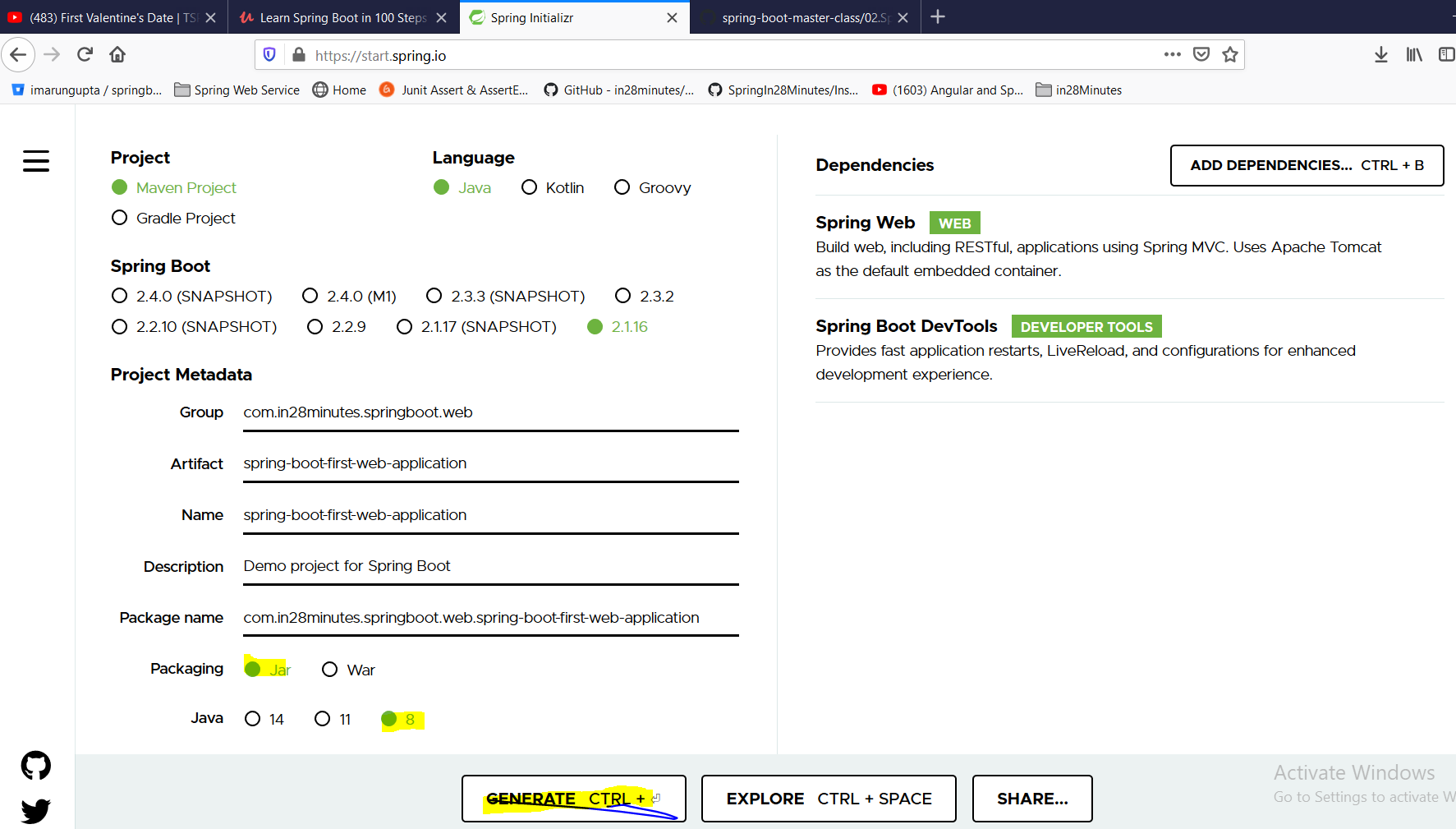


**9. Step 01: Part 1 Basic Spring Boot Web Application Setup**

1- We would use spring initializer to setup our project.

2- We will get a sample web application from <https://start.spring.io/>

3- One of the important projects that the spring boot provides is called spring initializer hosted at <https://start.spring.io/>. This helps us to setup web application



**Group**: defines the package of the project

**Artifact**: Defines the name of the project

**Packaging** : jar

**Dependencies** : Spring Web and Spring Boot DevTools

**Spring Web Web**Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container

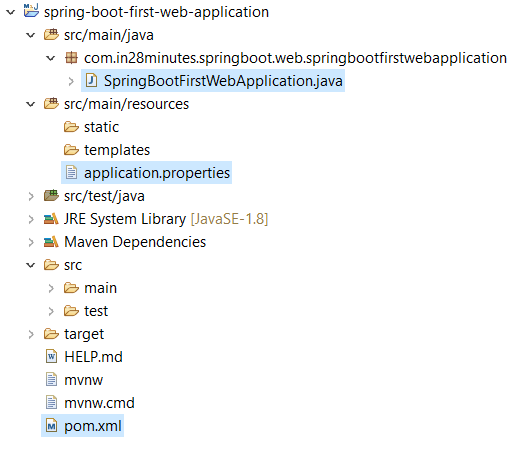
**Spring Boot DevTools Developer Tools** Provides fast application restarts, LiveReload, and configurations for enhanced development experience. This devtools makes the development faster and make the developer productive. It automatically build the project and we just need to refresh web browser to see new changes.

As soon as we click on Generate, it will generate zip folder which will contain a maven project.

Now unzip and import it as existing maven project into eclipse.

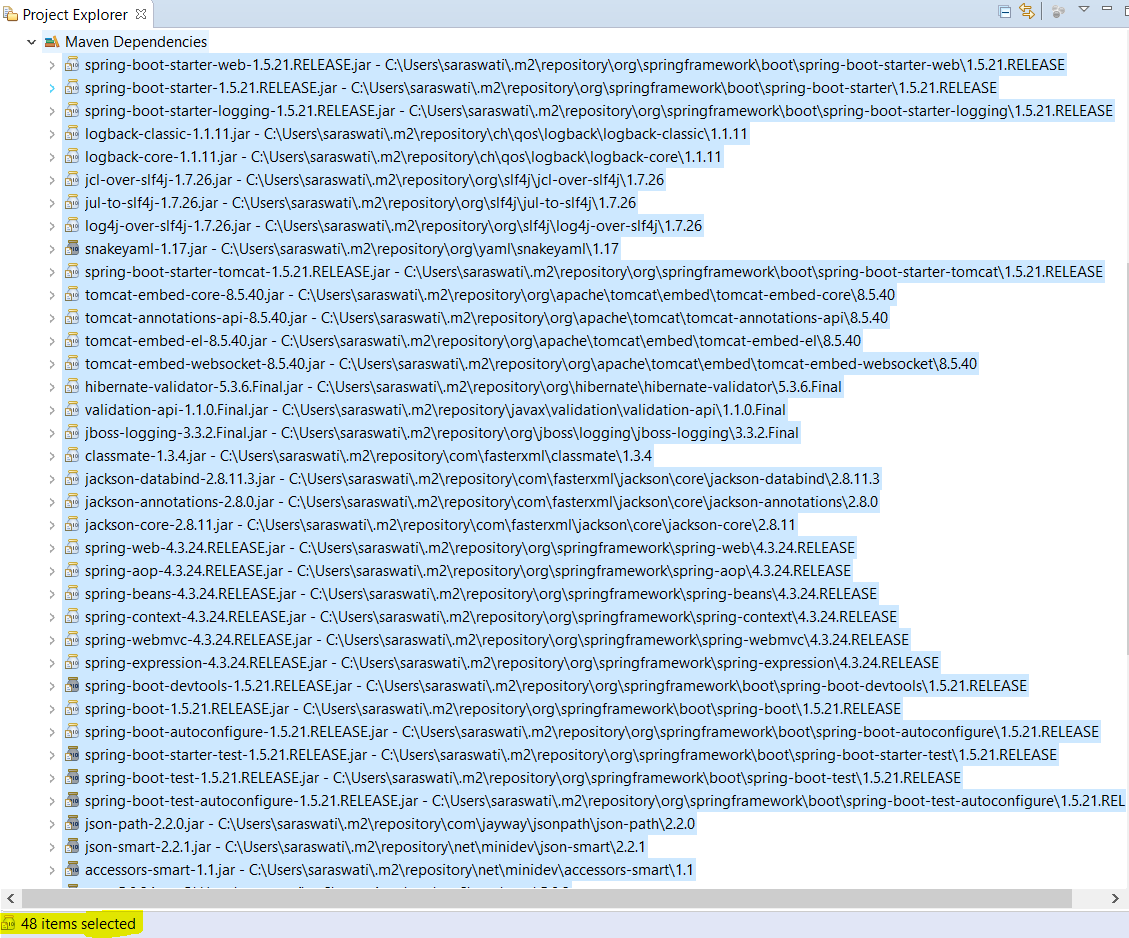
Now if we expand the project in eclipse we would see three important file

* 1. SpringBootFirstWebApplication.java
  2. application.properties
  3. POM.xml (Project Object Model) : Describes the characteristics of the project



* 1. **Pom.xml**

|  |
| --- |
| <?xmlversion=*"1.0"*encoding=*"UTF-8"*?>  <projectxmlns=*"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 https://maven.apache.org/xsd/maven-4.0.0.xsd"*>  <!-- Project Detail -->  <modelVersion>4.0.0</modelVersion>  <groupId>com.in28minutes.springboot.web</groupId>  <artifactId>spring-boot-first-web-application</artifactId>  <version>0.0.1-SNAPSHOT</version>  <name>spring-boot-first-web-application</name>  <description>Demo project for Spring Boot</description>  <! -- adding parent: It is very similar to java inheritance, just inherit parent ability into this POM.xml. This parent would manage all the dependencies version accordingly without any conflict -->  <parent>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-parent</artifactId>  <version>1.5.21.RELEASE</version>  <relativePath/><!-- lookup parent from repository -->  </parent>  <!-- java version -->  <properties>  <java.version>1.8</java.version>  </properties>  <!-- Adding dependencies web for MVC and rest, devtools for auto server stsrt, test for unit and Integration testing-->  <dependencies>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-web</artifactId>  </dependency>  <!-- Adding dependencies web for MVC and rest>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-devtools</artifactId>  <scope>runtime</scope>  <optional>true</optional>  </dependency>  <!-- devtools for auto server start 🡪  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-test</artifactId>  <scope>test</scope>  </dependency>  <!— For doing unit and integration testing in the project 🡪  </dependencies>  <!— -maven plug-in helps in running spring boot maven project and helps in creating jar or war files for spring boot projects- - 🡪  <build>  <plugins>  <plugin>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-maven-plugin</artifactId>  </plugin>  </plugins>  </build>  </project> |



All jars got downloaded and managed by Pom.xml (Maven). As soon as we add dependency in the dependency section of POM

And also it will download something called transitive dependencies. For example I am using spring dependency and for to be able to work it need some other dependency like slf4j. So now I need spring as well as its dependency slf4j as dependencies.

So the fallowing things we have seen so far in the POM.xml.

* Parent
* Dependency
* Plug-in
* Java version

2- Now the next important file of the project is SpringBootFirstWebApplication.java

This class is annotated with @**SpringBootApplication** and inside this class we have one main method. This class basically spring boot launcher class.

|  |
| --- |
| package com.in28minutes.springboot.web.springbootfirstwebapplication;  import org.springframework.boot.SpringApplication;  import org.springframework.boot.autoconfigure.SpringBootApplication;  @SpringBootApplication  public class SpringBootFirstWebApplication {  public static void main(String[] args) {  SpringApplication.run(SpringBootFirstWebApplication.class, args);  }  }  **Note**: SpringApplication.run: launches spring boot application with embedded tomcat server. |

**@SpringBootApplication**: This annotation does two things

1. It initializes the spring framework (component scan)and
2. It initializes spring boot itself called auto configuration. Spring boot run on top of spring framework

