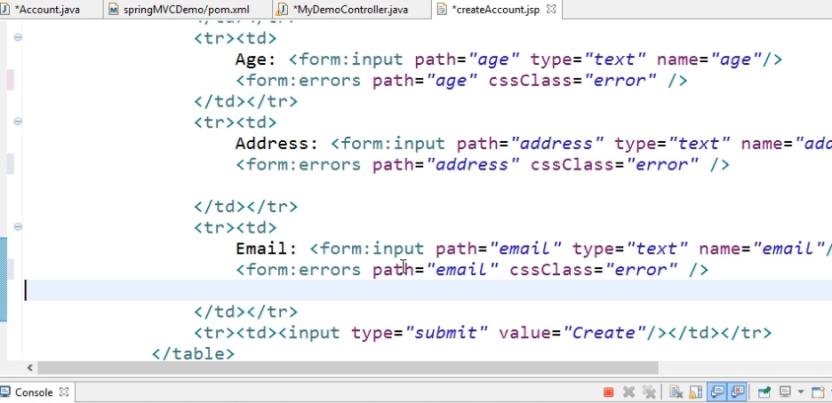




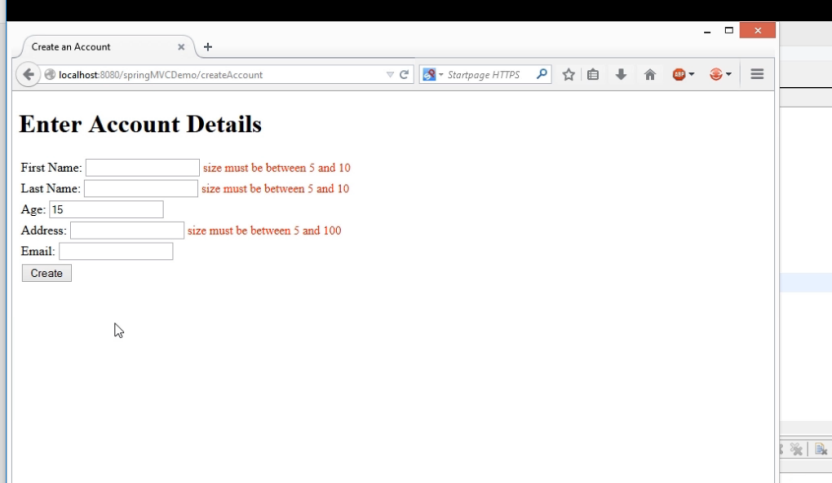
### 2.2.5 Display error in JSP page





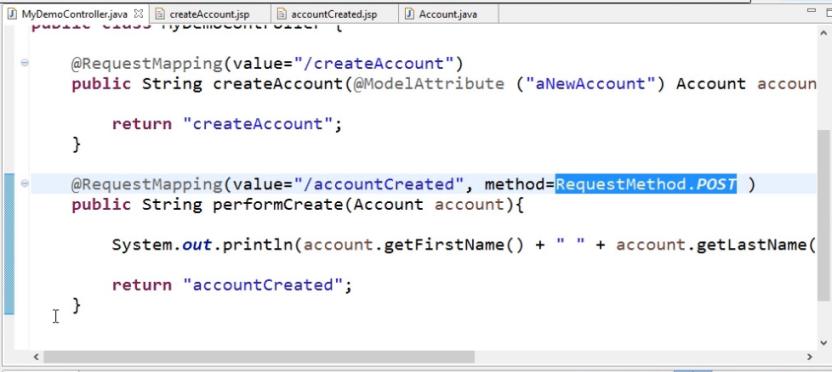


### 2.2.6 Run the Application

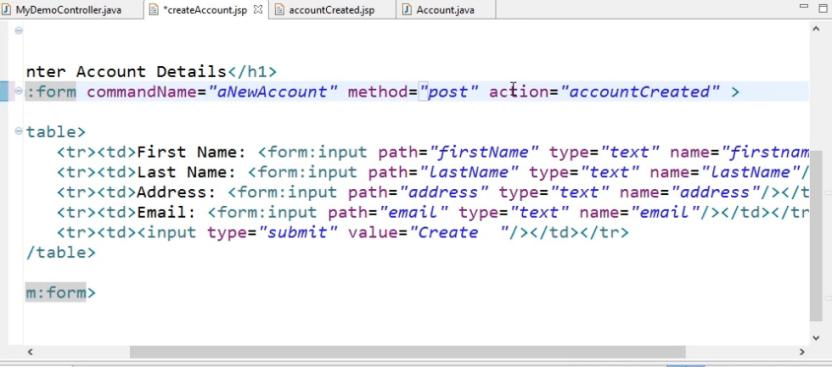


## 2.3 Narrowing Request Mapping Using Http Request Method

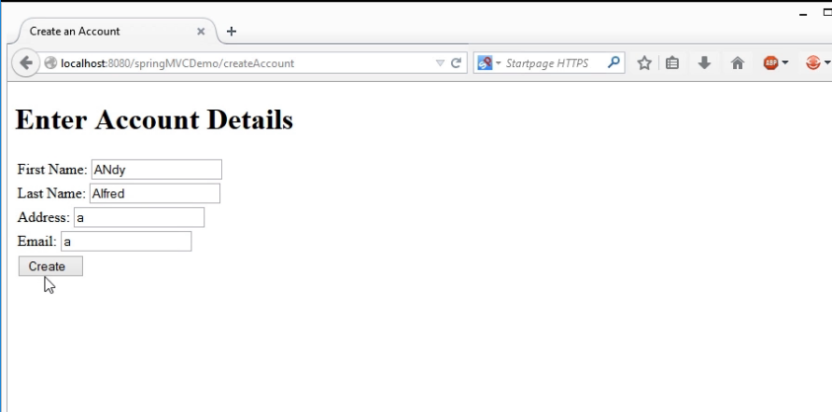
### 2.3.1 Add the Request Method in Request mapping in Controller

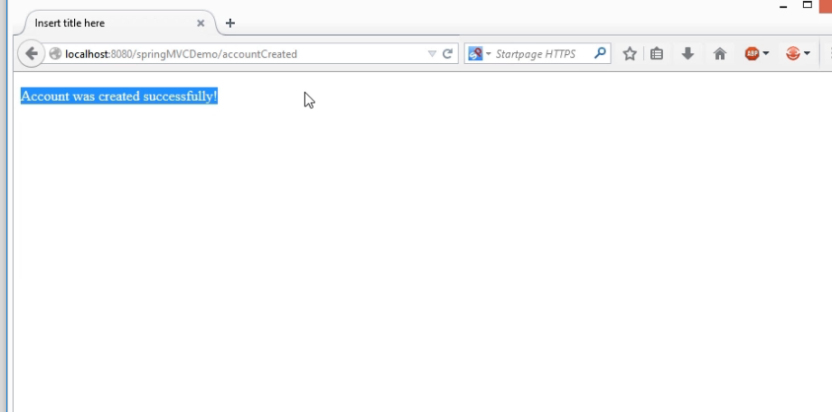


### 2.3.2 Add the Request Method in Request mapping in UI



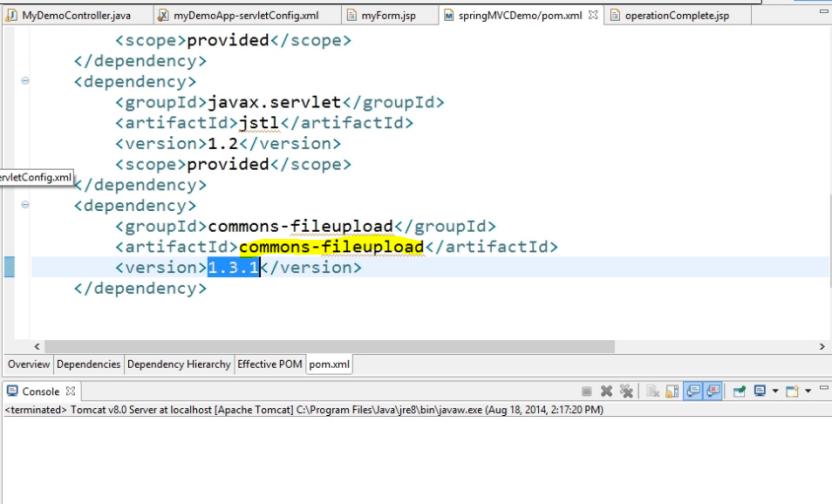
### 2.3.3 Run the Application



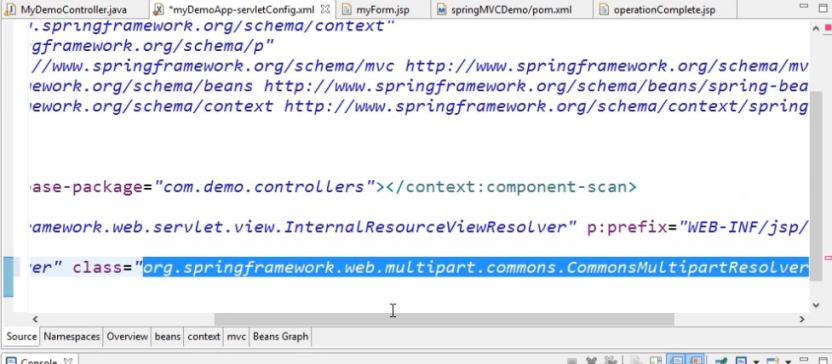


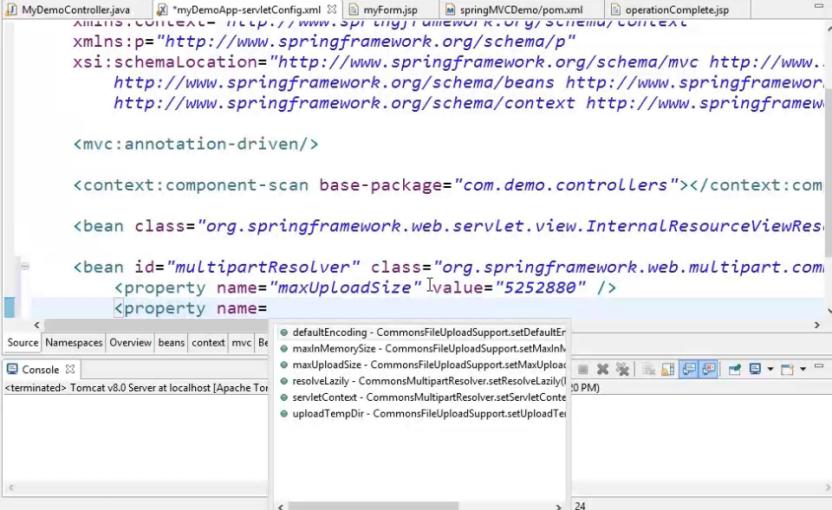
## 2.4 File Upload using Commons File Upload