|  |
| --- |
| [-->[Open Declaration](eclipse-open:%E2%98%82=spring-boot-first-web-application/C:\/Users\/saraswati\/.m2\/repository\/org\/springframework\/boot\/spring-boot-autoconfigure\/1.5.21.RELEASE\/spring-boot-autoconfigure-1.5.21.RELEASE.jar%3corg.springframework.boot.autoconfigure(SpringBootApplication.class%E2%98%83SpringBootApplication)](eclipse-open:%E2%98%82=spring-boot-first-web-application/C:%5C/Users%5C/saraswati%5C/.m2%5C/repository%5C/org%5C/springframework%5C/boot%5C/spring-boot-autoconfigure%5C/1.5.21.RELEASE%5C/spring-boot-autoconfigure-1.5.21.RELEASE.jar%3Corg.springframework.boot.autoconfigure(SpringBootApplication.class%E2%98%83SpringBootApplication)[org](eclipse-javadoc:%E2%98%82=spring-boot-first-web-application/C:%5C/Users%5C/saraswati%5C/.m2%5C/repository%5C/org%5C/springframework%5C/boot%5C/spring-boot-autoconfigure%5C/1.5.21.RELEASE%5C/spring-boot-autoconfigure-1.5.21.RELEASE.jar%3Corg).[springframework](eclipse-javadoc:%E2%98%82=spring-boot-first-web-application/C:%5C/Users%5C/saraswati%5C/.m2%5C/repository%5C/org%5C/springframework%5C/boot%5C/spring-boot-autoconfigure%5C/1.5.21.RELEASE%5C/spring-boot-autoconfigure-1.5.21.RELEASE.jar%3Corg.springframework).[boot](eclipse-javadoc:%E2%98%82=spring-boot-first-web-application/C:%5C/Users%5C/saraswati%5C/.m2%5C/repository%5C/org%5C/springframework%5C/boot%5C/spring-boot-autoconfigure%5C/1.5.21.RELEASE%5C/spring-boot-autoconfigure-1.5.21.RELEASE.jar%3Corg.springframework.boot).[autoconfigure](eclipse-javadoc:%E2%98%82=spring-boot-first-web-application/C:%5C/Users%5C/saraswati%5C/.m2%5C/repository%5C/org%5C/springframework%5C/boot%5C/spring-boot-autoconfigure%5C/1.5.21.RELEASE%5C/spring-boot-autoconfigure-1.5.21.RELEASE.jar%3Corg.springframework.boot.autoconfigure).SpringBootApplication @[SpringBootConfiguration](eclipse-javadoc:%E2%98%82=spring-boot-first-web-application/C:%5C/Users%5C/saraswati%5C/.m2%5C/repository%5C/org%5C/springframework%5C/boot%5C/spring-boot%5C/1.5.21.RELEASE%5C/spring-boot-1.5.21.RELEASE.jar%3Corg.springframework.boot(SpringBootConfiguration.class%E2%98%83SpringBootConfiguration) @[EnableAutoConfiguration](eclipse-javadoc:%E2%98%82=spring-boot-first-web-application/C:%5C/Users%5C/saraswati%5C/.m2%5C/repository%5C/org%5C/springframework%5C/boot%5C/spring-boot-autoconfigure%5C/1.5.21.RELEASE%5C/spring-boot-autoconfigure-1.5.21.RELEASE.jar%3Corg.springframework.boot.autoconfigure(EnableAutoConfiguration.class%E2%98%83EnableAutoConfiguration) @[ComponentScan](eclipse-javadoc:%E2%98%82=spring-boot-first-web-application/C:%5C/Users%5C/saraswati%5C/.m2%5C/repository%5C/org%5C/springframework%5C/spring-context%5C/4.3.24.RELEASE%5C/spring-context-4.3.24.RELEASE.jar%3Corg.springframework.context.annotation(ComponentScan.class%E2%98%83ComponentScan)([excludeFilters](eclipse-javadoc:%E2%98%82=spring-boot-first-web-application/C:%5C/Users%5C/saraswati%5C/.m2%5C/repository%5C/org%5C/springframework%5C/spring-context%5C/4.3.24.RELEASE%5C/spring-context-4.3.24.RELEASE.jar%3Corg.springframework.context.annotation(ComponentScan.class%E2%98%83ComponentScan~excludeFilters)={@[Filter](eclipse-javadoc:%E2%98%82=spring-boot-first-web-application/C:%5C/Users%5C/saraswati%5C/.m2%5C/repository%5C/org%5C/springframework%5C/spring-context%5C/4.3.24.RELEASE%5C/spring-context-4.3.24.RELEASE.jar%3Corg.springframework.context.annotation(ComponentScan$Filter.class%E2%98%83Filter)([type](eclipse-javadoc:%E2%98%82=spring-boot-first-web-application/C:%5C/Users%5C/saraswati%5C/.m2%5C/repository%5C/org%5C/springframework%5C/spring-context%5C/4.3.24.RELEASE%5C/spring-context-4.3.24.RELEASE.jar%3Corg.springframework.context.annotation(ComponentScan$Filter.class%E2%98%83Filter~type)=[CUSTOM](eclipse-javadoc:%E2%98%82=spring-boot-first-web-application/C:%5C/Users%5C/saraswati%5C/.m2%5C/repository%5C/org%5C/springframework%5C/spring-context%5C/4.3.24.RELEASE%5C/spring-context-4.3.24.RELEASE.jar%3Corg.springframework.context.annotation(FilterType.class%E2%98%83FilterType%5ECUSTOM), [classes](eclipse-javadoc:%E2%98%82=spring-boot-first-web-application/C:%5C/Users%5C/saraswati%5C/.m2%5C/repository%5C/org%5C/springframework%5C/spring-context%5C/4.3.24.RELEASE%5C/spring-context-4.3.24.RELEASE.jar%3Corg.springframework.context.annotation(ComponentScan$Filter.class%E2%98%83Filter~classes)={[TypeExcludeFilter](eclipse-javadoc:%E2%98%82=spring-boot-first-web-application/C:%5C/Users%5C/saraswati%5C/.m2%5C/repository%5C/org%5C/springframework%5C/boot%5C/spring-boot%5C/1.5.21.RELEASE%5C/spring-boot-1.5.21.RELEASE.jar%3Corg.springframework.boot.context(TypeExcludeFilter.class%E2%98%83TypeExcludeFilter).class}), @[Filter](eclipse-javadoc:%E2%98%82=spring-boot-first-web-application/C:%5C/Users%5C/saraswati%5C/.m2%5C/repository%5C/org%5C/springframework%5C/spring-context%5C/4.3.24.RELEASE%5C/spring-context-4.3.24.RELEASE.jar%3Corg.springframework.context.annotation(ComponentScan$Filter.class%E2%98%83Filter)([type](eclipse-javadoc:%E2%98%82=spring-boot-first-web-application/C:%5C/Users%5C/saraswati%5C/.m2%5C/repository%5C/org%5C/springframework%5C/spring-context%5C/4.3.24.RELEASE%5C/spring-context-4.3.24.RELEASE.jar%3Corg.springframework.context.annotation(ComponentScan$Filter.class%E2%98%83Filter~type)=[CUSTOM](eclipse-javadoc:%E2%98%82=spring-boot-first-web-application/C:%5C/Users%5C/saraswati%5C/.m2%5C/repository%5C/org%5C/springframework%5C/spring-context%5C/4.3.24.RELEASE%5C/spring-context-4.3.24.RELEASE.jar%3Corg.springframework.context.annotation(FilterType.class%E2%98%83FilterType%5ECUSTOM), [classes](eclipse-javadoc:%E2%98%82=spring-boot-first-web-application/C:%5C/Users%5C/saraswati%5C/.m2%5C/repository%5C/org%5C/springframework%5C/spring-context%5C/4.3.24.RELEASE%5C/spring-context-4.3.24.RELEASE.jar%3Corg.springframework.context.annotation(ComponentScan$Filter.class%E2%98%83Filter~classes)={[AutoConfigurationExcludeFilter](eclipse-javadoc:%E2%98%82=spring-boot-first-web-application/C:%5C/Users%5C/saraswati%5C/.m2%5C/repository%5C/org%5C/springframework%5C/boot%5C/spring-boot-autoconfigure%5C/1.5.21.RELEASE%5C/spring-boot-autoconfigure-1.5.21.RELEASE.jar%3Corg.springframework.boot.autoconfigure(AutoConfigurationExcludeFilter.class%E2%98%83AutoConfigurationExcludeFilter).class})}) @[Target](eclipse-javadoc:%E2%98%82=spring-boot-first-web-application/C:%5C/Program%20Files%5C/Java%5C/jre1.8.0_211%5C/lib%5C/rt.jar%3Cjava.lang.annotation(Target.class%E2%98%83Target)([value](eclipse-javadoc:%E2%98%82=spring-boot-first-web-application/C:%5C/Program%20Files%5C/Java%5C/jre1.8.0_211%5C/lib%5C/rt.jar%3Cjava.lang.annotation(Target.class%E2%98%83Target~value)={[TYPE](eclipse-javadoc:%E2%98%82=spring-boot-first-web-application/C:%5C/Program%20Files%5C/Java%5C/jre1.8.0_211%5C/lib%5C/rt.jar%3Cjava.lang.annotation(ElementType.class%E2%98%83ElementType%5ETYPE)}) @[Retention](eclipse-javadoc:%E2%98%82=spring-boot-first-web-application/C:%5C/Program%20Files%5C/Java%5C/jre1.8.0_211%5C/lib%5C/rt.jar%3Cjava.lang.annotation(Retention.class%E2%98%83Retention)([value](eclipse-javadoc:%E2%98%82=spring-boot-first-web-application/C:%5C/Program%20Files%5C/Java%5C/jre1.8.0_211%5C/lib%5C/rt.jar%3Cjava.lang.annotation(Retention.class%E2%98%83Retention~value)=[RUNTIME](eclipse-javadoc:%E2%98%82=spring-boot-first-web-application/C:%5C/Program%20Files%5C/Java%5C/jre1.8.0_211%5C/lib%5C/rt.jar%3Cjava.lang.annotation(RetentionPolicy.class%E2%98%83RetentionPolicy%5ERUNTIME)) @[Documented](eclipse-javadoc:%E2%98%82=spring-boot-first-web-application/C:%5C/Program%20Files%5C/Java%5C/jre1.8.0_211%5C/lib%5C/rt.jar%3Cjava.lang.annotation(Documented.class%E2%98%83Documented) @[Inherited](eclipse-javadoc:%E2%98%82=spring-boot-first-web-application/C:%5C/Program%20Files%5C/Java%5C/jre1.8.0_211%5C/lib%5C/rt.jar%3Cjava.lang.annotation(Inherited.class%E2%98%83Inherited)  Indicates a [configuration](eclipse-javadoc:%E2%98%82=spring-boot-first-web-application/C:%5C/Users%5C/saraswati%5C/.m2%5C/repository%5C/org%5C/springframework%5C/boot%5C/spring-boot-autoconfigure%5C/1.5.21.RELEASE%5C/spring-boot-autoconfigure-1.5.21.RELEASE.jar%3Corg.springframework.boot.autoconfigure(SpringBootApplication.class%E2%98%83SpringBootApplication%E2%98%82Configuration) class that declares one or more [@Bean](eclipse-javadoc:%E2%98%82=spring-boot-first-web-application/C:%5C/Users%5C/saraswati%5C/.m2%5C/repository%5C/org%5C/springframework%5C/boot%5C/spring-boot-autoconfigure%5C/1.5.21.RELEASE%5C/spring-boot-autoconfigure-1.5.21.RELEASE.jar%3Corg.springframework.boot.autoconfigure(SpringBootApplication.class%E2%98%83SpringBootApplication%E2%98%82Bean) methods and also triggers [auto-configuration](eclipse-javadoc:%E2%98%82=spring-boot-first-web-application/C:%5C/Users%5C/saraswati%5C/.m2%5C/repository%5C/org%5C/springframework%5C/boot%5C/spring-boot-autoconfigure%5C/1.5.21.RELEASE%5C/spring-boot-autoconfigure-1.5.21.RELEASE.jar%3Corg.springframework.boot.autoconfigure(SpringBootApplication.class%E2%98%83SpringBootApplication%E2%98%82EnableAutoConfiguration) and [component scanning](eclipse-javadoc:%E2%98%82=spring-boot-first-web-application/C:%5C/Users%5C/saraswati%5C/.m2%5C/repository%5C/org%5C/springframework%5C/boot%5C/spring-boot-autoconfigure%5C/1.5.21.RELEASE%5C/spring-boot-autoconfigure-1.5.21.RELEASE.jar%3Corg.springframework.boot.autoconfigure(SpringBootApplication.class%E2%98%83SpringBootApplication%E2%98%82ComponentScan). This is a convenience annotation that is equivalent to declaring @Configuration, @EnableAutoConfiguration and @ComponentScan.  **Since:**1.2.0  **Author:**Phillip WebbStephane Nicoll |

To launch this spring boot application just right click and run as – java application which will start the embedded tomcat server and launch the spring boot application.

Now the third important file in this project is application.properties

1. **application.properties**

* This application.properties can be used as configuration file
* Let’s say if we want to run our application not default port 8080 but on 9000 then we can configure or change this port in this file (server.port=9000)

**So far we have learnt the following things**

* Creating maven project
* spring boot starter parent
* spring boot starter web
* @SpringBootApplication
* Auto configuration
* Devtools
* spring boot maven plugin

**Step01 -**

[**https://github.com/in28minutes/spring-boot-master-class/blob/master/02.Spring-Boot-Web-Application/Step01.md**](https://github.com/in28minutes/spring-boot-master-class/blob/master/02.Spring-Boot-Web-Application/Step01.md)

**What You Will Learn during this Step:**

* Lets create a simple web application using Spring Boot
* Lets run the Spring Boot Application
* There is a lot of magic happening in here! We will take a deep dive into the magic in Step 03.

**Files List**

pom.xml

src/main/java/com/in28minutes/springboot/web/SpringBootFirstWebApplication.java

src/main/resources/application.properties

src/test/java/com/in28minutes/springboot/web/SpringBootFirstWebApplicationTests.java

**todo.txt**

Spring Boot Starter Parent

Spring Boot Starter Web

@SpringBootApplication

Auto Configuration

**Step02-**

<https://github.com/in28minutes/spring-boot-master-class/blob/master/02.Spring-Boot-Web-Application/Step02.md>

## What You Will Learn during this Step:

* @RequestMapping(value = "/login", method = RequestMethod.GET)
* <http://localhost:8080/login>
* Why @ResponseBody?
* Important of RequestMapping method
* How do web applications work? Request and Response
* Browser sends Http Request to Web Server
* Code in Web Server => Input:HttpRequest, Output: HttpResponse
* Web Server responds with Http Response

## Useful Snippets and References

|  |
| --- |
| First Snippet  @**Controller**  public class LoginController {  @RequestMapping(value = "/login")  @ResponseBody  public String sayHello() {  return "Hello World dummy";  }  } |
| src/main/resources/application.properties logging.level.org.springframework.web: DEBUG |

## Now let’s use @RestController instead of using @Controller

### src/main/java/com/in28minutes/springboot/web/controller/LoginController.java

|  |
| --- |
| **package com.in28minutes.springboot.web.springbootfirstwebapplication.controller;**  **import org.springframework.stereotype.Controller;**  **import org.springframework.web.bind.annotation.RequestMapping;**  **import org.springframework.web.bind.annotation.ResponseBody;**  **import org.springframework.web.bind.annotation.RestController;**  **//@Controller**  **@RestController**  **publicclass LoginController {**    **@RequestMapping("/login")**  **//@ResponseBody**  **public String loginMessage() {**  **return"Hello: My first spring boot web app with RestController";**  **}**  **}** |

**Note**: We don’t need to use @ResponseBody annotation while using @RestController because @RestController is the combination of (@Controller + @ResponseBody)

Console Log: Here we have given correct URL but @ResponseBody is missing at method level: click enter or refresh page then: it first goes to @Controller Class > and method @RequestMapping to map URL >Goes inside the method and return string. Further it search for view resolver and it does not find any matching bean for view name because we have not mentioned @ResponseBody at method level.