### 2.4.1 Add the Dependency



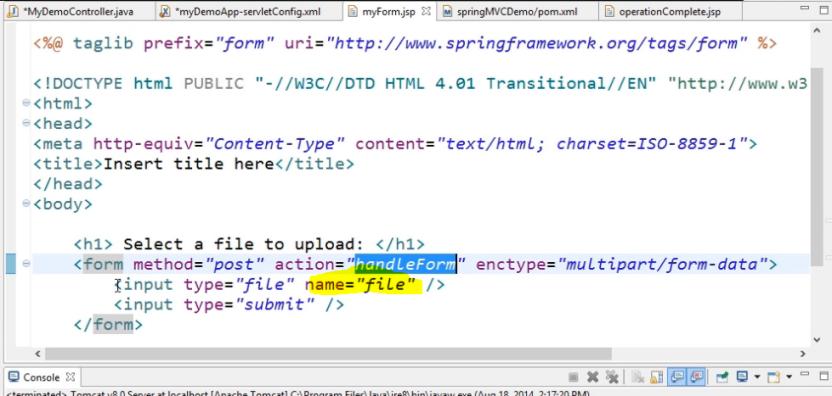
### 2.4.2 Add the Multipart configuration in servletconfig xml file



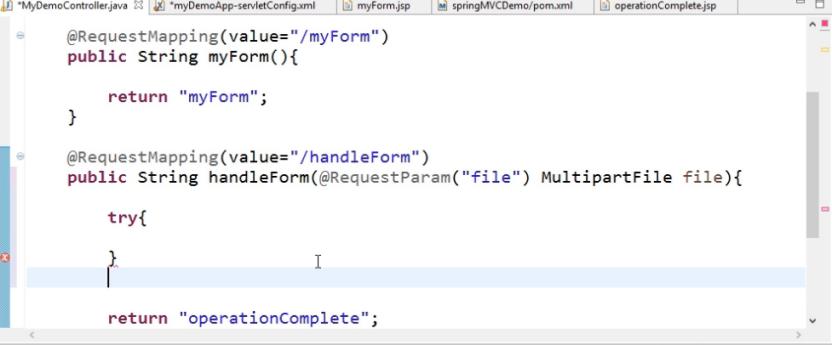


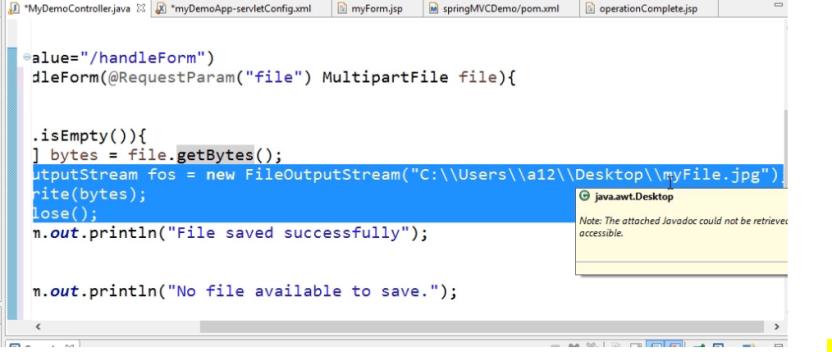


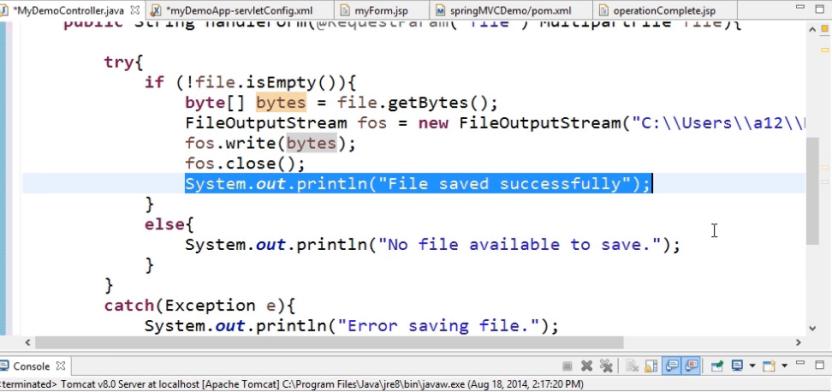
### 2.4.3 Add the file upload tag in JSP page



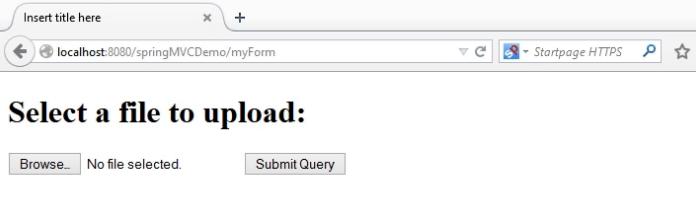
### 2.4.4 Read the File from UI to Controller

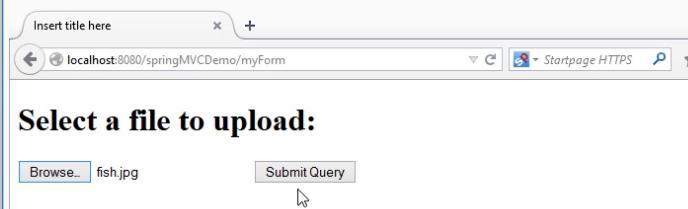


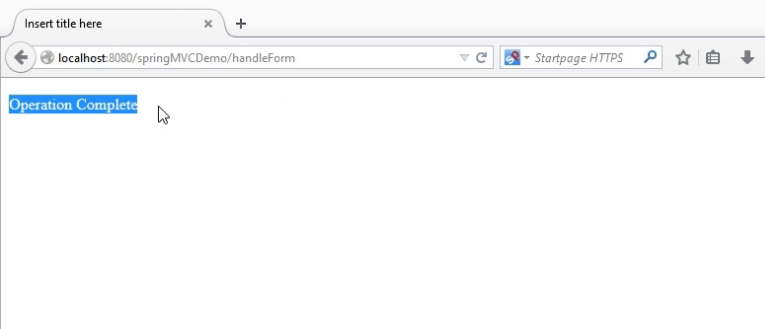


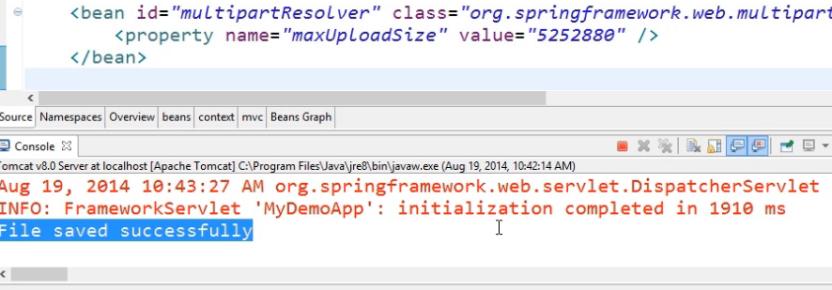


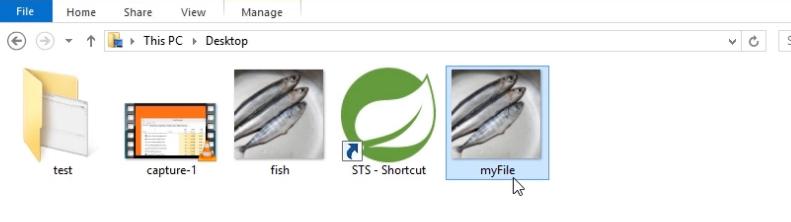
### 2.4.5 Run the Application



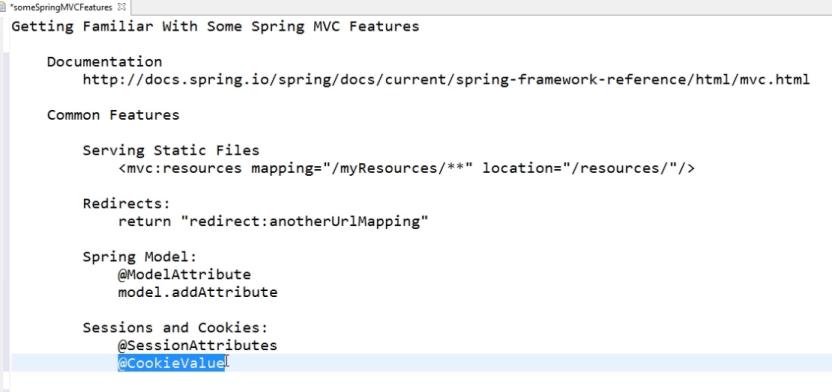




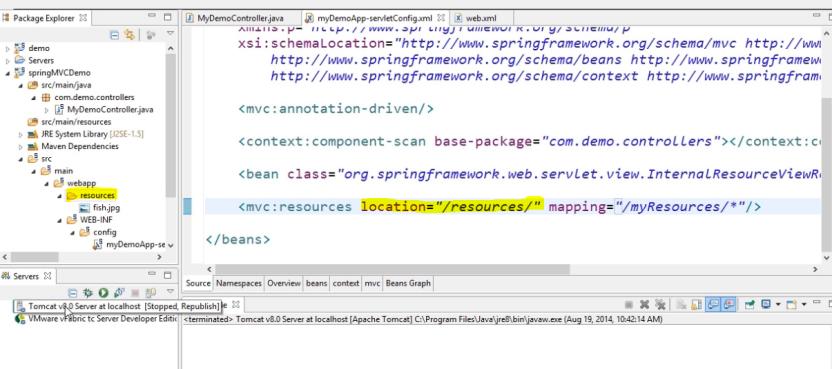




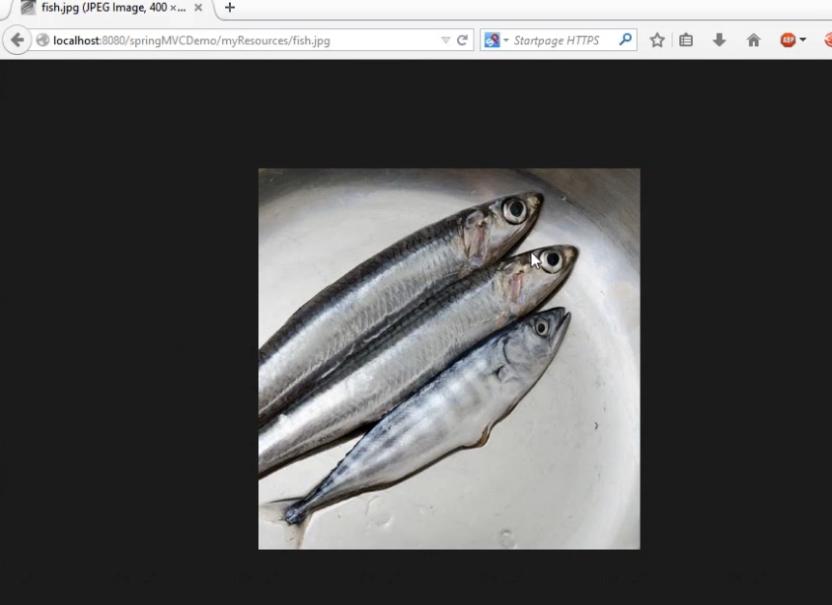
# 3. Some more Feature of SpringMVC



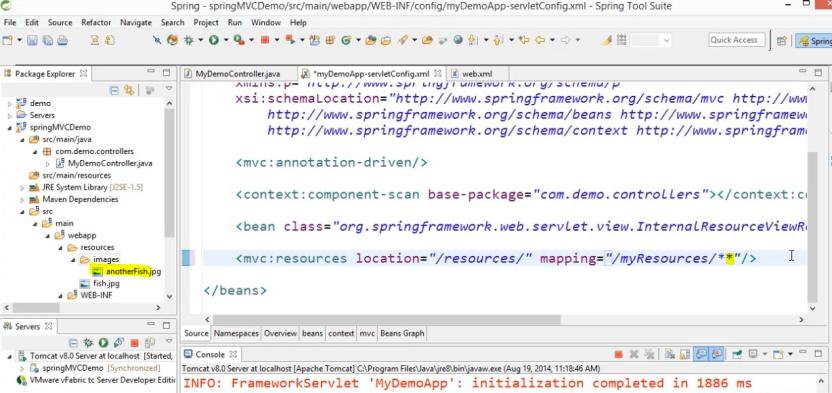
### 3.1 Serving Static Files from SpringMVC



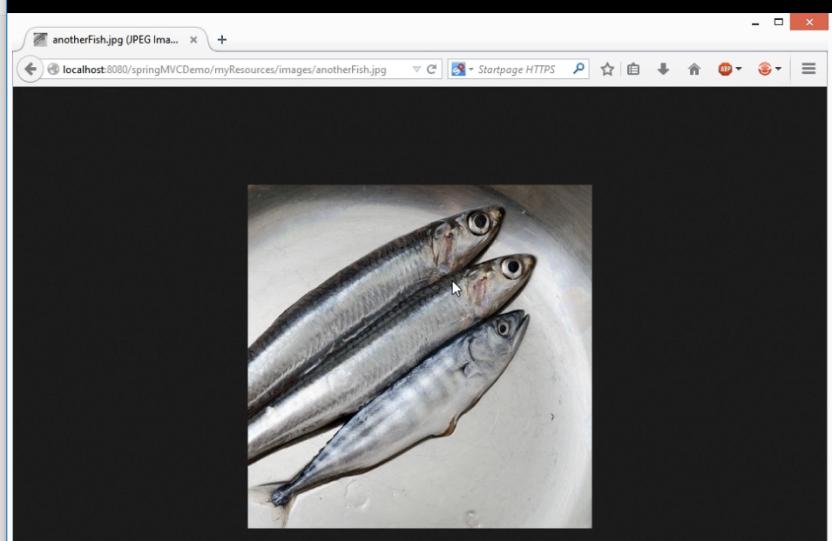
Run the application



#### 3.1.1 Serving Static Files from subfolder



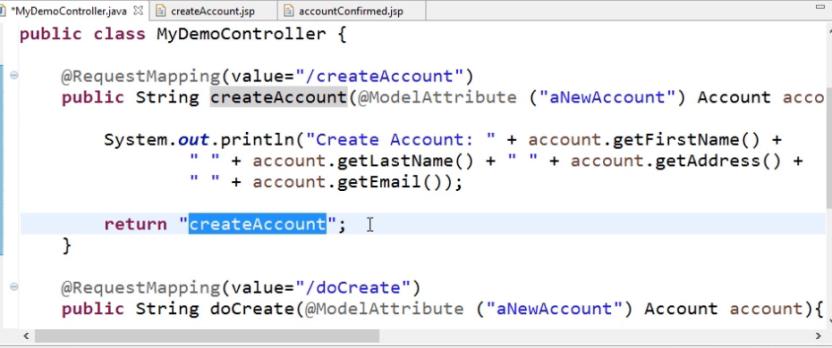
Run the Application



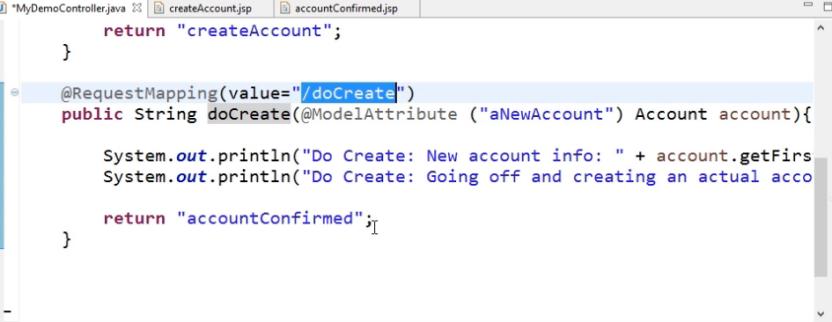
### 3.2 Redirecting in SpringMvC

### 3.2.1 Redirect from UI Click on Submit button

Initial page loading request



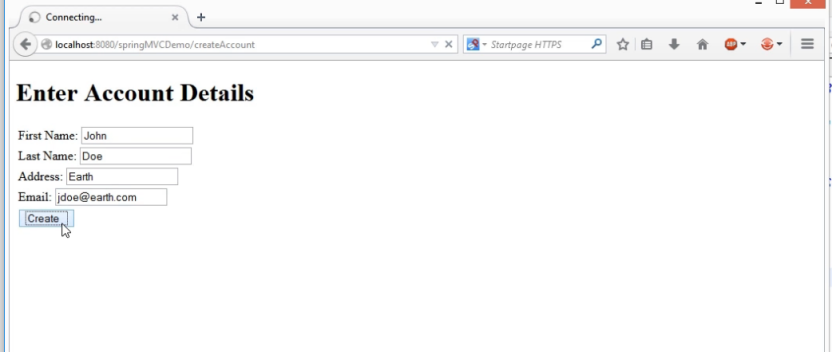
After click ok it’s call to doCreate call



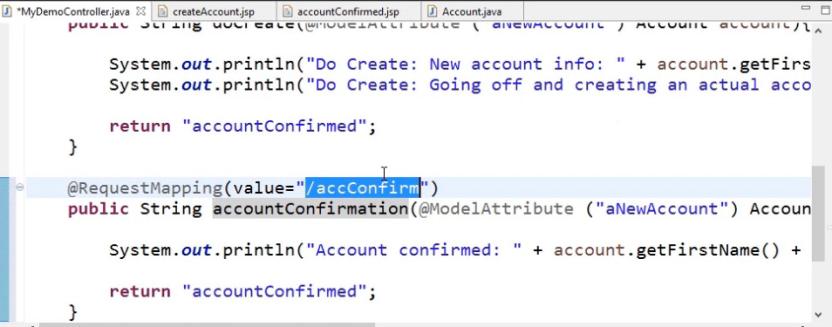
UI



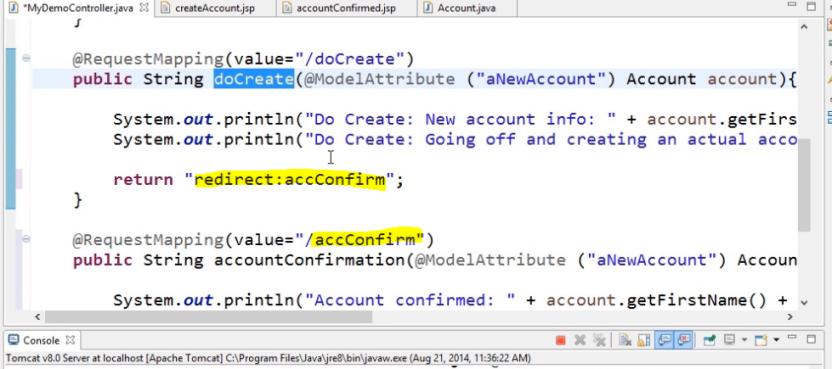
Run the Application



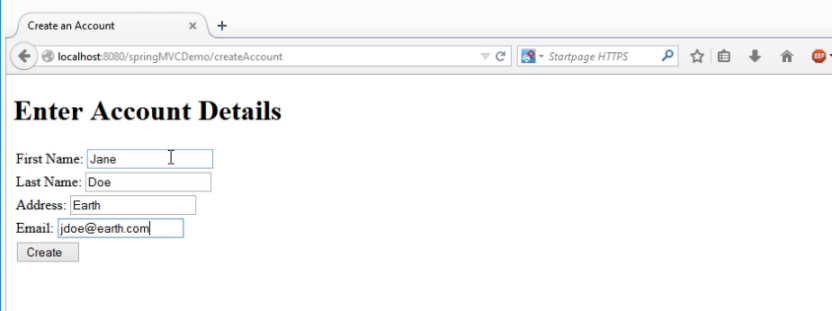
### 3.2.2 Redirect directly from Controller