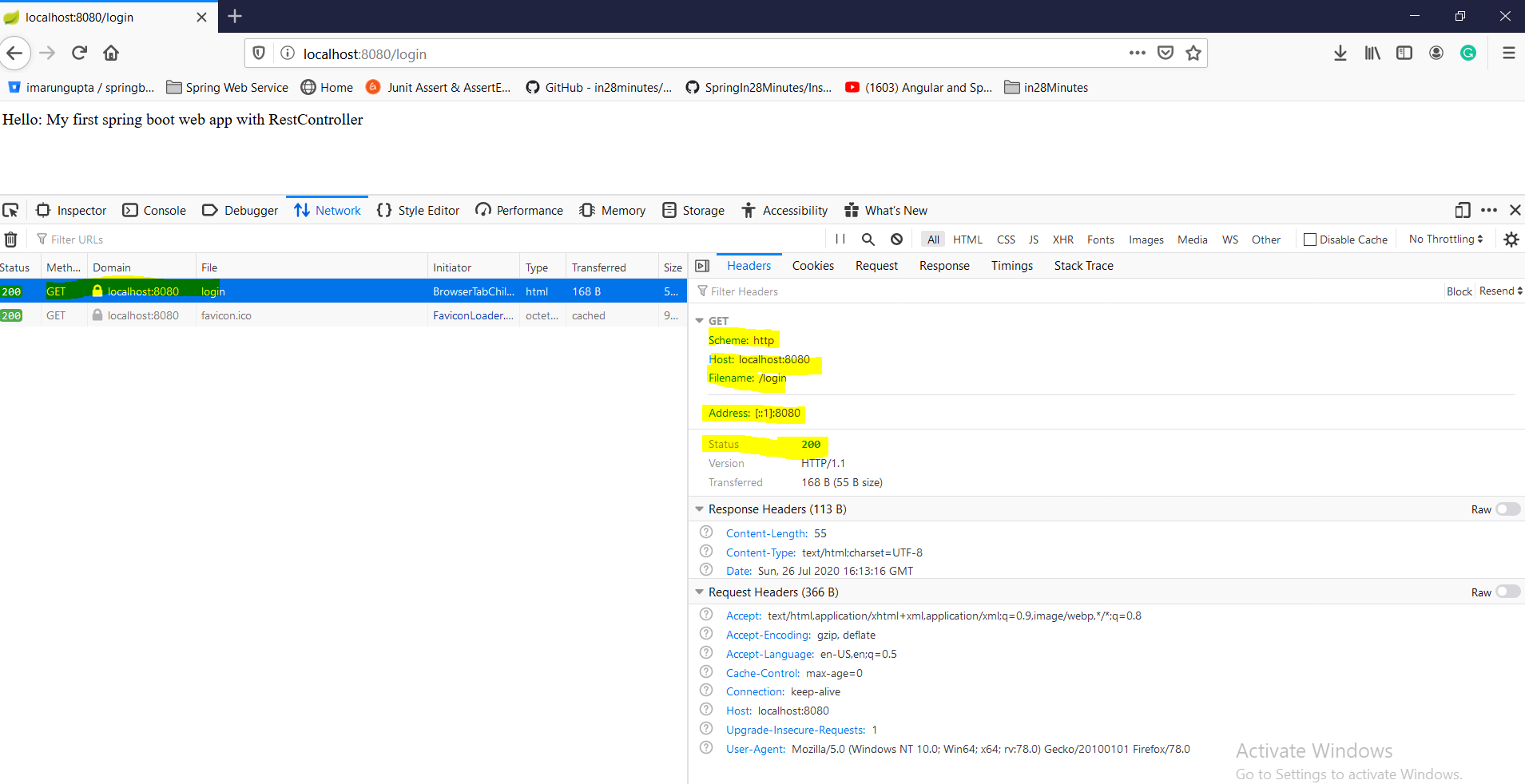
|  |
| --- |
| 2020-07-26 21:22:01.450 DEBUG 8448 --- [nio-9090-exec-1] o.s.web.servlet.DispatcherServlet : Initializing servlet 'dispatcherServlet'  2020-07-26 21:22:01.451 INFO 8448 --- [nio-9090-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring FrameworkServlet 'dispatcherServlet'  2020-07-26 21:22:01.451 INFO 8448 --- [nio-9090-exec-1] o.s.web.servlet.DispatcherServlet : FrameworkServlet 'dispatcherServlet': initialization started  2020-07-26 21:22:01.451 DEBUG 8448 --- [nio-9090-exec-1] o.s.web.servlet.DispatcherServlet : Using MultipartResolver [org.springframework.web.multipart.support.StandardServletMultipartResolver@2822282a]  2020-07-26 21:22:01.451 DEBUG 8448 --- [nio-9090-exec-1] o.s.web.servlet.DispatcherServlet : Unable to locate LocaleResolver with name 'localeResolver': using default [org.springframework.web.servlet.i18n.AcceptHeaderLocaleResolver@2c1da9f6]  2020-07-26 21:22:01.451 DEBUG 8448 --- [nio-9090-exec-1] o.s.web.servlet.DispatcherServlet : Unable to locate ThemeResolver with name 'themeResolver': using default [org.springframework.web.servlet.theme.FixedThemeResolver@691f2211]  2020-07-26 21:22:01.452 DEBUG 8448 --- [nio-9090-exec-1] o.s.web.servlet.DispatcherServlet : Unable to locate RequestToViewNameTranslator with name 'viewNameTranslator': using default [org.springframework.web.servlet.view.DefaultRequestToViewNameTranslator@61662c46]  2020-07-26 21:22:01.453 DEBUG 8448 --- [nio-9090-exec-1] o.s.web.servlet.DispatcherServlet : Unable to locate FlashMapManager with name 'flashMapManager': using default [org.springframework.web.servlet.support.SessionFlashMapManager@583c2b38]  2020-07-26 21:22:01.453 DEBUG 8448 --- [nio-9090-exec-1] o.s.web.servlet.DispatcherServlet : Published WebApplicationContext of servlet 'dispatcherServlet' as ServletContext attribute with name [org.springframework.web.servlet.FrameworkServlet.CONTEXT.dispatcherServlet]  2020-07-26 21:22:01.453 INFO 8448 --- [nio-9090-exec-1] o.s.web.servlet.DispatcherServlet : FrameworkServlet 'dispatcherServlet': initialization completed in 2 ms  2020-07-26 21:22:01.453 DEBUG 8448 --- [nio-9090-exec-1] o.s.web.servlet.DispatcherServlet : Servlet 'dispatcherServlet' configured successfully  2020-07-26 21:22:01.453 DEBUG 8448 --- [nio-9090-exec-1] o.s.web.servlet.DispatcherServlet : **DispatcherServlet with name 'dispatcherServlet' processing GET request for [/login]**  2020-07-26 21:22:01.454 DEBUG 8448 --- [nio-9090-exec-1] s.w.s.m.m.a.RequestMappingHandlerMapping : Looking up handler method for path /login  2020-07-26 21:22:01.454 DEBUG 8448 --- [nio-9090-exec-1] s.w.s.m.m.a.RequestMappingHandlerMapping : **Returning handler method [public java.lang.String com.in28minutes.springboot.web.springbootfirstwebapplication.controller.LoginController.loginMessage()]**  2020-07-26 21:22:01.454 DEBUG 8448 --- [nio-9090-exec-1] o.s.web.servlet.DispatcherServlet : Last-Modified value for [/login] is: -1  2020-07-26 21:22:01.457 DEBUG 8448 --- [nio-9090-exec-1] o.s.w.s.v.ContentNegotiatingViewResolver : Requested media types are [text/html, application/xhtml+xml, image/webp, application/xml;q=0.9, \*/\*;q=0.8] based on Accept header types and producible media types [\*/\*])  2020-07-26 21:22:01.457 DEBUG 8448 --- [nio-9090-exec-1] o.s.w.servlet.view.BeanNameViewResolver : **No matching bean found for view name 'Hello: My first spring boot web app with RestController'**  2020-07-26 21:22:01.460 DEBUG 8448 --- [nio-9090-exec-1] o.s.w.s.v.ContentNegotiatingViewResolver : **Returning [org.springframework.web.servlet.view.InternalResourceView: name 'Hello: My first spring boot web app with RestController'; URL [Hello: My first spring boot web app with RestController]] based on requested media type 'text/html'**  2020-07-26 21:22:01.460 DEBUG 8448 --- [nio-9090-exec-1] o.s.web.servlet.DispatcherServlet : **Rendering view [org.springframework.web.servlet.view.InternalResourceView: name 'Hello: My first spring boot web app with RestController'; URL [Hello: My first spring boot web app with RestController]] in DispatcherServlet with name 'dispatcherServlet'**  2020-07-26 21:22:01.462 DEBUG 8448 --- [nio-9090-exec-1] o.s.w.servlet.view.InternalResourceView : Forwarding to resource [Hello: My first spring boot web app with RestController] in InternalResourceView 'Hello: My first spring boot web app with RestController'  2020-07-26 21:22:01.466 DEBUG 8448 --- [nio-9090-exec-1] o.s.web.servlet.DispatcherServlet : **DispatcherServlet with name 'dispatcherServlet' processing GET request for [/Hello: My first spring boot web app with RestController]**  2020-07-26 21:22:01.466 DEBUG 8448 --- [nio-9090-exec-1] s.w.s.m.m.a.RequestMappingHandlerMapping : **Looking up handler method for path /Hello: My first spring boot web app with RestController**  2020-07-26 21:22:01.468 DEBUG 8448 --- [nio-9090-exec-1] s.w.s.m.m.a.RequestMappingHandlerMapping : **Did not find handler method for [/Hello: My first spring boot web app with RestController]**  2020-07-26 21:22:01.468 DEBUG 8448 --- [nio-9090-exec-1] o.s.w.s.handler.SimpleUrlHandlerMapping : Matching patterns for request [/Hello: My first spring boot web app with RestController] are [/\*\*]  2020-07-26 21:22:01.468 DEBUG 8448 --- [nio-9090-exec-1] o.s.w.s.handler.SimpleUrlHandlerMapping : URI Template variables for request [/Hello: My first spring boot web app with RestController] are {}  2020-07-26 21:22:01.468 DEBUG 8448 --- [nio-9090-exec-1] o.s.w.s.handler.SimpleUrlHandlerMapping : Mapping [/Hello: My first spring boot web app with RestController] to HandlerExecutionChain with handler [ResourceHttpRequestHandler [locations=[ServletContext resource [/], class path resource [META-INF/resources/], class path resource [resources/], class path resource [static/], class path resource [public/]], resolvers=[org.springframework.web.servlet.resource.PathResourceResolver@8208b4]]] and 1 interceptor  2020-07-26 21:22:01.469 DEBUG 8448 --- [nio-9090-exec-1] o.s.web.servlet.DispatcherServlet : Last-Modified value for [/Hello: My first spring boot web app with RestController] is: -1  2020-07-26 21:22:01.470 DEBUG 8448 --- [nio-9090-exec-1] o.s.web.servlet.DispatcherServlet : Null ModelAndView returned to DispatcherServlet with name 'dispatcherServlet': assuming HandlerAdapter completed request handling  2020-07-26 21:22:01.470 DEBUG 8448 --- [nio-9090-exec-1] o.s.web.servlet.DispatcherServlet : Successfully completed request  2020-07-26 21:22:01.471 DEBUG 8448 --- [nio-9090-exec-1] o.s.web.servlet.DispatcherServlet : Successfully completed request  2020-07-26 21:22:01.471 DEBUG 8448 --- [nio-9090-exec-1] o.s.web.servlet.DispatcherServlet : DispatcherServlet with name 'dispatcherServlet' processing GET request for [/error]  2020-07-26 21:22:01.471 DEBUG 8448 --- [nio-9090-exec-1] s.w.s.m.m.a.RequestMappingHandlerMapping : Looking up handler method for path /error  2020-07-26 21:22:01.473 DEBUG 8448 --- [nio-9090-exec-1] s.w.s.m.m.a.RequestMappingHandlerMapping : Returning handler method [public org.springframework.web.servlet.ModelAndView org.springframework.boot.autoconfigure.web.BasicErrorController.errorHtml(javax.servlet.http.HttpServletRequest,javax.servlet.http.HttpServletResponse)]  2020-07-26 21:22:01.473 DEBUG 8448 --- [nio-9090-exec-1] o.s.web.servlet.DispatcherServlet : Last-Modified value for [/error] is: -1  2020-07-26 21:22:01.483 DEBUG 8448 --- [nio-9090-exec-1] o.s.w.s.v.ContentNegotiatingViewResolver : Requested media types are [text/html, text/html;q=0.8] based on Accept header types and producible media types [text/html])  2020-07-26 21:22:01.485 DEBUG 8448 --- [nio-9090-exec-1] o.s.w.s.v.ContentNegotiatingViewResolver : Returning [org.springframework.boot.autoconfigure.web.ErrorMvcAutoConfiguration$SpelView@610f3009] based on requested media type 'text/html'  2020-07-26 21:22:01.485 DEBUG 8448 --- [nio-9090-exec-1] o.s.web.servlet.DispatcherServlet : Rendering view [org.springframework.boot.autoconfigure.web.ErrorMvcAutoConfiguration$SpelView@610f3009] in DispatcherServlet with name 'dispatcherServlet'  2020-07-26 21:22:01.566 DEBUG 8448 --- [nio-9090-exec-1] o.s.web.servlet.DispatcherServlet : Successfully completed request |

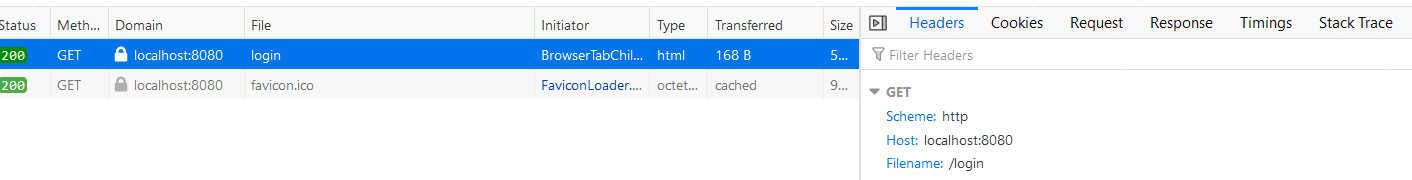
So here we will get whiteLevel error and will not get any response. Spring Boot provides this default error page.