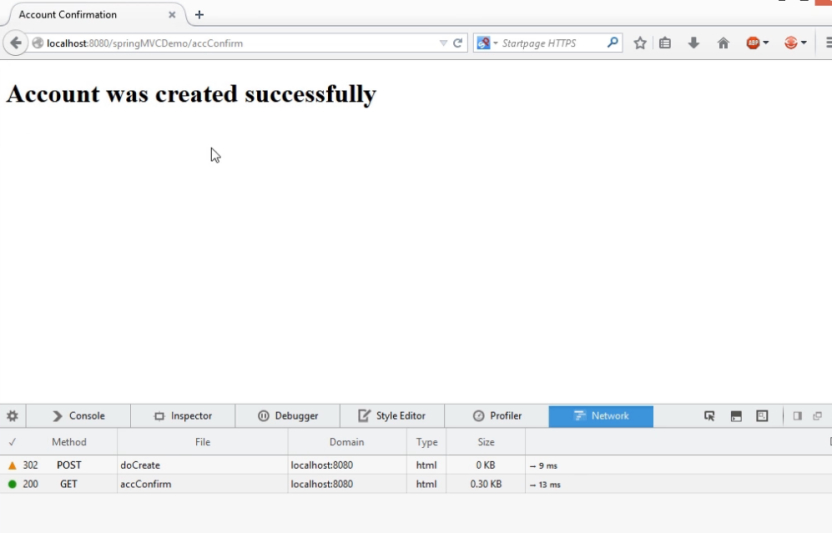


Instead of directly redirect to accountConfirmed page we are going to redirect to call accConfirm call

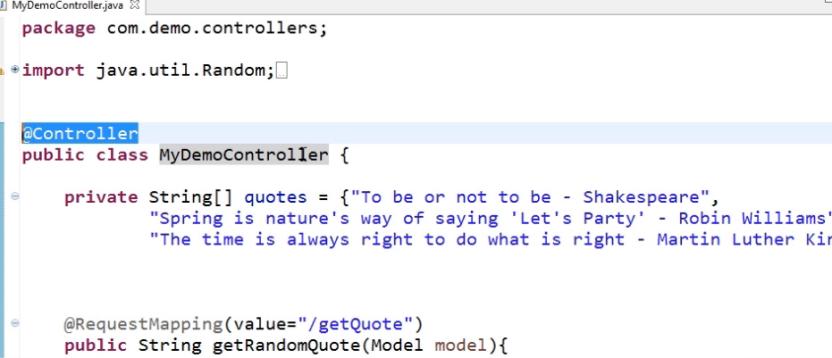


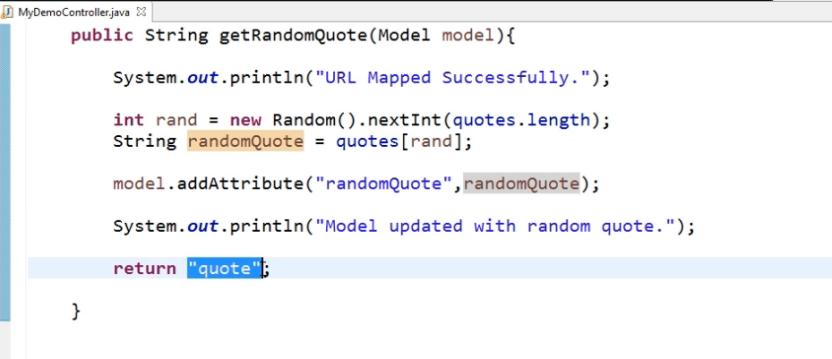
Run the Application

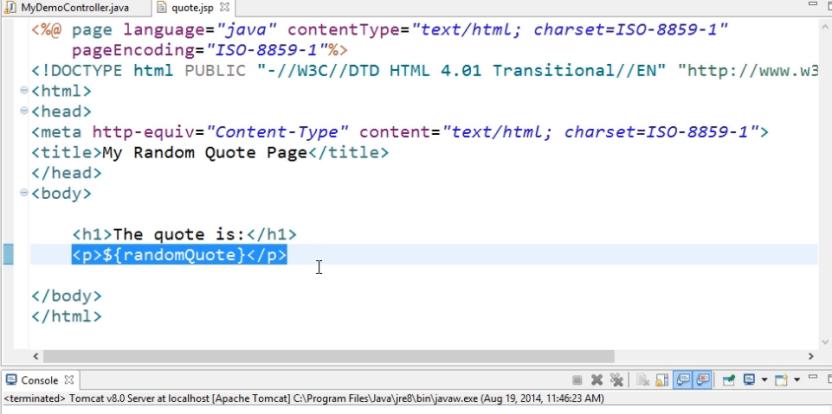




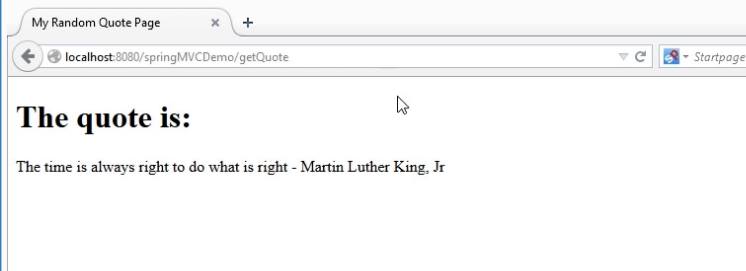
### 3.3 Enhance the Spring Model



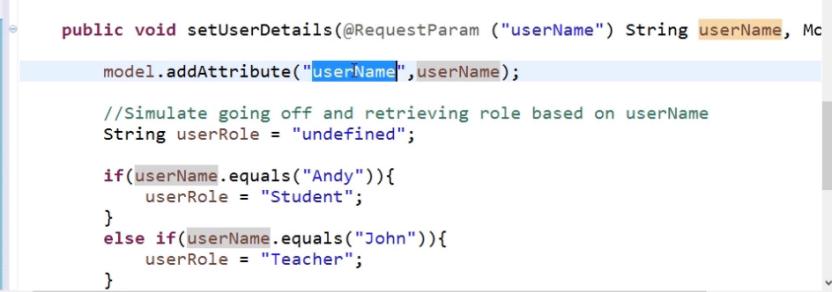


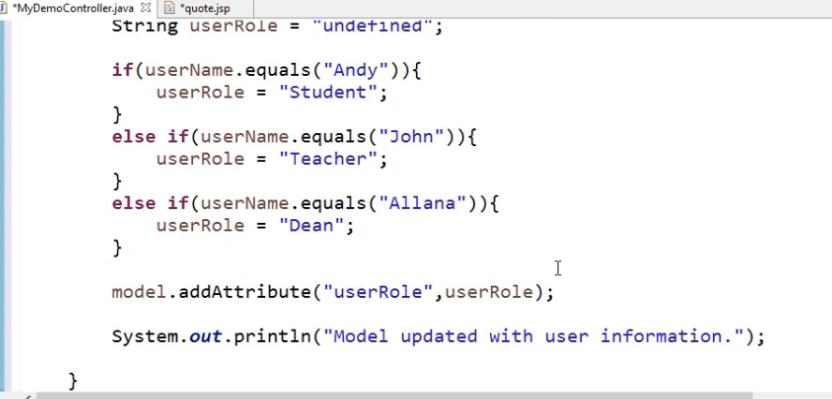


Output

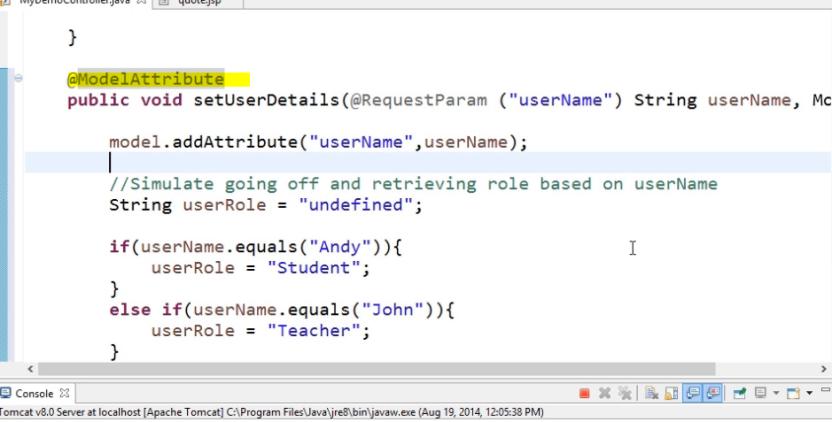


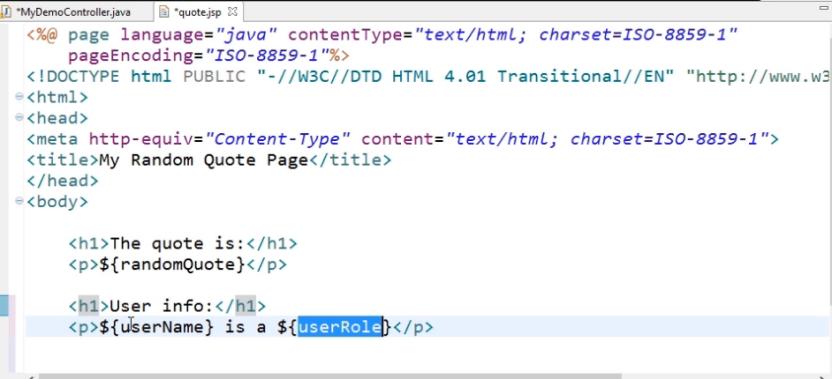
### 3.3.1 @ModelAttribute





Above setUserDetails Method should run before getRandomQuote method. If any method are annotation with Model Attribute that’s take more priority means it will run the before. Why we are running this method because we are getting queryparam from Url and then display to UI in quote.jsp



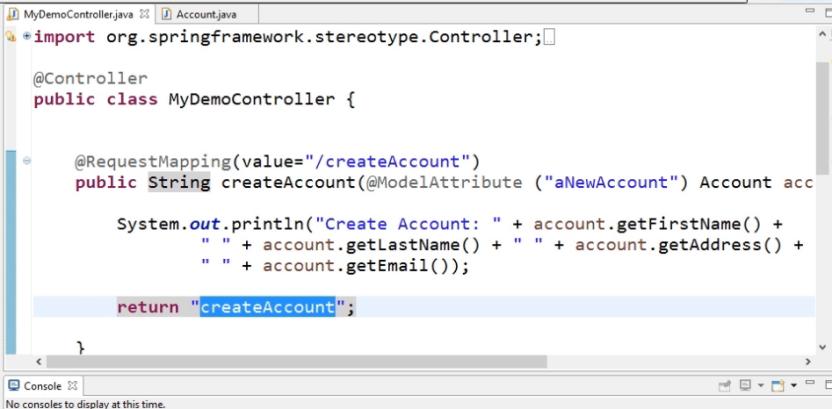


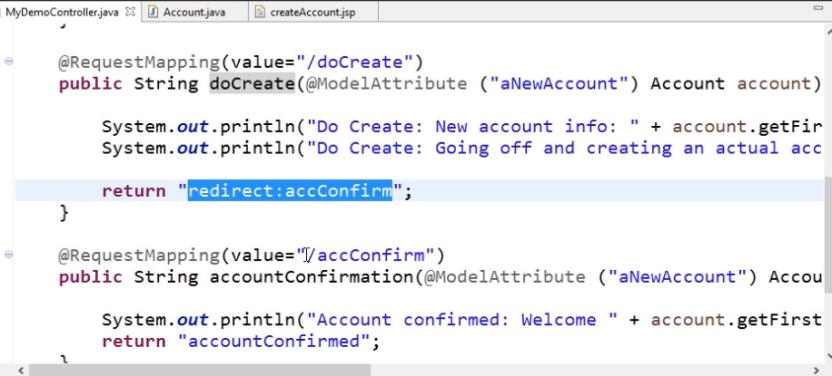
Run the Application

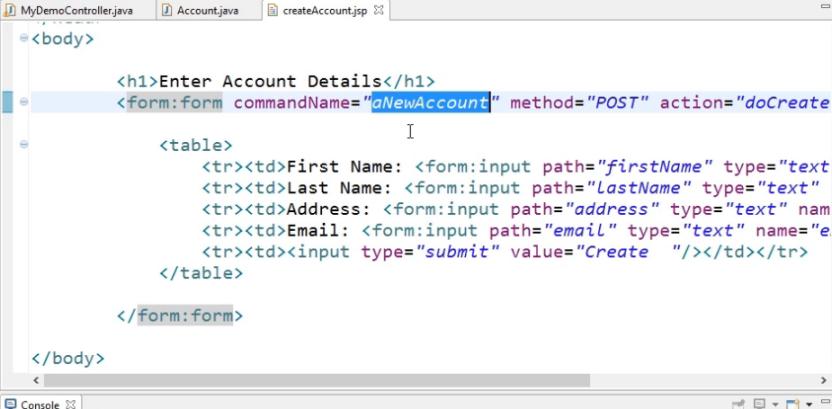


### 3.3.2 @SessionAttribute

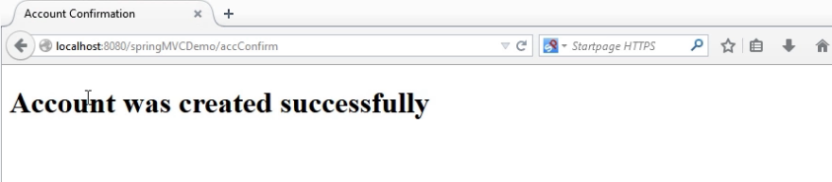
Without store a session attribute when we are going to redirect the page we will receive the null the to the redirected page

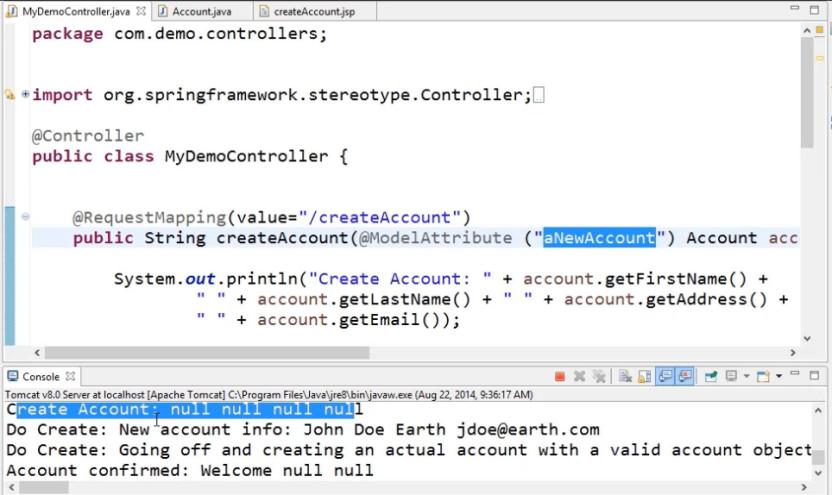






Run the Application



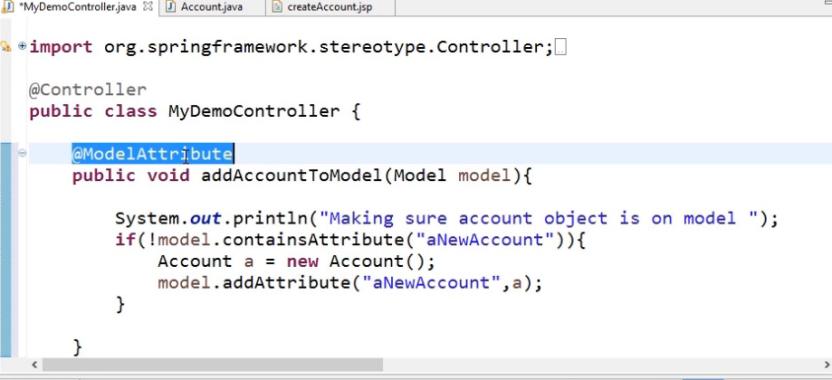


Initial page Account object is null show all are display as null

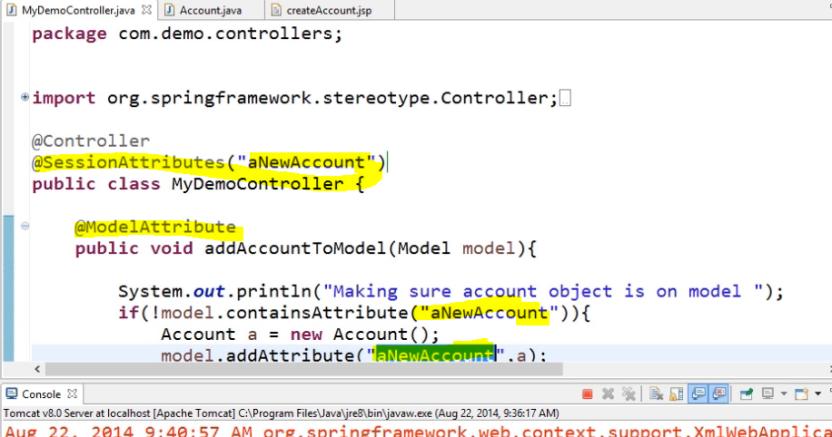
After click the button it will call to doCreate method that time we are receive the account object values after doCreate method is call to accConfirm handler method that we are getting null value

. In this situation we are going to use Session attribute

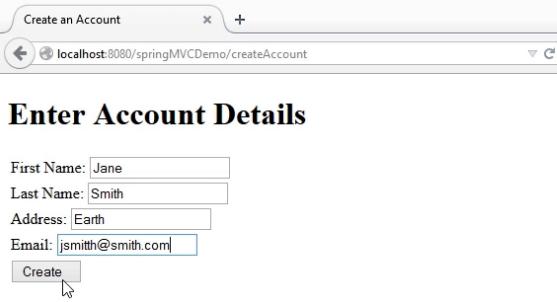
1. First we have to create a method and annotation with @ModelAtribute . In this situation we are going to store the aNewAccount object in to session. @Model Attribute is call prior before call any handler method