But when we include @ResponseBody then we will get proper response: compare both the logs

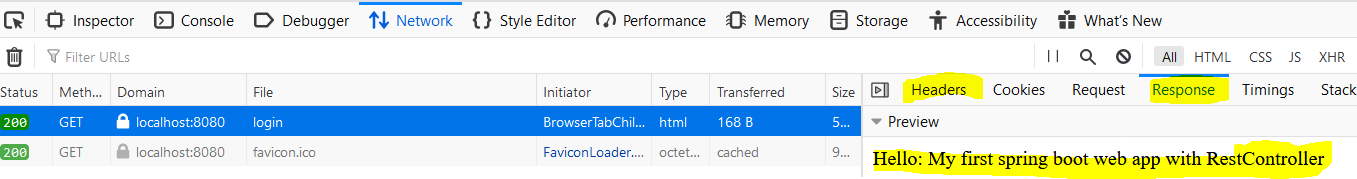
|  |
| --- |
| 2020-07-26 21:15:07.682 DEBUG 8448 --- [nio-9090-exec-1] o.s.web.servlet.DispatcherServlet : **Initializing servlet 'dispatcherServlet'**  2020-07-26 21:15:07.682 INFO 8448 --- [nio-9090-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring FrameworkServlet 'dispatcherServlet'  2020-07-26 21:15:07.682 INFO 8448 --- [nio-9090-exec-1] o.s.web.servlet.DispatcherServlet : FrameworkServlet 'dispatcherServlet': initialization started  2020-07-26 21:15:07.682 DEBUG 8448 --- [nio-9090-exec-1] o.s.web.servlet.DispatcherServlet : Using MultipartResolver [org.springframework.web.multipart.support.StandardServletMultipartResolver@340dec05]  2020-07-26 21:15:07.682 DEBUG 8448 --- [nio-9090-exec-1] o.s.web.servlet.DispatcherServlet : Unable to locate LocaleResolver with name 'localeResolver': using default [org.springframework.web.servlet.i18n.AcceptHeaderLocaleResolver@30327e5c]  2020-07-26 21:15:07.683 DEBUG 8448 --- [nio-9090-exec-1] o.s.web.servlet.DispatcherServlet : Unable to locate ThemeResolver with name 'themeResolver': using default [org.springframework.web.servlet.theme.FixedThemeResolver@7530e63e]  2020-07-26 21:15:07.684 DEBUG 8448 --- [nio-9090-exec-1] o.s.web.servlet.DispatcherServlet : Unable to locate RequestToViewNameTranslator with name 'viewNameTranslator': using default [org.springframework.web.servlet.view.DefaultRequestToViewNameTranslator@1f39eec1]  2020-07-26 21:15:07.685 DEBUG 8448 --- [nio-9090-exec-1] o.s.web.servlet.DispatcherServlet : Unable to locate FlashMapManager with name 'flashMapManager': using default [org.springframework.web.servlet.support.SessionFlashMapManager@448f54e7]  2020-07-26 21:15:07.685 DEBUG 8448 --- [nio-9090-exec-1] o.s.web.servlet.DispatcherServlet : Published WebApplicationContext of servlet 'dispatcherServlet' as ServletContext attribute with name [org.springframework.web.servlet.FrameworkServlet.CONTEXT.dispatcherServlet]  2020-07-26 21:15:07.685 INFO 8448 --- [nio-9090-exec-1] o.s.web.servlet.DispatcherServlet : FrameworkServlet 'dispatcherServlet': initialization completed in 3 ms  2020-07-26 21:15:07.685 DEBUG 8448 --- [nio-9090-exec-1] o.s.web.servlet.DispatcherServlet : Servlet 'dispatcherServlet' configured successfully  2020-07-26 21:15:07.685 DEBUG 8448 --- [nio-9090-exec-1] o.s.web.servlet.DispatcherServlet : **DispatcherServlet with name 'dispatcherServlet' processing GET request for [/login]**  2020-07-26 21:15:07.686 DEBUG 8448 --- [nio-9090-exec-1] s.w.s.m.m.a.RequestMappingHandlerMapping : **Looking up handler method for path /login**  2020-07-26 21:15:07.686 DEBUG 8448 --- [nio-9090-exec-1] s.w.s.m.m.a.RequestMappingHandlerMapping **: Returning handler method [public java.lang.String com.in28minutes.springboot.web.springbootfirstwebapplication.controller.LoginController.loginMessage()]**  2020-07-26 21:15:07.686 DEBUG 8448 --- [nio-9090-exec-1] o.s.web.servlet.DispatcherServlet : Last-Modified value for [/login] is: -1  2020-07-26 21:15:07.703 DEBUG 8448 --- [nio-9090-exec-1] m.m.a.RequestResponseBodyMethodProcessor : **Written [Hello: My first spring boot web app with RestController] as "text/html" using [org.springframework.http.converter.StringHttpMessageConverter@c4ee0e4]**  2020-07-26 21:15:07.703 DEBUG 8448 --- [nio-9090-exec-1] o.s.web.servlet.DispatcherServlet : Null ModelAndView returned to DispatcherServlet with name 'dispatcherServlet': assuming HandlerAdapter completed request handling  2020-07-26 21:15:07.703 DEBUG 8448 --- [nio-9090-exec-1] o.s.web.servlet.DispatcherServlet : Successfully completed request |

**What basically happens when we hit url(**[**http://localhost:8080/login**](http://localhost:8080/login) **),what happens in the background?**





When we hit the url : it goes the server <http://localhost:8080/login> and in the response section we get the response as shown below.



So here we have done following things.

1. Created controller class (LoginController) with annotation @Controller so that this class could be picked up as controller class.
2. Created a method of return type String and returned string “ Hello world”
3. Gave two annotation @RequestMapping for mapping request URL (/login) and @ResponseBody for binding the returned string to view resolver and displaying the same on to client browser.

**Step03-**

[**https://github.com/in28minutes/spring-boot-master-class/blob/master/02.Spring-Boot-Web-Application/Step03.md**](https://github.com/in28minutes/spring-boot-master-class/blob/master/02.Spring-Boot-Web-Application/Step03.md)

|  |
| --- |
| **What You Will Learn during this Step:**   * Demystifying some of the magic * Spring Boot Starter Parent * Spring Boot Starter Web * Embedded Tomcat * Dev Tools   **Spring Boot Starter Parent**: hover the mouse on pom.xml parent >Ctrl+click> it will take you to another parent pom.xml (it is like inheritance class A extends B) which will contain another spring boot parent(just like inheritance, Class A extends class B) [spring-boot-dependencies], default java version 1.6, Spring core, Maven plug-in for running the project, creating jar or war.  Again hover the mouse on its parent >ctrl+click> it will take us to another pom.xml where the versions of all the dependencies (jars) is mentioned.  In this way spring-boot-starter-parent manages the dependencies and its versions.  **Spring Boot Starter Web: Basically it does two things**   1. **All the dependencies that we need to run the web and rest application (like spring core, spring mvc, validation, logging, embedded tomcat, Jackson data binding etc.) To get more go to pom.xml>hover the mouse on spring-boot-starter-web> ctrl+click and here we will see all the things under this. Tomcat, spring web, MVC dependencies are mentioned inside Spring Boot Starter Web dependency only.** 2. **Embedded Tomcat server, which is now the part of application. Like earlier Tomcat server were sitting outside where we were creating war file and then pushing into tomcat server. That is now we are not doing here in spring boot. So here we are including tomcat in application that is why it is called embedded tomcat. So the tomcat server is now the part of application, which allows us to do the configure in other server e.g. if we need to deploy this application jar on Linux server then we don’t need configure tomcat server separately because tomcat server is already embedded in the application.**   **Dev Tools:** Provides fast application restarts, Live Reload, and configurations for enhanced development experience. This devtools makes the development faster and make the developer productive. Basically the dev tool keeps monitoring the application folders and as soon as we do any changes in any of the project file, it reloaded the application into the tomcat server, and we don’t need to restart tomcat server explicitly. Generally the dynamic part like java file will be reloaded again and again if we do any changes.  This tool will be only for dynamic part like java but if we have made any changes in pom.xml (added any new dependency) then in this case we need to restart the server. This very important things that SpringBoot brings up. |

## ****Step04- Redirect to Login JSP -@ResponseBody and View Resolver.****

## <https://github.com/in28minutes/spring-boot-master-class/blob/master/02.Spring-Boot-Web-Application/Step04.md>

|  |
| --- |
| Untill now we had only controller, now we will use JSP and for that we will write view (jsp). Three things we need to add into so far existing project mentioned in yellow colour. **What You Will Learn during this Step:**   * Your First JSP( for view resolver and for viewing the response) * Application.properties : where MVC prefix and suffix will be defined for view resolving) * There is a bit of setup before we get there! * Introduction to View Resolver   **Useful Snippets and References**  First Snippet - /src/main/webapp/WEB-INF/jsp/login.jsp  <html>  <head>  <title>Yahoo!!</title>  </head>  <body>  My First JSP!!!  </body>  </html>  Second Snippet - /src/main/resources/application.properties  spring.mvc.view.prefix: /WEB-INF/jsp/  spring.mvc.view.suffix: .jsp  logging.level.: DEBUG  Third Snippet : To enable jsp support in embedded tomcat server!  <dependency>  <groupId>org.apache.tomcat.embed</groupId>  <artifactId>tomcat-embed-jasper</artifactId>  <scope>provided</scope>  </dependency>  **Exercises**   * Create a new jsp and a new controller method to redirect to it! * Play around!   **Files List**  **pom.xml**  <?xml version="1.0" encoding="UTF-8"?>  <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">  <modelVersion>4.0.0</modelVersion>  <groupId>com.in28minutes.springboot.web</groupId>  <artifactId>spring-boot-first-web-application</artifactId>  <version>0.0.1-SNAPSHOT</version>  <packaging>jar</packaging>  <name>spring-boot-first-web-application</name>  <description>Demo project for Spring Boot</description>  <parent>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-parent</artifactId>  <version>1.4.3.RELEASE</version>  <relativePath/><!-- lookup parent from repository -->  </parent>  <properties>  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>  <java.version>1.8</java.version>  </properties>  <dependencies>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-web</artifactId>  </dependency>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-devtools</artifactId>  <scope>runtime</scope>  </dependency>  <dependency>  <groupId>org.apache.tomcat.embed</groupId>  <artifactId>tomcat-embed-jasper</artifactId>  <scope>provided</scope>  </dependency>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-test</artifactId>  <scope>test</scope>  </dependency>  </dependencies>  <build>  <plugins>  <plugin>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-maven-plugin</artifactId>  </plugin>  </plugins>  </build>  </project>  **src/main/java/com/in28minutes/springboot/web/controller/LoginController.java**  package com.in28minutes.springboot.web.controller;  import org.springframework.stereotype.Controller;  import org.springframework.web.bind.annotation.RequestMapping;  import org.springframework.web.bind.annotation.ResponseBody;  @Controller  public class LoginController {    @RequestMapping("/login")  //@ResponseBody  public String loginMessage(){  return "login";  }  }  **Note: @ResponseBody annotation made commented because it used to print method response into browser. But here we are printing response from view (jsp) that is why we need not to use this annotation.**  **src/main/java/com/in28minutes/springboot/web/SpringBootFirstWebApplication.java**  package com.in28minutes.springboot.web;  import org.springframework.boot.SpringApplication;  import org.springframework.boot.autoconfigure.SpringBootApplication;  @SpringBootApplication  public class SpringBootFirstWebApplication {  public static void main(String[] args) {  SpringApplication.run(SpringBootFirstWebApplication.class, args);  }  }  **src/main/resources/application.properties**  spring.mvc.view.prefix=/WEB-INF/jsp/  spring.mvc.view.suffix=.jsp  logging.level.org.springframework.web=DEBUG  **src/main/webapp/WEB-INF/jsp/login.jsp**  <html>  <head>  <title>First Web Application</title>  </head>  <body>  My First JSP!!  </body>  </html>  **src/test/java/com/in28minutes/springboot/web/SpringBootFirstWebApplicationTests.java**  package com.in28minutes.springboot.web;  import org.junit.Test;  import org.junit.runner.RunWith;  import org.springframework.boot.test.context.SpringBootTest;  import org.springframework.test.context.junit4.SpringRunner;  @RunWith(SpringRunner.class)  @SpringBootTest  public class SpringBootFirstWebApplicationTests {  @Test  public void contextLoads() {  }  }  **todo.txt**  Spring Boot Starter Parent  Spring Boot Starter Web  @SpringBootApplication  Auto Configuration  Dispatcher Servlet  /login => "login"  "login" => src/main/webapp/WEB-INF/jsp/login.jsp  Search for a view named "login"  /login => LoginController |

So here we have done following things.

1. Created controller class (LoginController) with annotation @Controller so that this class could be picked up as controller class.
2. Created a method of return type String and returned string “ login” (jsp page name)
3. Created a jsp page inside WEB-INF with the name of login.
4. Gave two annotation @RequestMapping for mapping request URL (/login) and @ResponseBody for binding the returned string to view resolver and displaying the same on to client browser.
5. Created view prefix and view suffix
6. Added on dependency in POM.xml just to enable JSP support in embedded tomcat server.

## ****Step05-** Show userid and password on welcome page –ModelMap & @RequestParameter**

## <https://github.com/in28minutes/spring-boot-master-class/blob/master/02.Spring-Boot-Web-Application/Step05.md>

|  |
| --- |
| **What You Will Learn during this Step:**   * Here we will create first GET Parameter and will see how we will pass this parameter to JSP with request URL (<http://localhost:8080/login?name=Arun>). * This parameter what we are passing with URL is called get parameter and will see how will mention in the controller class and method level. * Problem with using GET   **Snippets**  ModelMap model  model.put("name", name);  My First JSP!!! My name is ${name}  **Files List**  **pom.xml: Same as just above step-4**  **src/main/java/com/in28minutes/springboot/web/controller/LoginController.java**  package com.in28minutes.springboot.web.controller;  import org.springframework.stereotype.Controller;  import org.springframework.ui.ModelMap;  import org.springframework.web.bind.annotation.RequestMapping;  import org.springframework.web.bind.annotation.RequestParam;  @Controller  public class LoginController {    //Model  @RequestMapping("/login")  public String loginMessage(@RequestParam String name, **ModelMap** model){  model.put("namekey", name);  return "login";  }  }  **Note: Model is used to pass data from controller to view(jsp)The same name we will have in model class as well as in jsp(where this value need to display)**  **@Controller : is stereotype**  **@RequestMapping, @RequestParam : web binding annoataion**  **src/main/java/com/in28minutes/springboot/web/SpringBootFirstWebApplication.java**  Launcher class No change exactly same as defined above:  **src/main/resources/application.properties**  spring.mvc.view.prefix=/WEB-INF/jsp/  spring.mvc.view.suffix=.jsp  logging.level.org.springframework.web=DEBUG  **src/main/webapp/WEB-INF/jsp/login.jsp**  <html>  <head>  <title>First Web Application</title>  </head>  <body>  My First JSP!! Welcome ${namekey}!  </body>  </html>  src/test/java/com/in28minutes/springboot/web/SpringBootFirstWebApplicationTests.java  No change in this class same as above in other example  **todo.txt**  Spring Boot Starter Parent  Spring Boot Starter Web  @SpringBootApplication  Auto Configuration  Dispatcher Servlet  /login => "login"  "login" => src/main/webapp/WEB-INF/jsp/login.jsp  Search for a view named "login"  /login => LoginController |

## So as here we can see that if we want to send some data from the controller to the view, in that case we will use model to map controller data to the view (jsp). So controller control the entire flow and once the controller has some data, then it put into the model and redirects to the view and view used to model to render the data on the screen. That is why it called MVC (Model-View-Controller)

|  |
| --- |
| public String loginMessage(@RequestParam String name, **ModelMap** model){  model.put("namekey", name);  return "login"; Here we have created name as string request parameter using @RequestParam annotation and putting this request parameter value into model and using model put method we are putting this value into model key and now this model will be available into view hence finally using this key we print the request parameter value into jsp using expression language${ }.My First JSP!!! welcome:${namekey} |

## 

## 

## So far:

|  |  |
| --- | --- |
| Spring Boot Starter ParentSpring Boot Starter Webtomcat-embed-jasperspring-boot-devtoolsspring-boot-starter-testspring-boot-maven-plugin@SpringBootApplicationAuto Configuration | @Controller@RequestMapping("/mylogin")RequestParam String name,ModelMap model@RunWith(SpringRunner.class)@SpringBootTest |

## ****Step06- DispatcherServlet and Spring MVC flow.****

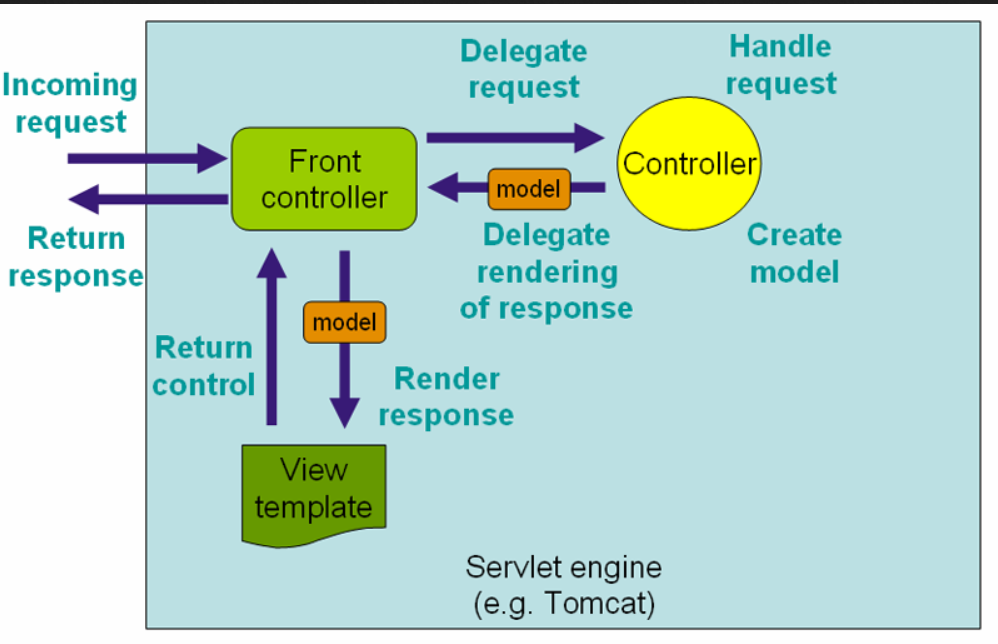
## <https://github.com/in28minutes/spring-boot-master-class/blob/master/02.Spring-Boot-Web-Application/Step06.md>

## What You Will Learn during this Step:

* Understand importance of DispatcherServlet.