1. Annotation class with session attribute



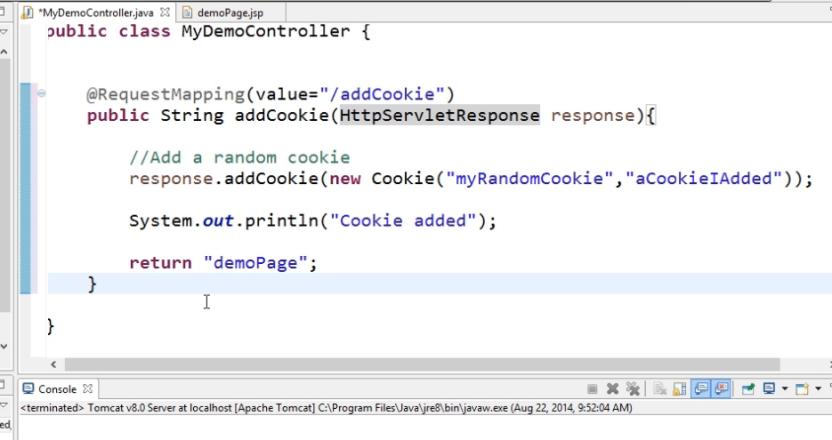
Run the application

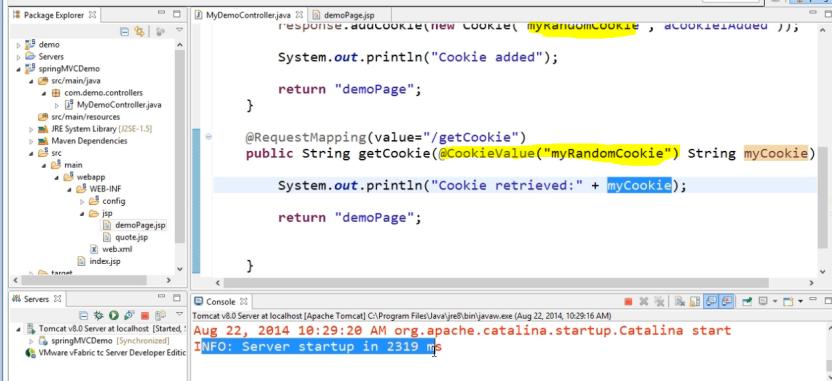


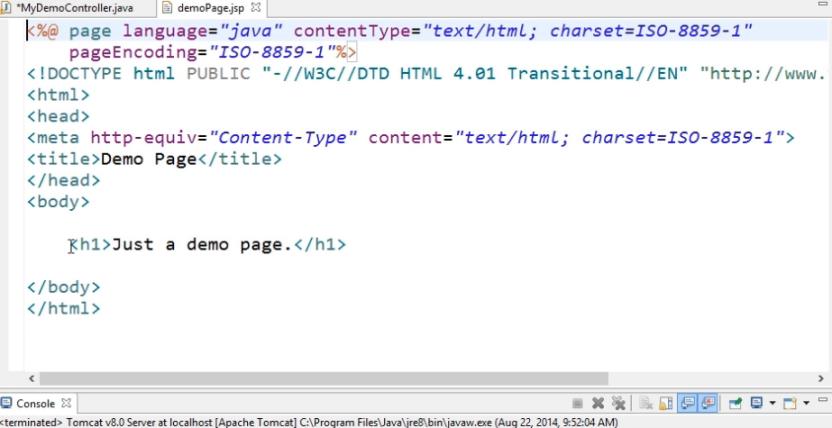




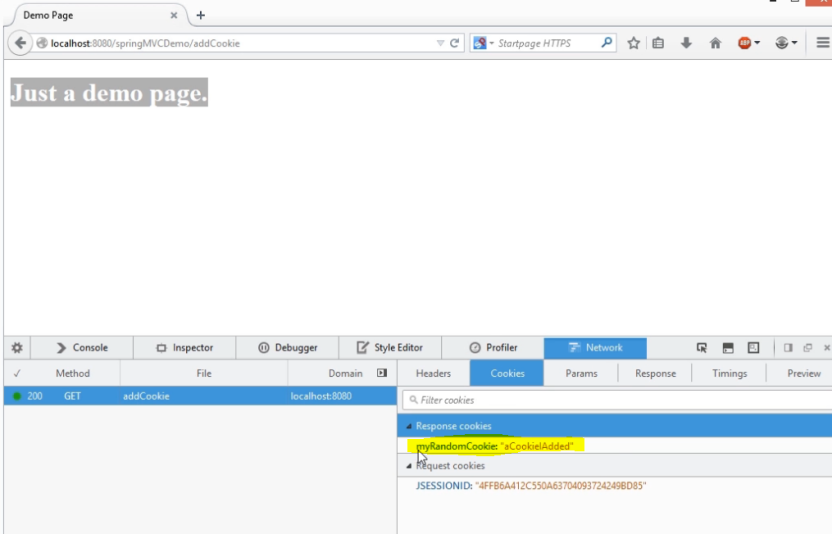
### 3.3.3 Working with cookies

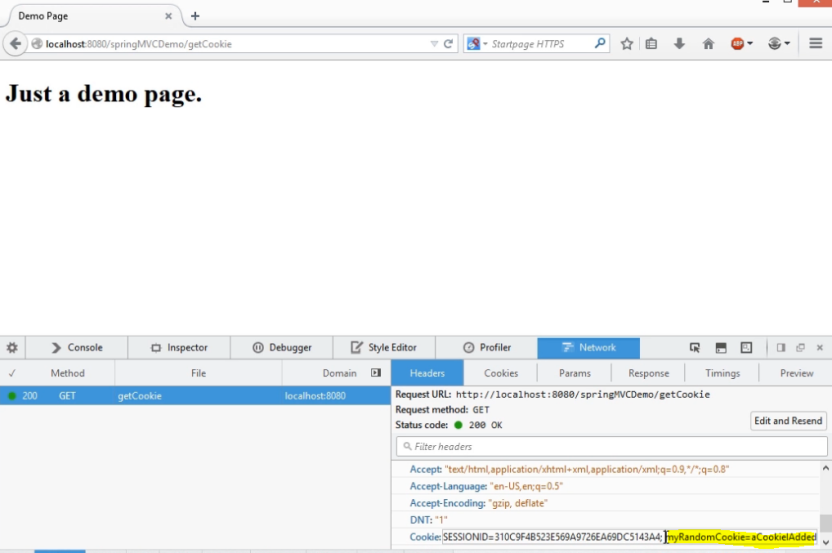






Run the Application



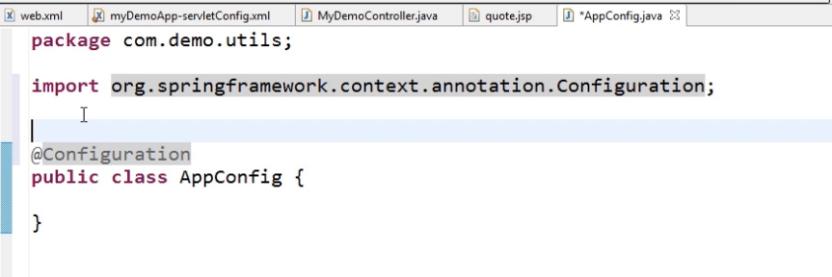


## 4. SpringMVC App with Java Based Configuration

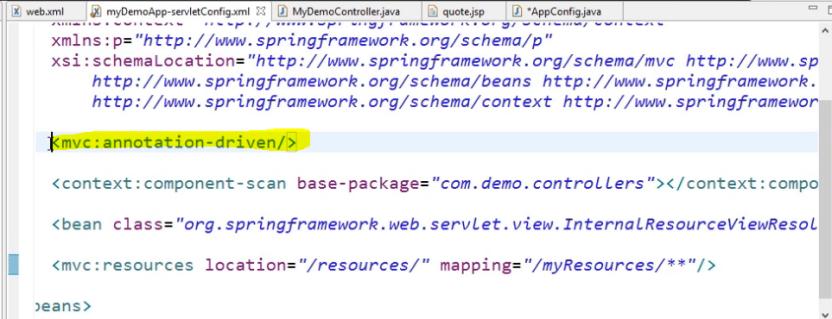
Rather than use servletconfig.xml ( applicationContext.xml ) we can configure Java based configuration



### 4.1 Create the Configuration Class

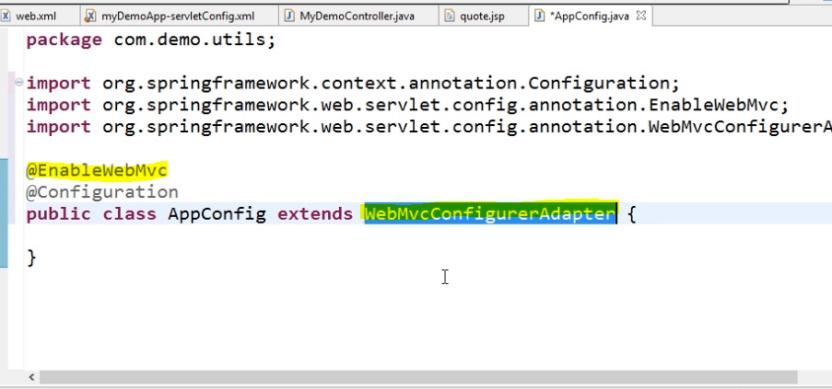


### 4.2 Enable @WebMVC Annotation and extends WebMvcConfigureAdaptor



@EnableWebMVC annotation is basically use rather than xml <mvc:annotation-driven/> tag to Java Configuration.

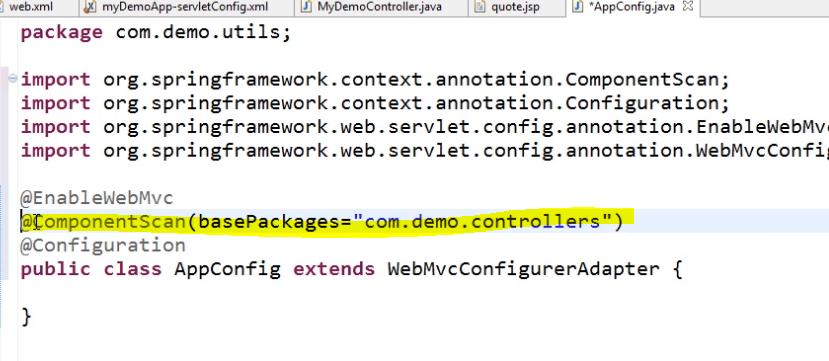
The configuration class extends with WebmvcConfigureAdaptor class will enable default configuration. if we want to we can override the default configuration



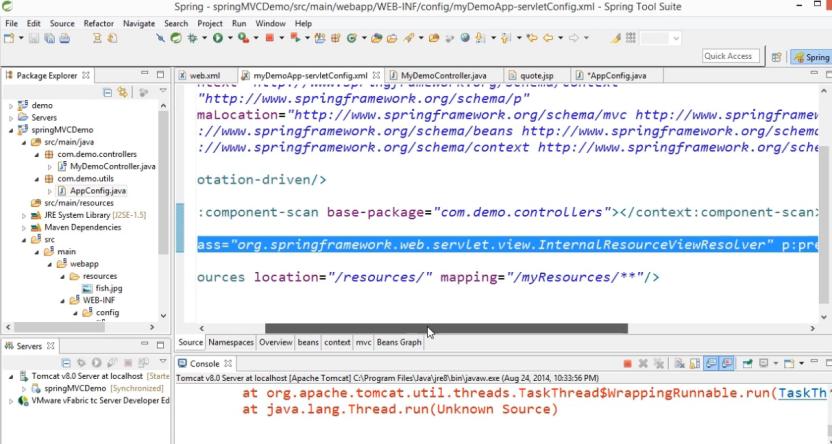
### 4.3 Enable ComponentScan

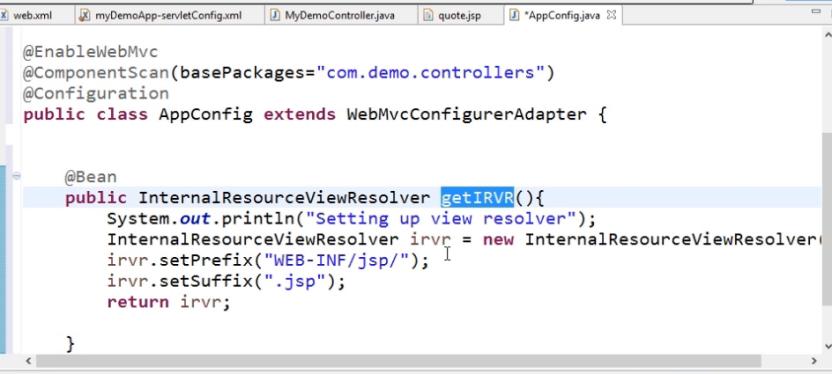


Enable component scan in Java based configuration.

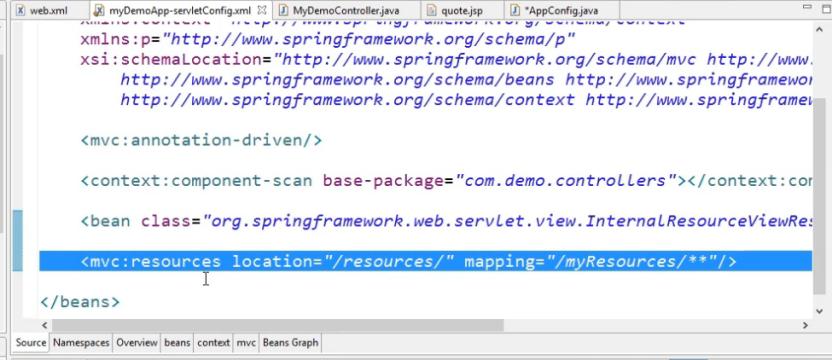


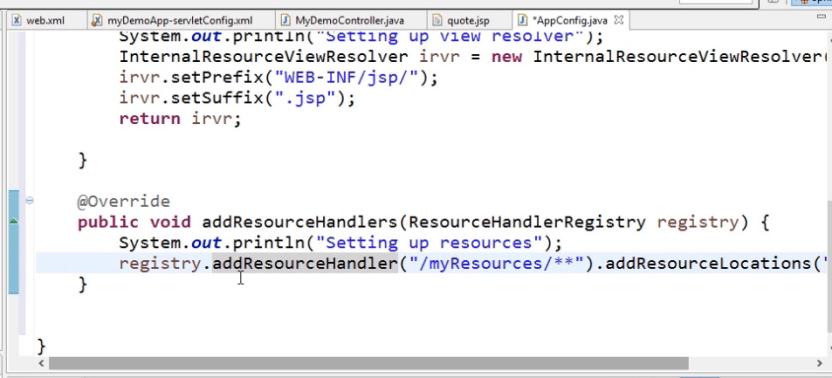
### 4.4 Enable Bean in Java Configuration

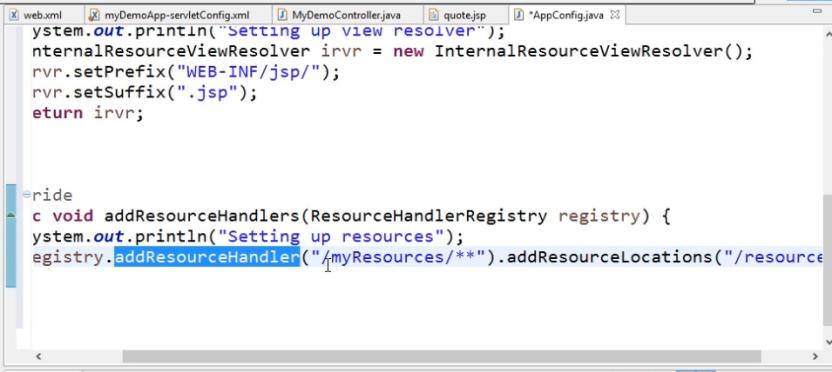




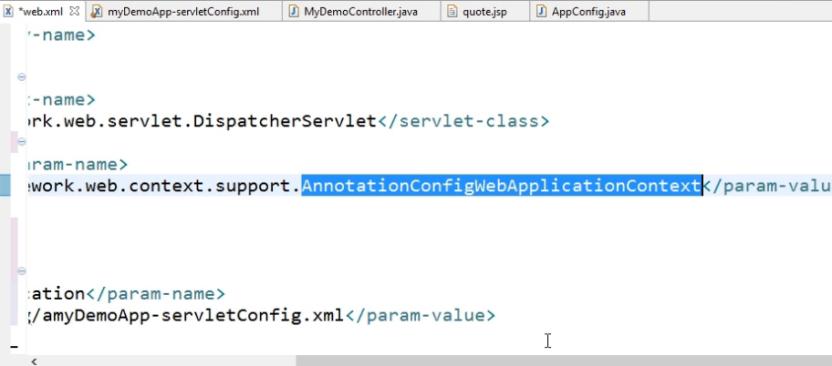
### 4.5 Enable Resources

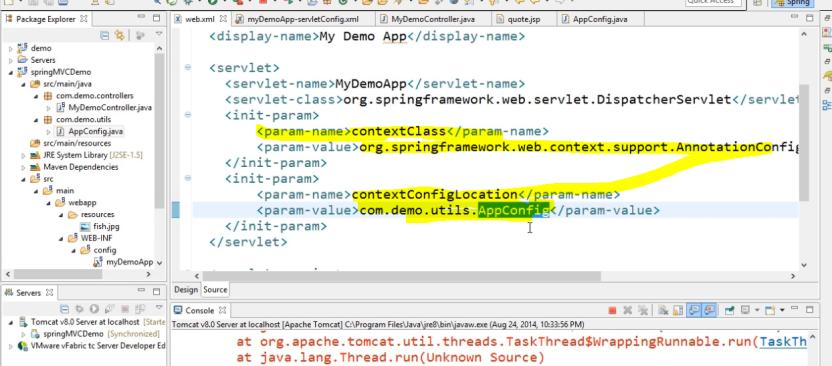




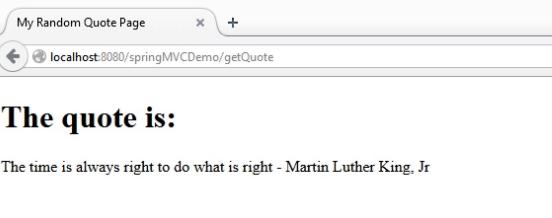


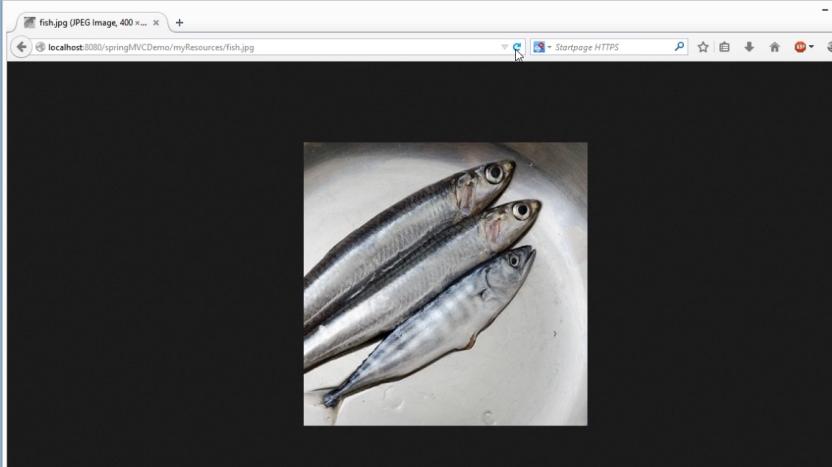
### 4.6 Add the Context Class in web.xml



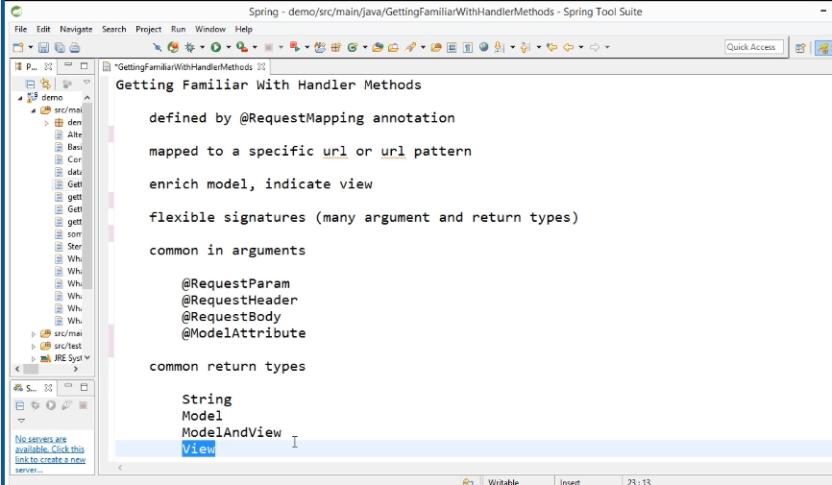


### 4.7 Run the Application



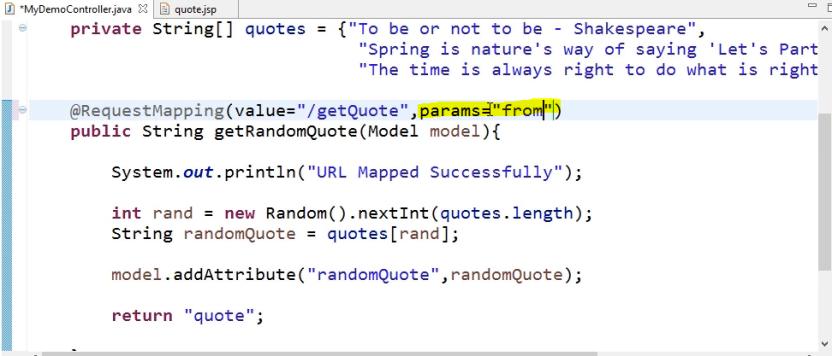


# 5. Spring MVC Controller Handler Methods

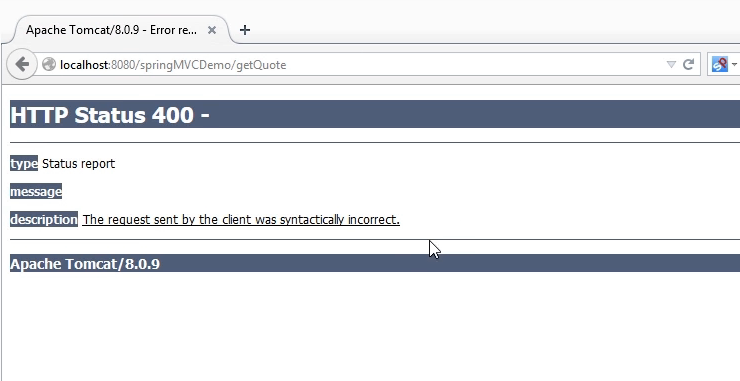


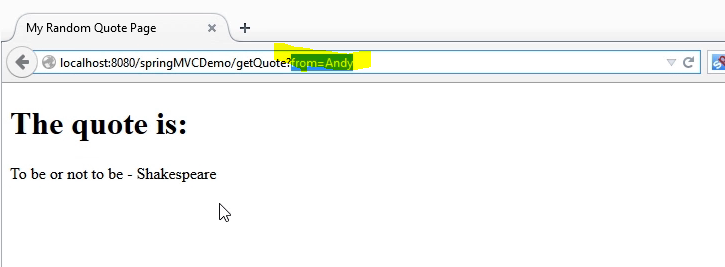
### 5.1 Matching Request Based on URL Parameters