## Spring MVC Request Flow

* DispatcherServlet is nothing but front controller which first receives all the request (whether it is login or error or anything) comes from web application. i.e. DispatcherServlet receives HTTP Request.
* After that it search that what are the mapping(like /login) available and/or param(like- ?name=arun) in request comes in the form of URL
* DispatcherServlet identifies the right Controller (LoginController) based on the URL (/login).
* Controller executes Business Logic.
* Controller returns a) Model b) View Name (jsp page name) Back to DispatcherServlet.
* DispatcherServlet identifies the correct view (ViewResolver- i.e. logical jsp name).
* DispatcherServlet makes the model available to view (jsp) and executes it using prefix and suffix which is configured in the application.properties file.
* DispatcherServlet returns HTTP Response Back.
* Flow : <http://docs.spring.io/spring-framework/docs/2.0.8/reference/images/mvc.png>

****

In POM.xml: We have four main tags

1- Parent tags

2- Properties tags

3- dependencies tags

4- maven build tag

## ****Step07-Create Html form to get request.****

## <https://github.com/in28minutes/spring-boot-master-class/blob/master/02.Spring-Boot-Web-Application/Step07.md>

|  |
| --- |
| **What You Will Learn during this Step:**  So far we have passed the get request parameter with the URL but the problem with this method is here everyone can see the request because this request parameter goes with URL. Let’s get the name from the user (request url using @RequestParam) in a form. Here instead of passing parameter in the URL will pass the request from the HTML form and instead of using Get method will use Post method for sending request to the server. So here we will create two methods.   * 1. One method (showLoginPage) to render login pate with URL- Pattern value=”/login” and RequestMethod is Get. And here we don’t need @RequestParam as here we are not passing any parameter to server. Only we are directing view (jsp page)   2. Another method (handleLogin) which will pass the request to server and put into Model objects and directing view along with model. Here, we are using POST method to send request parameter to server because Get method is not secure as the parameter value which will send to server will be displayed in the request URL. Here we are using @RequestParam to get the request parameter from the page.   First Snippet  @RequestMapping(value = "/login", method = RequestMethod.GET)  public String showLoginPage(ModelMap model, @RequestParam String name) {  return "login";  }  @RequestMapping(value = "/login", method = RequestMethod.POST)  public String handleLogin(ModelMap model, @RequestParam String name) {  model.put("name", name);  return "welcome";  }  Second Snippet  <form method="POST">  Name : <input name="name" type="text" /><input type="submit" />  </form>  **Files List**  pom.xml: Same what we have used in the above example. No change.  src/main/java/com/in28minutes/springboot/web/controller/LoginController.java  src/main/java/com/in28minutes/springboot/web/SpringBootFirstWebApplication.java  src/main/resources/application.properties  src/main/webapp/WEB-INF/jsp/login.jsp  src/main/webapp/WEB-INF/jsp/welcome.jsp  src/test/java/com/in28minutes/springboot/web/SpringBootFirstWebApplicationTests.java  **src/main/java/com/in28minutes/springboot/web/controller/LoginController.java**  package com.in28minutes.springboot.web.controller;  import org.springframework.stereotype.Controller;  import org.springframework.ui.ModelMap;  import org.springframework.web.bind.annotation.RequestMapping;  import org.springframework.web.bind.annotation.RequestMethod;  import org.springframework.web.bind.annotation.RequestParam;  @Controller  public class LoginController {    @RequestMapping(value="/login", method = RequestMethod.GET)  public String showLoginPage(ModelMap model){  return "login";  }  // This method just redirecting login page  @RequestMapping(value="/login", method = RequestMethod.POST)  public String showWelcomePage(ModelMap model, @RequestParam String name){  model.put("name", name);  return "welcome";  }  }  **src/main/java/com/in28minutes/springboot/web/SpringBootFirstWebApplication.java**  No change, will same as in the other example SpringBoot launcher class.  **src/main/resources/application.properties**  spring.mvc.view.prefix=/WEB-INF/jsp/  spring.mvc.view.suffix=.jsp  logging.level.org.springframework.web=DEBUG  **src/main/webapp/WEB-INF/jsp/login.jsp**  <html>  <head>  <title>First Web Application</title>  </head>  <body>  <form method="post">  Name : <input type="text" name="name" />  Password : <input type="password" name="password" />  <input type="submit" />  </form>  </body>  </html>  **Note: Created a form and input tag to accept request parameter**  **src/main/webapp/WEB-INF/jsp/welcome.jsp**  <html>  <head>  <title>First Web Application</title>  </head>  <body>  Welcome ${name}!!  </body>  </html>  **src/test/java/com/in28minutes/springboot/web/SpringBootFirstWebApplicationTests.java**  No change, same as others above example.  **todo.txt**  Spring Boot Starter Parent  Spring Boot Starter Web  @SpringBootApplication  Auto Configuration  Dispatcher Servlet  /login => "login"  "login" => src/main/webapp/WEB-INF/jsp/login.jsp  Search for a view named "login"  /login => LoginController |

## So far we have learnt how to send request parameter through the URL and learnt how to print the same on JSP. Now we will learn an alternative way to send request parameter through form.

## Get method is not secure because whatever we will type in the text box it will show in the request URL. Let’s say if we are passing name=Ranga and password= dummy and click on submit then in the request URL we will get like as:

## http://localhost:8080/login?name=Ranga&password=dummy

## 

## While hit this URL then there is a number of router between browser and server which trough this URL goes and some can read and inject other things.

## Now when we hit <http://localhost:8080/getLogin>

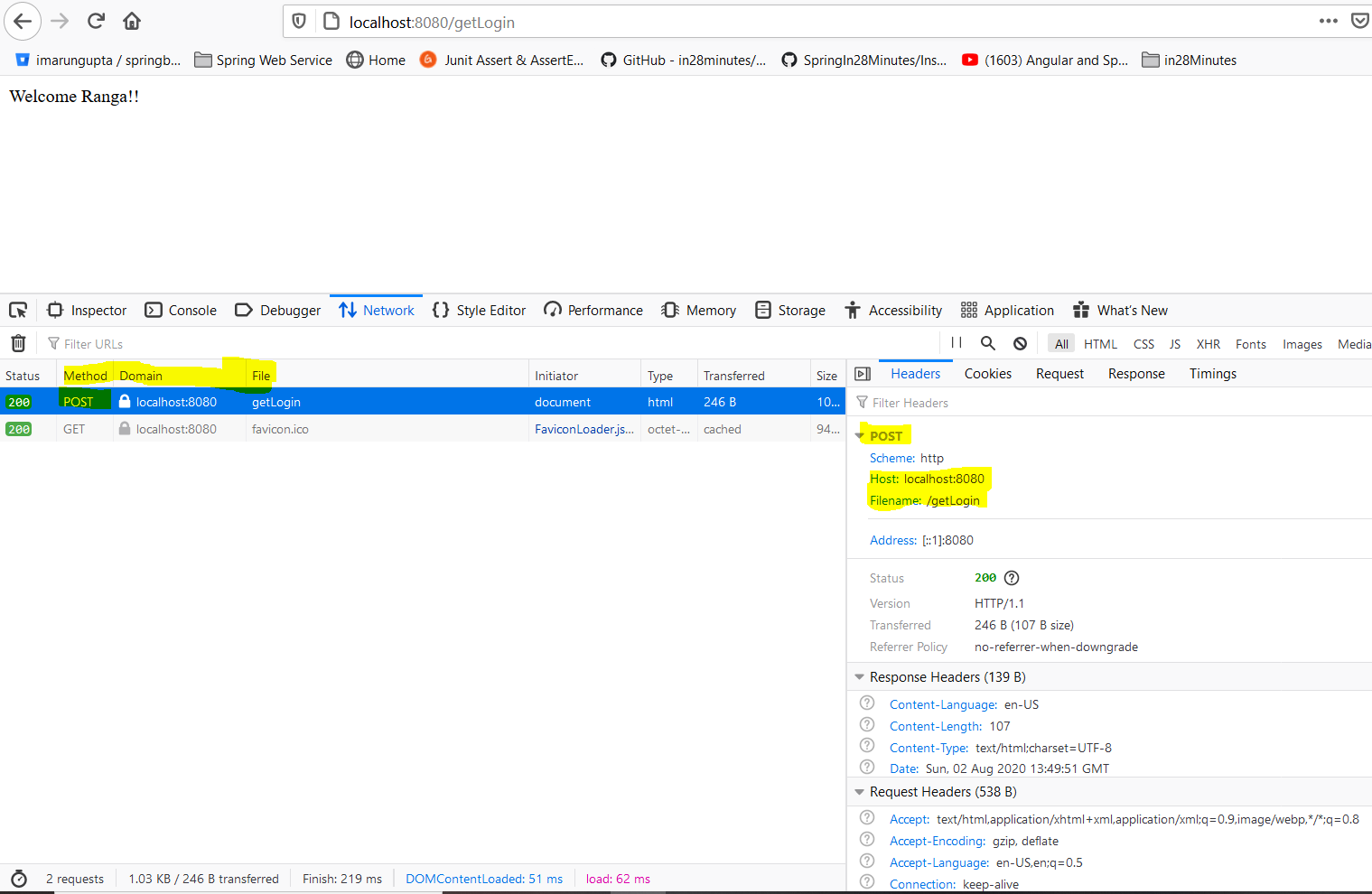
## 

## In the above screen Method is Get and in the header section we see detail.

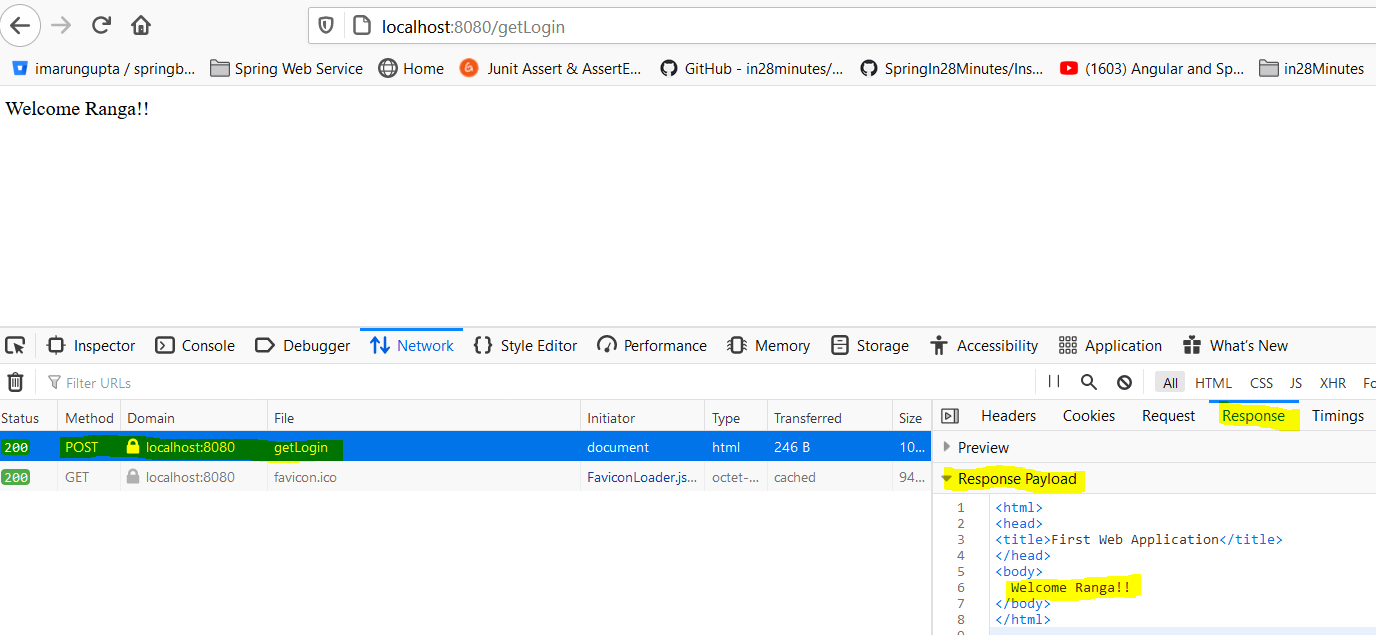
## 

In the Response tab we get response payload (html) and preview.

Now let enter name and password and click on Submit Query.



And in Response tab: Here method is POST



**But post method is secure because it does not send the request parameter with the Request URL. So make sure that when we are sending any parameter to the server, use only post method.**

**So as we have seen that we have two methods in the LoginController class one method is having Get request method and another method is having POST request method. So once we hit the URL then first method having Get request method gets called because Get is default method and its gets called automatically once hit the request URL. After that we get login page. Now in the login form the method type is POST so once we click on submit button the handler start matching the mapping URL (/getLogin) and method POST and hence it called another method annotated with POST method. This is the logic behind of matching and calling particular method based on the URL mapped and request method.**

## ****Step08-Add hard coded validation of userid and password.****

[**https://github.com/in28minutes/spring-boot-master-class/blob/master/02.Spring-Boot-Web-Application/Step08.md**](https://github.com/in28minutes/spring-boot-master-class/blob/master/02.Spring-Boot-Web-Application/Step08.md)

|  |
| --- |
| **What You Will Learn during this Step:**   * Add validation for userid and password * Hard coded validation!! * Here we will create service class / component class(LoginService) of Boolean type which will return true if userid and password gets matched and return false if userid or password does not match. * Create a handleLogin method and Autowiered LoginService just to validate userid and password.Redirect to login page if validation fails: or redirect to Welcome page if validation pass.   **First Snippet**  package com.in28minutes.springboot.web.service;  import org.springframework.stereotype.Component;  @Component  public class LoginService {  public boolean validateUser(String user, String password) {  return user.equalsIgnoreCase("in28Minutes") && password.equals("dummy");  }  }  **Second Snippet**  @Autowired  private LoginService service;  @RequestMapping(value = "/login", method = RequestMethod.POST)  public String handleLogin(ModelMap model, @RequestParam String name,  @RequestParam String password) {  boolean isValidUser = service.validateUser(name, password);  if (isValidUser) {  model.put("name", name);  return "welcome";  } else {  model.put("errorMessage", "Invalid Credentials!!");  return "login";  }  }  **Files List**  pom.xml: Same dependencies-No change  In POM.xml: We have four main tags  1- Parent tag- spring-boot-starter-parent  2- **Properties**- java versiontags  3 -Dependencies-a- spring-boot-starter-web  tomcat-embed-jasper  spring-boot-devtools  spring-boot-starter-test  4-build maven plugin- spring-boot-maven-plugin  File – Java , properties and jsp   * src/main/java/com/in28minutes/springboot/web/controller/LoginController.java * src/main/java/com/in28minutes/springboot/web/service/LoginService.java * src/main/java/com/in28minutes/springboot/web/SpringBootFirstWebApplication.java * src/main/resources/application.properties * src/main/webapp/WEB-INF/jsp/login.jsp * src/main/webapp/WEB-INF/jsp/welcome.jsp * src/test/java/com/in28minutes/springboot/web/SpringBootFirstWebApplicationTests.java   **src/main/java/com/in28minutes/springboot/web/controller/LoginController.java**  package com.in28minutes.springboot.web.controller;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.stereotype.Controller;  import org.springframework.ui.ModelMap;  import org.springframework.web.bind.annotation.RequestMapping;  import org.springframework.web.bind.annotation.RequestMethod;  import org.springframework.web.bind.annotation.RequestParam;  import com.in28minutes.springboot.web.service.LoginService;  @Controller  public class LoginController {    @Autowired  LoginService service;    @RequestMapping(value="/login", method = RequestMethod.GET)  public String showLoginPage(ModelMap model){  return "login";  }    @RequestMapping(value="/login", method = RequestMethod.POST)  public String showWelcomePage(ModelMap model, @RequestParam String name,  @RequestParam String password){  boolean isValidUser = service.validateUser(name, password);    if (!isValidUser) {  model.put("errorMessage", "Invalid Credentials");  return "login";  }    model.put("name", name);  model.put("password", password);    return "welcome";  }  }  **src/main/java/com/in28minutes/springboot/web/service/LoginService.java**  package com.in28minutes.springboot.web.service;  import org.springframework.stereotype.Component;  **@Component**  public class LoginService {  public boolean validateUser(String userid, String password) {  // in28minutes, dummy  return userid.equalsIgnoreCase("in28minutes")  && password.equalsIgnoreCase("dummy");  }  }  **src/main/java/com/in28minutes/springboot/web/SpringBootFirstWebApplication.java**  No change in this class  **src/main/resources/application.properties**  spring.mvc.view.prefix=/WEB-INF/jsp/  spring.mvc.view.suffix=.jsp  logging.level.org.springframework.web=DEBUG  **src/main/webapp/WEB-INF/jsp/login.jsp**  <html>  <head>  <title>First Web Application</title>  </head>  <body>  <font color="red">${errorMessage}</font>  <form method="post">  Name : <input type="text" name="name" />  Password : <input type="password" name="password" />  <input type="submit" />  </form>  </body>  </html>  **src/main/webapp/WEB-INF/jsp/welcome.jsp**  <html>  <head>  <title>First Web Application</title>  </head>  <body>  Welcome ${name}!!  </body>  </html>  **src/test/java/com/in28minutes/springboot/web/SpringBootFirstWebApplicationTests.java**  No change in test class  **todo.txt**  Component, Service, Repository, Controller  Autowired  ComponentScan  Spring Boot Starter Parent  Spring Boot Starter Web  @SpringBootApplication  Auto Configuration  Dispatcher Servlet  /login => "login"  "login" => src/main/webapp/WEB-INF/jsp/login.jsp  Search for a view named "login"  /login => LoginController |

## ****Step09-Magic of spring (****@Autowired, @Component, @Repository, @Controller, @ComponentScan

[**https://github.com/in28minutes/spring-boot-master-class/blob/master/02.Spring-Boot-Web-Application/Step09.md**](https://github.com/in28minutes/spring-boot-master-class/blob/master/02.Spring-Boot-Web-Application/Step09.md)

**What You Will Learn during this Step:**

* Magic of Spring
* Learn about Spring Auto-wiring and Dependency Management.
* @Autowired, @Component, @Repository, @Controller, @ComponentScan

In the last step we have create one component service and we Autowired it into LoginController. Let’s learn some more about auotwiring and dependency injection. Spring is one of most famous dependency injection framework. Before dependency injection it was very difficult to do unit testing. The only way was that we had to deploy the code into server and then test it. Hence everything was tightly coupled (like earlier we had to write fully [ClassName obj = new ClassName();]. So make it loosely coupled spring brought dependency injection which makes it very easy to do unit testing.

If we mention @Component annotation at any class then it means we are requesting spring framework to manage its object initiation and creation. Similarly we have another annotation like [@Service, @Repository @Controller etc.] if we mention these annotation at class level then spring will take care of its object creation and initiation. @ Controller is a specialization of @Component annotation allowing for implementation classes to be autodetected through classpath scanning i.e. @Controller is typically used specially in case of MVC but @Component @Service, @Repository is being used mostly everywhere.

@Service can be used for business service. @Repository can be used in terms of data store or database operations. @Component can be used for general purpose for reusability just like we have created one component class to validate userid and password. This component can be reused in many places to validate. So these four annotations are being managed by spring itself and the class bean(object) annotated with one of these annotation are managed by spring itself.

**Now let’s talk about @Autowired:** It is generally used inside the class [like inside controller class] so once the bean is created, spring will Autowired automatically wherever it is needed [like inside the controller class]

**@ComponentScan**: Generally **@SpringBootApplication** annotated class Indicates a [configuration](eclipse-javadoc:%E2%98%82=first-springboot-projecct/C:%5C/Users%5C/saraswati%5C/.m2%5C/repository%5C/org%5C/springframework%5C/boot%5C/spring-boot-autoconfigure%5C/1.4.0.RELEASE%5C/spring-boot-autoconfigure-1.4.0.RELEASE.jar%3Corg.springframework.boot.autoconfigure(SpringBootApplication.class%E2%98%83SpringBootApplication%E2%98%82Configuration) class which triggers [auto-configuration](eclipse-javadoc:%E2%98%82=first-springboot-projecct/C:%5C/Users%5C/saraswati%5C/.m2%5C/repository%5C/org%5C/springframework%5C/boot%5C/spring-boot-autoconfigure%5C/1.4.0.RELEASE%5C/spring-boot-autoconfigure-1.4.0.RELEASE.jar%3Corg.springframework.boot.autoconfigure(SpringBootApplication.class%E2%98%83SpringBootApplication%E2%98%82EnableAutoConfiguration) and [component scanning](eclipse-javadoc:%E2%98%82=first-springboot-projecct/C:%5C/Users%5C/saraswati%5C/.m2%5C/repository%5C/org%5C/springframework%5C/boot%5C/spring-boot-autoconfigure%5C/1.4.0.RELEASE%5C/spring-boot-autoconfigure-1.4.0.RELEASE.jar%3Corg.springframework.boot.autoconfigure(SpringBootApplication.class%E2%98%83SpringBootApplication%E2%98%82ComponentScan). This **@SpringBootApplication** annotation consists of @Configuration, @EnableAutoConfiguration and @ComponentScan. Generally scanning happens in the same package defined for @SpringBootApplication class for all the @Component, @Service, @Repository, and @Controller. But if want to scan these annotation in another different package then we have to explicitly mention the package name to be scan. If we hover the mouse on @SpringBootApplication we will found that @ComponentScan comes under @SpringBootApplication and by default it does the scanning into main package and its sub packages.

|  |
| --- |
| package **com.in28minutes.springboot.web**;  import org.springframework.boot.SpringApplication;  import org.springframework.boot.autoconfigure.SpringBootApplication;  @SpringBootApplication  public class SpringBootFirstFormWebApplication {  public static void main(String[] args) {  SpringApplication.run(SpringBootFirstFormWebApplication.class, args);  }  }  So here the default scanning will be done for **com.in28minutes.springboot.web** So the @Component will be scanned and initialized for auotwiring if it is available in the com.in28minutes.springboot.web package.  But if @Component is available in some other package then we need to explicitly mention to be scanned.  The best way is we always mention base package so that the entire package associated with it will be the part of this and will be available for auto scan.  So to make available auto scan in the entire package we will mention the packaging like given below.  @ComponentScan (**com.in28minutes**). This declaration will scan this package as well as its associated packages. |

This is the main basics of spring.

**Step-10: Todo Management Application(add,delete& update todos)**

* 1. **Create a LoginController and login.jsp. Once logged in successfully redirect the page to welcome.jsp having link to display todos list.**
  2. **Create TodoController.java and list-todos.jsp**
  3. **Create a model class with name Todo.java having variable (id,user,desc,targetDate,isDone)**
  4. **Make a TodoService.java using @Service and create static list of Todo.java variable and inject it into controller class.**
  5. **Create retriveTodos (String user) in TodoService.java to display Todo.**
  6. **Create addTodo (id, user,desc ,targetDate,isDone) inside the TodoService.java. Here we will use in-memory arraylist to add the records**
  7. **Create deleteTodo(int id) inside TodoService.java**
  8. **Autowired the TodoService.java inside the TodoController.java and call all the above method.**

**What we will do: inshort**

* Create TodoController and list-todos.jsp
* Make TodoService a @Service and inject it.
* We will perform add, update and delete operation.
* For performing above operation we need to have some business logic. So to perform business logic we will use in memory data base like array list.
* We will include these operation in TodoService.java file

**Files List**

pom.xml: Same dependencies-No change

In POM.xml: We have four main tags

1- Parent tag- spring-boot-starter-parent

2- **Properties**- java versiontags

3 -Dependencies-a- spring-boot-starter-web

tomcat-embed-jasper

spring-boot-devtools

spring-boot-starter-test

4-build maven plugin- spring-boot-maven-plugin

File – Java, properties and jsp

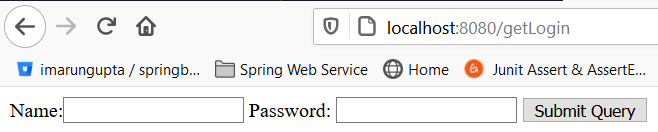
* 1. /src/main/java/com/in28minutes/springboot/web/model/Todo.java (Model class having variable and getters,setters)
  2. /src/main/java/com/in28minutes/springboot/web/service/TodoService.java (Business logic having add, update, delete, retrieve operations)
  3. /src/main/java/com/in28minutes/springboot/web/controller/TodoController.java(Here TodoService will be injected to get all the operation values in controller class
  4. src/main/webapp/WEB-INF/jsp/todo-list.jsp( This jsp will used to display todo list using model object passed from controller class

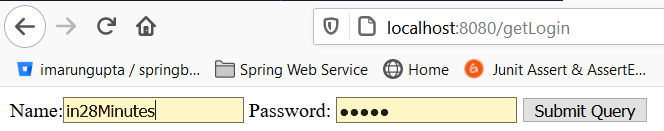
## Snippets

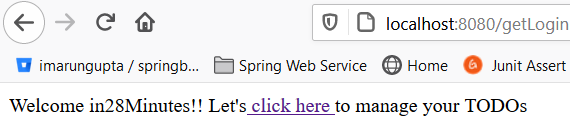
Snippet - /src/main/java/com/in28minutes/springboot/web/model/Todo.java

|  |
| --- |
| **package com.in28minutes.springboot.web.model;**  **import java.util.Date;**  **public class Todo {**  **private int id;**  **private String user;**  **private String desc;**  **private Date targetDate;**  **private boolean isDone;**  **public Todo(int id, String user, String desc, Date targetDate,**  **boolean isDone) {**  **super();**  **this.id = id;**  **this.user = user;**  **this.desc = desc;**  **this.targetDate = targetDate;**  **this.isDone = isDone;**  **}**  **// getters and setters**  **// override hashCode and equals method**  **// overriede toString method** |
| **Snippet - /src/main/java/com/in28minutes/springboot/web/service/TodoService.java**  **package com.in28minutes.springboot.web.service;**  **import java.util.ArrayList;**  **import java.util.Date;**  **import java.util.Iterator;**  **import java.util.List;**  **import org.springframework.stereotype.Service;**  **import com.in28minutes.springboot.web.model.Todo;**  **@Service**  **public class TodoService {**  **private static List<Todo> todos = new ArrayList<Todo>();**  **private static int todoCount = 3;**  **static {**  **todos.add(new Todo(1, "in28Minutes", "Learn Spring MVC", new Date(),false));**  **todos.add(new Todo(2, "in28Minutes", "Learn Struts", new Date(), false));**  **todos.add(new Todo(3, "in28Minutes", "Learn Hibernate", new Date(),false));**  **}**  **public List<Todo> retrieveTodos(String user) {**  **List<Todo> filteredTodos = new ArrayList<Todo>();**  **for (Todo todo : todos) {**  **if (todo.getUser().equals(user)) {**  **filteredTodos.add(todo);**  **}**  **}**  **return filteredTodos;**  **}**  **public void addTodo(String name, String desc, Date targetDate,**  **boolean isDone) {**  **todos.add(new Todo(++todoCount, name, desc, targetDate, isDone));**  **}**  **public void deleteTodo(int id) {**  **Iterator<Todo> iterator = todos.iterator();**  **while (iterator.hasNext()) {**  **Todo todo = iterator.next();**  **if (todo.getId() == id) {**  **iterator.remove();**  **}**  **}**  **}**  **}** |

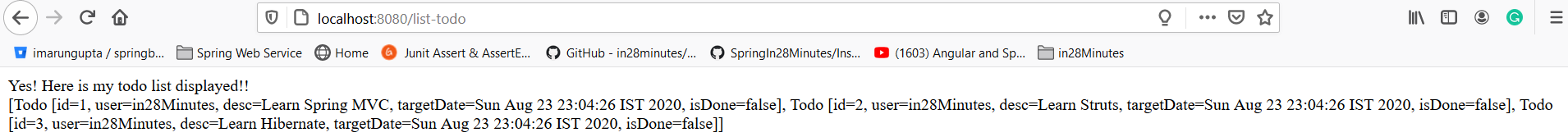
|  |
| --- |
| **package** com.in28minutes.springboot.web.controller;  **import** org.springframework.beans.factory.annotation.Autowired;  **import** org.springframework.stereotype.Controller;  **import** org.springframework.ui.ModelMap;  **import** org.springframework.web.bind.annotation.RequestMapping;  **import** org.springframework.web.bind.annotation.RequestMethod;  **import** com.in28minutes.springboot.web.service.TodoService;  @Controller  **publicclass** TodoController {    @Autowired  TodoService service;  @RequestMapping(value="/list-todo", method=RequestMethod.***GET***)  **public** String showTodos(ModelMap model) {  System.***out***.println("Inside list todo mentod controller");  model.put("todo", service.retrieveTodos("in28Minutes"));    **return**"todo-list";    }  } |
| <html>  <head>  <title>First Web Application</title>  </head>  <body>  Yes! Here is my todo list displayed!!</br>  ${todo}</br>  </body>  </html> |

****

****

****

**List of todo:**

****

## ****Step12-**** Session vs Model vs Request - @SessionAttributes

[**https://github.com/in28minutes/spring-boot-master-class/blob/master/02.Spring-Boot-Web-Application/Step12.md**](https://github.com/in28minutes/spring-boot-master-class/blob/master/02.Spring-Boot-Web-Application/Step12.md)

## What You Will Learn during this Step:

* Session vs Model vs Request.
* Be cautious about what you use Session for.
* @SessionAttributes("name") and how it works?
* Why use Model? "Adding elements directly to the HttpServletRequest (as request attributes) would seem to serve the same purpose. The reason to do this is obvious when taking a look at one of the requirements we have set for the MVC framework: It should be as view-agnostic as possible, which means we’d like to be able to incorporate view technologies not bound to the HttpServletRequest as well." - Rod Johnson etc. al’s book Professional Java Development with the Spring Framework
* Spring documentation states that the @SessionAttributes annotation “list the names of model attributes which should be transparently stored in the session or some conversational storage.”