Mapping should work when the url has parameter from then only it will access the method

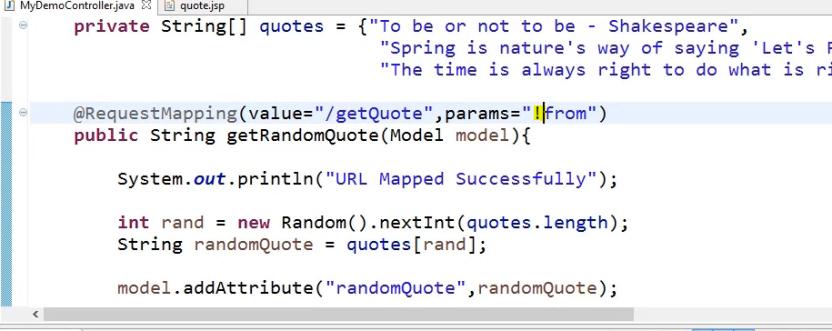


Run the Application

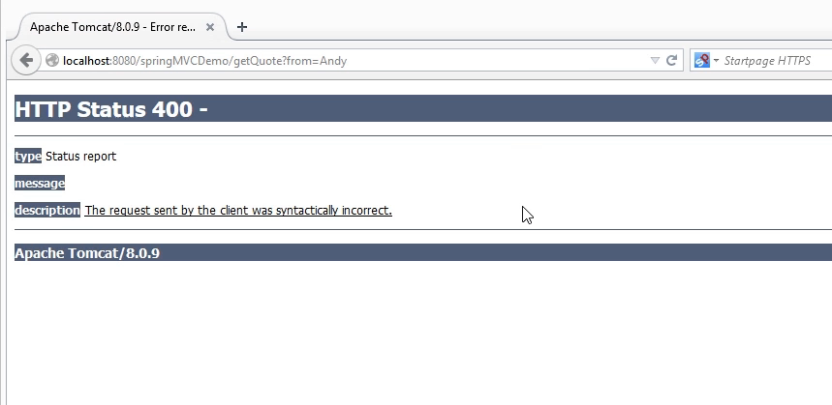




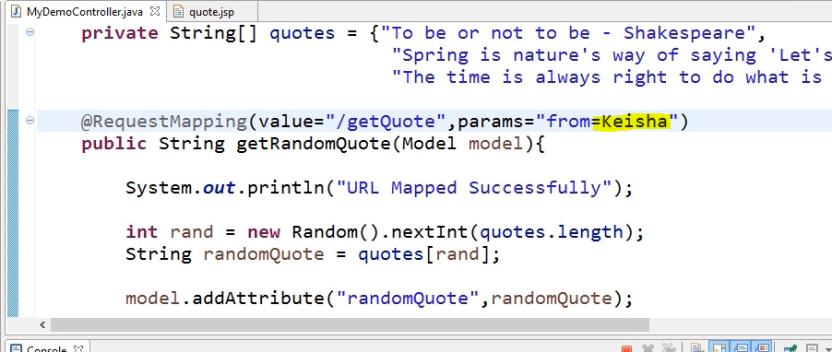
If the URL prams not contains from then below method will run



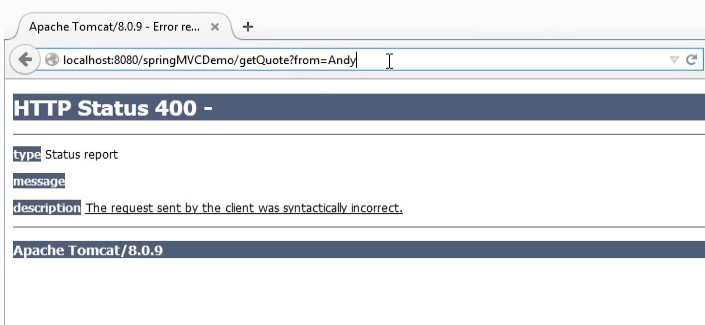
Run the Application

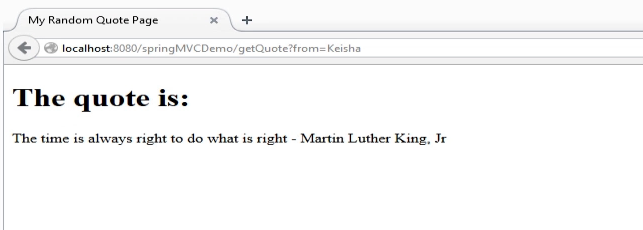


If the URL prams values is match from then below method will run



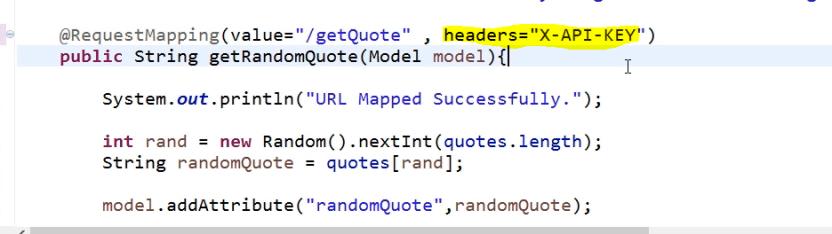
Run the Application





### 5.2 Matching Request Based on Headers

Restrict the URL we are going to add the headers.

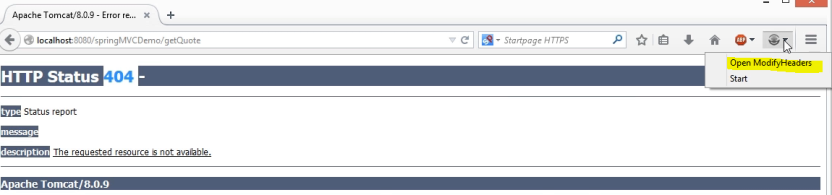


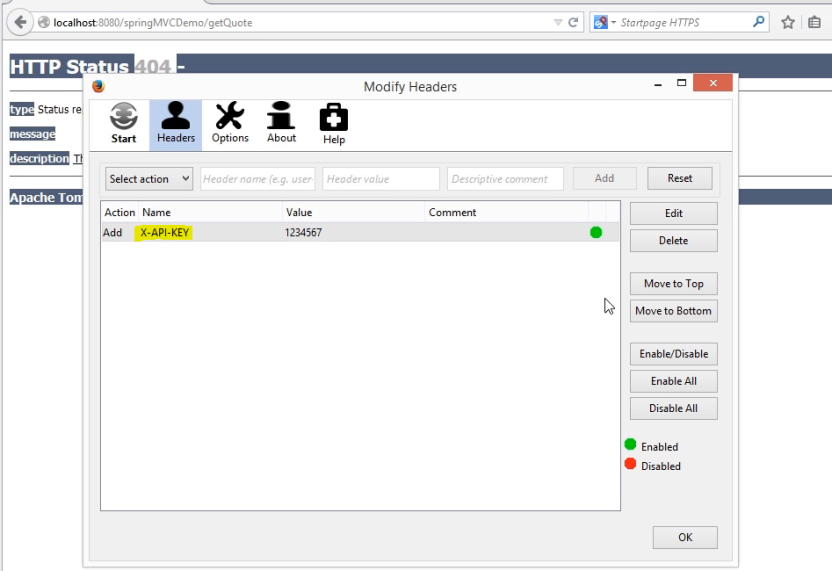
Run the Application

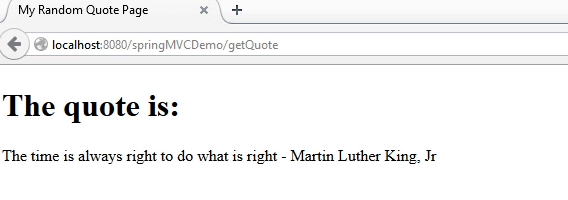
Without set headers in the request



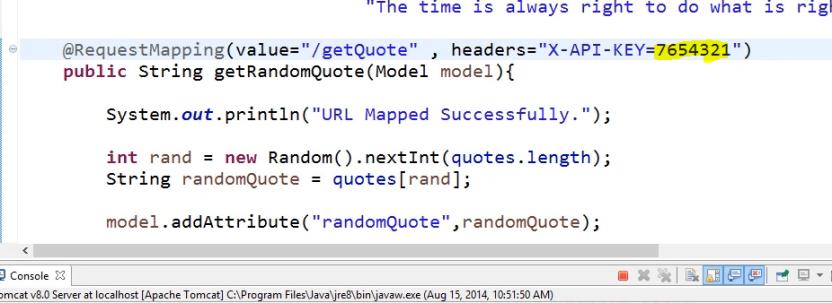
Set the headers using Add-ons





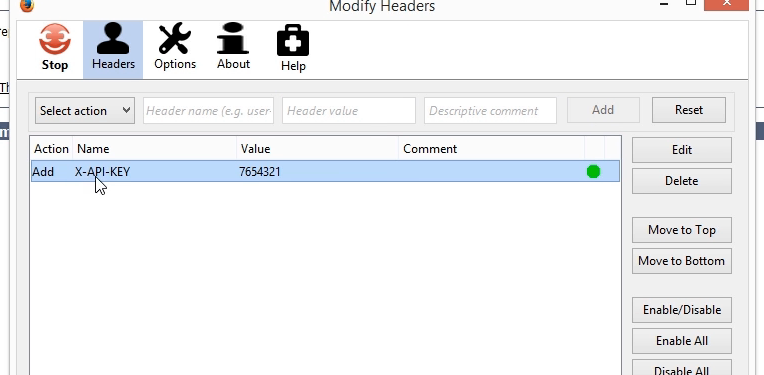


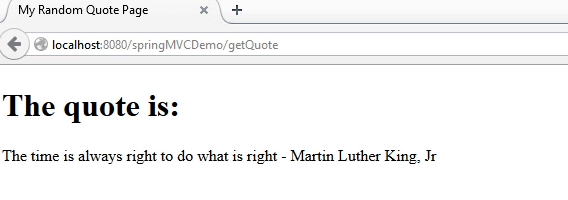
Set the Headers values



Run the Application

Wrong heads values we will be get the error. If we set the exact headers value then it will access the url





### 5.3 Accessing Request Parameters in Handler Methods