Useful Snippets and References

First Snippet

@SessionAttributes("name")

**Files List**

pom.xml: Same dependencies-No change

In POM.xml: We have four main tags

1- Parent tag- spring-boot-starter-parent

2- **Properties**- java versiontags

3 -Dependencies-a- spring-boot-starter-web

tomcat-embed-jasper

spring-boot-devtools

spring-boot-starter-test

4-build maven plugin- spring-boot-maven-plugin

File – Java, properties and jsp

1. src/main/java/com/in28minutes/springboot/web/SpringBootFirstWebApplication.java
2. src/main/resources/application.properties
3. src/main/java/com/in28minutes/springboot/web/controller/LoginController.java
4. src/main/java/com/in28minutes/springboot/web/controller/TodoController.java
5. src/main/java/com/in28minutes/springboot/web/model/Todo.java
6. src/main/java/com/in28minutes/springboot/web/service/LoginService.java
7. src/main/java/com/in28minutes/springboot/web/service/TodoService.java
8. src/main/webapp/WEB-INF/jsp/login.jsp
9. src/main/webapp/WEB-INF/jsp/welcome.jsp
10. src/main/webapp/WEB-INF/jsp/list-todos.jsp
11. src/test/java/com/in28minutes/springboot/web/SpringBootFirstWebApplicationTests.java

|  |
| --- |
| **Spring Boot Launcher class: To launch the application**  package com.in28minutes.springboot.web;  import org.springframework.boot.SpringApplication;  import org.springframework.boot.autoconfigure.SpringBootApplication;  @SpringBootApplication  public class SpringBootFirstFormWebApplication {  public static void main(String[] args) {  SpringApplication.run(SpringBootFirstFormWebApplication.class, args);  }  } |
| **Application.properties: Mentioned the view prefix, suffix and logging level**  spring.mvc.view.prefix=/WEB-INF/jsp/  spring.mvc.view.suffix=.jsp  logging.level.org.springframework.web=DEBUG |
| **Creating LoginService.java : this service will be used to authenticate user credential and return true if credential matches**  **package** com.in28minutes.springboot.web.service;  **import** org.springframework.stereotype.Component;  @Component  **publicclass** LoginService {  **publicboolean** validateUser(String userid, String password) {  **return** userid.equalsIgnoreCase("in28Minutes") && password.equalsIgnoreCase("dummy");  }  } |
| **Create a** LoginController.java class to consume LoginService.java and rendering the welcome page after successfully login.  **package com.in28minutes.springboot.web.controller;**  **import org.springframework.beans.factory.annotation.Autowired;**  **import org.springframework.beans.factory.parsing.PassThroughSourceExtractor;**  **import org.springframework.stereotype.Controller;**  **import org.springframework.ui.ModelMap;**  **import org.springframework.web.bind.annotation.RequestMapping;**  **import org.springframework.web.bind.annotation.RequestMethod;**  **import org.springframework.web.bind.annotation.RequestParam;**  **import org.springframework.web.bind.annotation.SessionAttributes;**  **import com.in28minutes.springboot.web.service.LoginService;**  **@Controller**  **@SessionAttributes("nameKey")**  **public class LoginController {**    **@Autowired**  **LoginService loginservice;**    **@RequestMapping(value="/getLogin", method=RequestMethod.GET)**  **public String loginMessage( ModelMap model) {**  **System.out.println("inside login controll");**  **return"login";**  **}**  **@RequestMapping(value="/getLogin", method=RequestMethod.POST)**  **public String showWelcomePage(@RequestParam String name,@RequestParam String password,ModelMap model) {**    **boolean isValidate=loginservice.validateUser(name, password);**  **if(!isValidate) {**  **model.put("errorMsg", "Invalid Credentials");**  **return"login";**  **}else**  **model.put("nameKey", name);**  **model.put("passKey", password);**  **return"welcome";**  **}**  **}**  In the controller class we have Autowired loginService component class for validating credential. If credential does not matches then return login page with error message. Else return welcome page and putting username and password into model object to be displayed on the view page.  **Since request object or model object scope is available till the current request so we have kept name into SessionAttribute object so that we could use name object till application session scope.**  In other word the Request parameter or model object is available till the current request only. If we map the same Request parameter or model object further in the next sub sequential request it will lost their values. So to maintain the request parameter or model values in the entire request we will use SessionAttribute. |
| **Login.jsp**  <html>  <head>  <title>SpringBootFirstWebApplication</title>  </head>  <body>  <font color=*"red"*>${errorMsg}</font>  <form action=*""*method=*"post"*>  Name:<input type=*"text"* name=*"name"* id=*"name"*/>  Password: <input type=*"password"* name=*"password"* id=*"password"*/>  <input type=*"submit"*>  </form>  </body>  </html> |
| **Welcome.jsp**  <html>  <head>  <title>First Web Application</title>  </head>  <body>  Welcome ${nameKey}!! Let's<ahref=*"/list-todo"*> click here </a>to manage your TODOs  </body>  </html>  **Note**: Here the namekey value will be displayed because the model object will be available in the very first request. But if now we click on “Clcik here” link then it will direct us to the Todo list page and now the model object (namekey) value will be lost in this next request because the model object is present till current request only. |
| **Creating Model class :**  **package** com.in28minutes.springboot.web.model;  **import** java.util.Date;  **public class** Todo {  **private int** id;  **private** String user;  **private** String desc;  **private** Date targetDate;  **private Boolean** isDone;  **public** Todo(**int** id, String user, String desc, Date targetDate,  **Boolean** isDone) {  **super**();  **this**.id = id;  **this**.user = user;  **this**.desc = desc;  **this**.targetDate = targetDate;  **this**.isDone = isDone;  System.***out***.println("Called TODO Model class");  }  // getters setter, hashCode() and equals()  @Override  **public** String toString() {  **return** String.*format*(  "Todo [id=%s, user=%s, desc=%s, targetDate=%s, isDone=%s]", id,  user, desc, targetDate, isDone);  }  } |
| **TodoService.java class used to add id, user, desc, targetDate into static list.**  **package com.in28minutes.springboot.web.service;**  **import java.util.ArrayList;**  **import java.util.Date;**  **import java.util.Iterator;**  **import java.util.List;**  **import org.springframework.stereotype.Service;**  **import com.in28minutes.springboot.web.model.Todo;**  **@Service**  **public class TodoService {**  **private static List<Todo> todos = new ArrayList<Todo>();**  **private static int todoCount = 3;**  **// Static block to add id,user,desc,targetDate into list object.**  **static {**  **todos.add(new Todo(1, "in28Minutes", "Learn Spring MVC", new Date(),false));**  **todos.add(new Todo(2, "in28Minutes", "Learn Struts", new Date(), false));**  **todos.add(new Todo(3, "in28Minutes", "Learn Hibernate", new Date(),false));**  **}**  **// Method to retrieve the values from static list and putting into filteredTodos list which will be used in the controller class to display.**  **public List<Todo> retrieveTodos(String user) {**  **List<Todo> filteredTodos = new ArrayList<Todo>();**  **for (Todo todo : todos) {**  **if (todo.getUser().equals(user)) {**  **filteredTodos.add(todo);**  **}**  **}**  **return filteredTodos;**  **}**  **// Method to add new record into static todo list object**  **public void addTodo(String name, String desc, Date targetDate,**  **boolean isDone) {**  **todos.add(new Todo(++todoCount, name, desc, targetDate, isDone));**  **}**  **public void deleteTodo(int id) {**  **Iterator<Todo> iterator = todos.iterator();**  **while (iterator.hasNext()) {**  **Todo todo = iterator.next();**  **if (todo.getId() == id) {**  **iterator.remove();**  **}**  **}**  **}**  **}** |
| TodoController.java class  **package** com.in28minutes.springboot.web.controller;  **import**org.springframework.beans.factory.annotation.Autowired;  **import** org.springframework.stereotype.Controller;  **import** org.springframework.ui.ModelMap;  **import** org.springframework.web.bind.annotation.RequestMapping;  **import** org.springframework.web.bind.annotation.RequestMethod;  **import** org.springframework.web.bind.annotation.SessionAttributes;  **import** com.in28minutes.springboot.web.service.TodoService;  @Controller  @SessionAttributes("nameKey")  **publicclass** TodoController {    @**Autowired**  TodoService service;  @RequestMapping(value="/list-todo", method=RequestMethod.***GET***)  **public** String showTodos(ModelMap model) {  System.***out***.println("Inside list todo mentod controller");  String userFromSession = (String) model.get("nameKey");  System.***out***.println("userFromSession"+userFromSession);  //model.put("todo", service.retrieveTodos("in28Minutes"));  model.put("todo", service.retrieveTodos(userFromSession));    **return**"todo-list";  }  }   * 1. **Autowired TodoService component class to consume their methods**   2. **Get the nameKey from SessionAttributes**   3. **Retrieving the todo list and putting into model object.**   4. **Finally return the todo-list view where all the todo list is being displayed** |
| **Welcome.jsp**  <html>  <head>  <title>First Web Application</title>  </head>  <body>  Welcome ${nameKey}!! Let's<ahref=*"/list-todo"*> click here </a>to manage your TODOs  </body>  </html>  Here as we can see that we have created on href link [click here] which will call to /list-todo method and in this method it is returning [todo-list.jsp] to view the list of todos. |
| **todo-list.jsp**  <html>  <head>  <title>First Web Application</title>  </head>  <body>  Yes! Here is my todo list displayed!!</br>  ${todo}</br>  your name is${nameKey}  </body>  </html> |
|  |
| **If we provide correct credential then:** |
| **No on clicking on** click herewe get list of todos |

**Why the request parameter or model object is not available across the application scope?**

Since here whenever any request or response is made in the browser then it is called http request and http response and http is a stateless protocol, and they do not store any state and do don’t know what happens previously it stores only the current action or request. So if we want to store any conversation state (or object) then we have to do it on server side, so here on the server side we create a session which stores the value across the request. But note one thing; we should not put everything in the session because it will consume lot of server’s memory. So we have to very cautious for using session in the application.

## ****Step13- Add New Todo & use of (****redirect:/pageNameToberedirected)

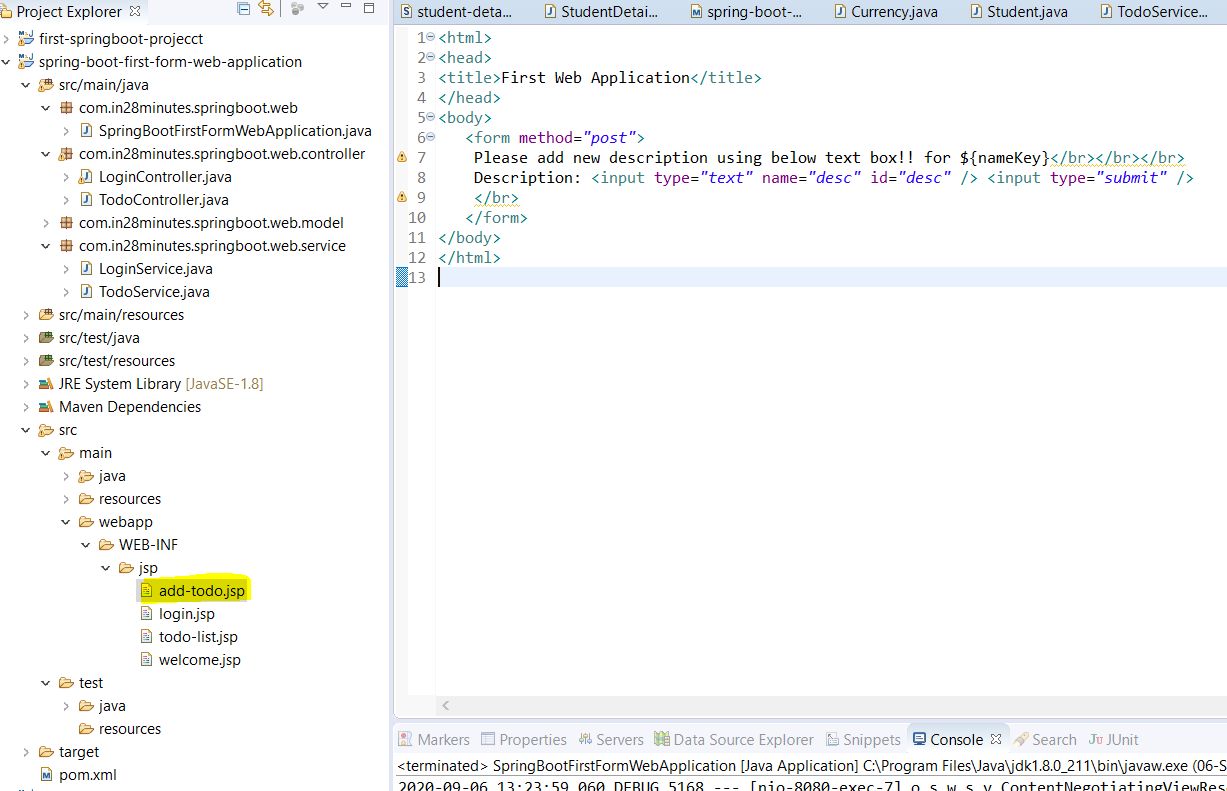
## https://github.com/in28minutes/spring-boot-master-class/blob/master/02.Spring-Boot-Web-Application/Step13.md

Now the next requirement is to add new description: And for that

* 1. In the **list-todo.jsp** create a link to add **<Add a Todo>**
  2. Create a jsp page(add-todo.jsp)having one text box and submit button to add todo which will be render on clicking of **<Add a Todo>** link
  3. In the **TodoService.java create a method to add description in the todo list.**
  4. Now in the TodoController class consume this method which will add desc in the todo list and after adding display the list-todo.jsp having newly added description as well.

|  |
| --- |
| Todo-list.jsl  <html>  <head>  <title>First Web Application</title>  </head>  <body>  Yes! Here is my todo list displayed!!</br>  ${todo}</br>  your name is${nameKey}  </br>  <a href=*"/add-todo"*>Add a Todo</a>  </body>  </html> |
| Add-todo.jsp  <html>  <head>  <title>First Web Application</title>  </head>  <body>  <form method=*"post"*>  Please add new description using below text box!! for ${nameKey}</br></br></br>  Description: <input type=*"text"*name=*"desc"*id=*"desc"*/><input type=*"submit"*/>  </br>  </form>  </body>  </html> |
| TodoService.java  **publicvoid**addTodo(String name, String desc, Date targetDate,  **boolean**isDone) {  *todos*.add(**new** Todo(++*todoCount*, name, desc, targetDate, isDone));  } |
| TodoController.java  @RequestMapping(value="/add-todo", method=RequestMethod.***GET***)  **public**String showAddTodoPage(ModelMap model) {  **return**"add-todo";    }  @RequestMapping(value="/add-todo", method=RequestMethod.***POST***)  **public** String addTodos(ModelMap model, @RequestParam String desc) {  System.***out***.println("Inside list todo mentod controller");  String userFromSession = (String) model.get("nameKey");  System.***out***.println("userFromSession"+userFromSession);  //model.put("todo", service.retrieveTodos("in28Minutes"));  service.addTodo(userFromSession, desc, **new** Date(), **false**);    //return "todo-list";  /\*  \* Here the above line we were trying to render the todo-list page having todo  \* list with newly added list. But the list was empty because here it was just  \* rendering jsp only so the solution is given below. Instead of using jsp page  \* we will use the request mapping url which populate the call the method to get  \* todo-list and display the same into todo-listjsp  \*/  **return**"redirect:/list-todo";  } |

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |



## ****Step14.md:** Display Todo in the table using JSTL tag**

## https://github.com/in28minutes/spring-boot-master-class/blob/master/02.Spring-Boot-Web-Application/Step14.md

## What we will do:

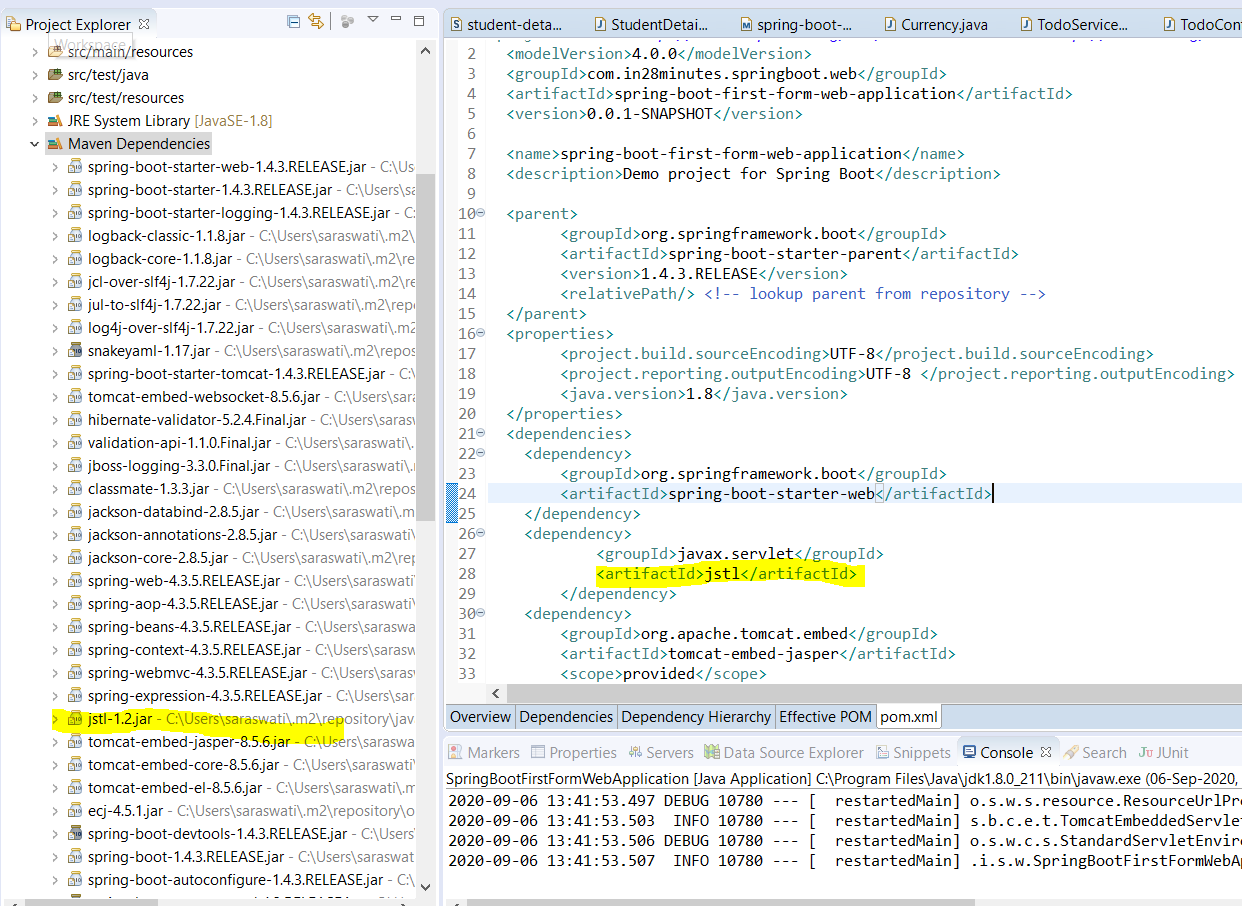
* Display Todos in a table using **JSTL Tags (Java Standard Tag Library). It provides lots of utility tag in jsp**
* <%@ taglib uri="<http://java.sun.com/jsp/jstl/core>" prefix="c"%>
* Add Dependency for jstl
* To enable JSTL tag add below dependency in POM.xml

**<dependency>**

**<groupId>javax.servlet</groupId>**

**<artifactId>jstl</artifactId>**

**</dependency>**



Note: After adding any dependency we need to restart the server.

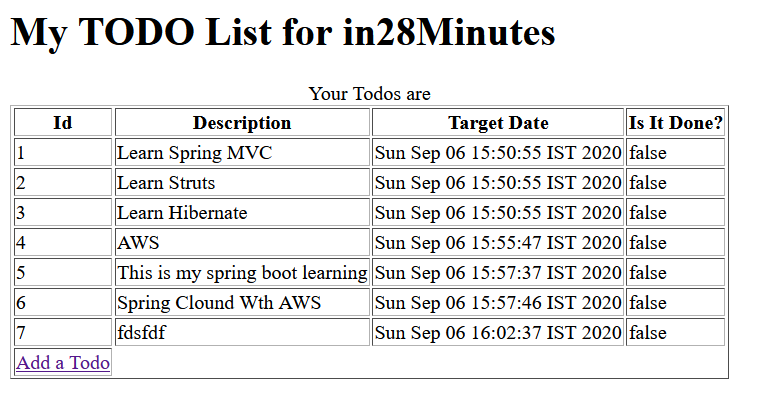
Here JSTL main role is to put the entire list into tabular format using foreach jstl function i.e. for better view of list of records. So mainly the changes will be happened in todo-list.jsp page as given below

* 1. Define jstl tag library on top of the jsp page.

<%@ taglib uri="<http://java.sun.com/jsp/jstl/core>" prefix="c"%>

* 1. First we will create one table and will create Table caption, table header and row inside the header.

|  |
| --- |
| **<%@tagliburi=*"http://java.sun.com/jsp/jstl/core"*prefix=*"c"*%>**  **<html>**  **<head>**  **<title>First Web Application TODO for ${nameKey}</title>**  **</head>**  **<body>**  **<h1>My TODO List for ${nameKey}</h1>**  **<tableborder=*"1"*>**  **<caption>Your Todos are</caption>**  **<thead>**  **<tr>**  **<th>Id</th>**  **<th>Description</th>**  **<th>Target Date</th>**  **<th>Is It Done?</th>**  **</tr>**  **</thead>**  **<tbody>**  **<!-- use JStl for loop todo list. We will use forEach loop-->**  **<!-- forEach items in todo list create one variable todoItem -->**  **<!-- using var name access the model class variable just like calling getter method to get values in JSTL instead of getter method just use the variable name to get value -->**  **<c:forEachitems=*"*${todo}*"*var=*"todoItem"*>**  **<tr>**  **<td>${todoItem.id}</td>**  **<td>${todoItem.desc}</td>**  **<td>${todoItem.targetDate}</td>**  **<td>${todoItem.done}</td>**  **</tr>**  **</c:forEach>**  **</tbody>**  **<tr>**  **<td><ahref=*"/add-todo"*>Add a Todo</a></td>**  **</tr>**  **</table>**  **<%-- Yes! Here is my todo list displayed!!</br>**  **${todo}</br>**  **your name is${nameKey} --%>**    **<!-- <a href="/add-todo">Add a Todo</a> -->**  **</body>**  **</html>** |



**Step15.md: Bootstrap for page formatting using webjars**

[**https://github.com/in28minutes/spring-boot-master-class/blob/master/02.Spring-Boot-Web-Application/Step15.md**](https://github.com/in28minutes/spring-boot-master-class/blob/master/02.Spring-Boot-Web-Application/Step15.md)

**What we will do:**

* Add bootstrap to give basic formatting to the page: We use bootstrap classes container, table and table-striped.
* We will add webjars dependency in the POM.xml

It is already auto configured by Spring Boot : o.s.w.s.handler.SimpleUrlHandlerMapping : Mapped URL path [/webjars/\*\*] onto handler of type [class org.springframework.web.servlet.resource.ResourceHttpRequestHandler]

**Useful Snippets**

|  |
| --- |
| <dependency>  <groupId>org.webjars</groupId>  <artifactId>bootstrap</artifactId>  <version>3.3.6</version>  </dependency>  <dependency>  <groupId>org.webjars</groupId>  <artifactId>jquery</artifactId>  <version>1.9.1</version>  </dependency>  <script src="webjars/jquery/1.9.1/jquery.min.js"></script>  <script src="webjars/bootstrap/3.3.6/js/bootstrap.min.js"></script>  <link href="webjars/bootstrap/3.3.6/css/bootstrap.min.css"rel="stylesheet"> |

**WebJars concept** Generally if we want use spring boot framework then usually we need to add spring boot parent and some of its dependency which gets downloaded automatically **similarly** if we want to use java script (e.g. JQuery) or CSS file in our application then we need to download the source file and need to put into WEB-INF folder so that we can use its features into html or JSP file.

Now if we want to upgrade this java script and CSS in future then we need to download the latest version and need to put physically in the application inside WEB-INF folder of the application.

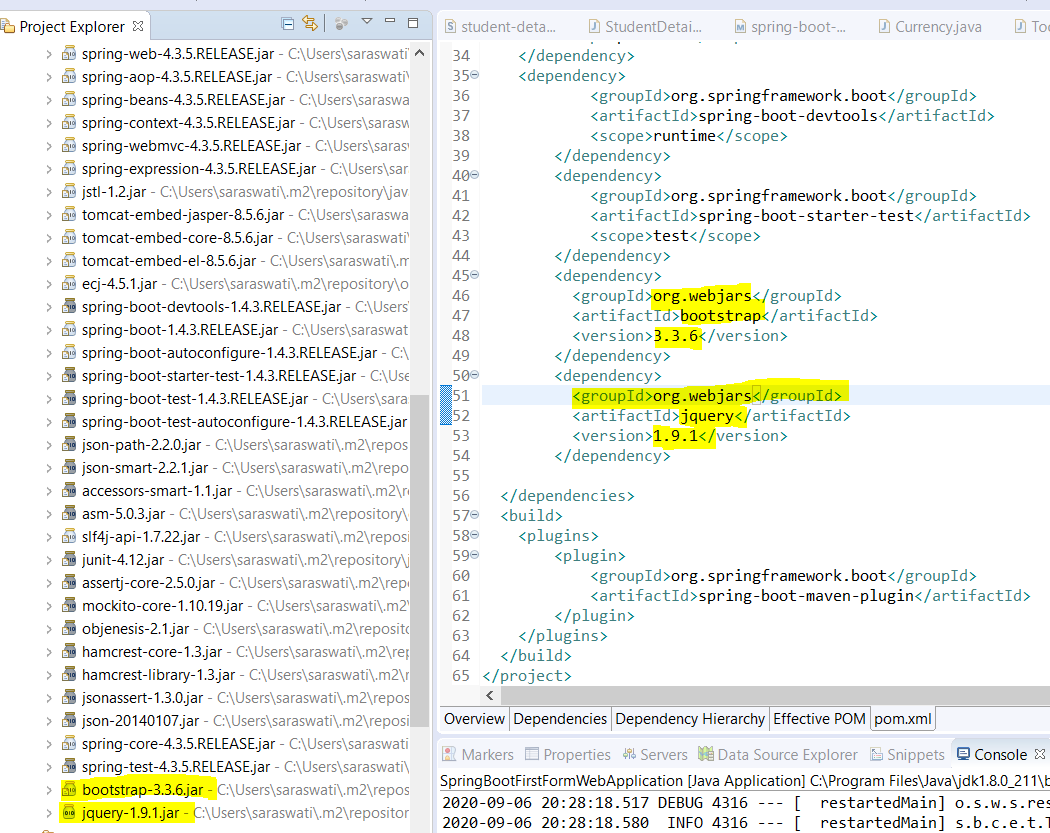
To avoid this headache, WebJars concept came into picture. Now it came into maven dependency which we can put into POM.xml and whenever we want to upgrade, we just need to change version into POM.xml file and everything will be take care automatically. Using this WebJars concept we can upgrade the version of these static files (java script and CSS)

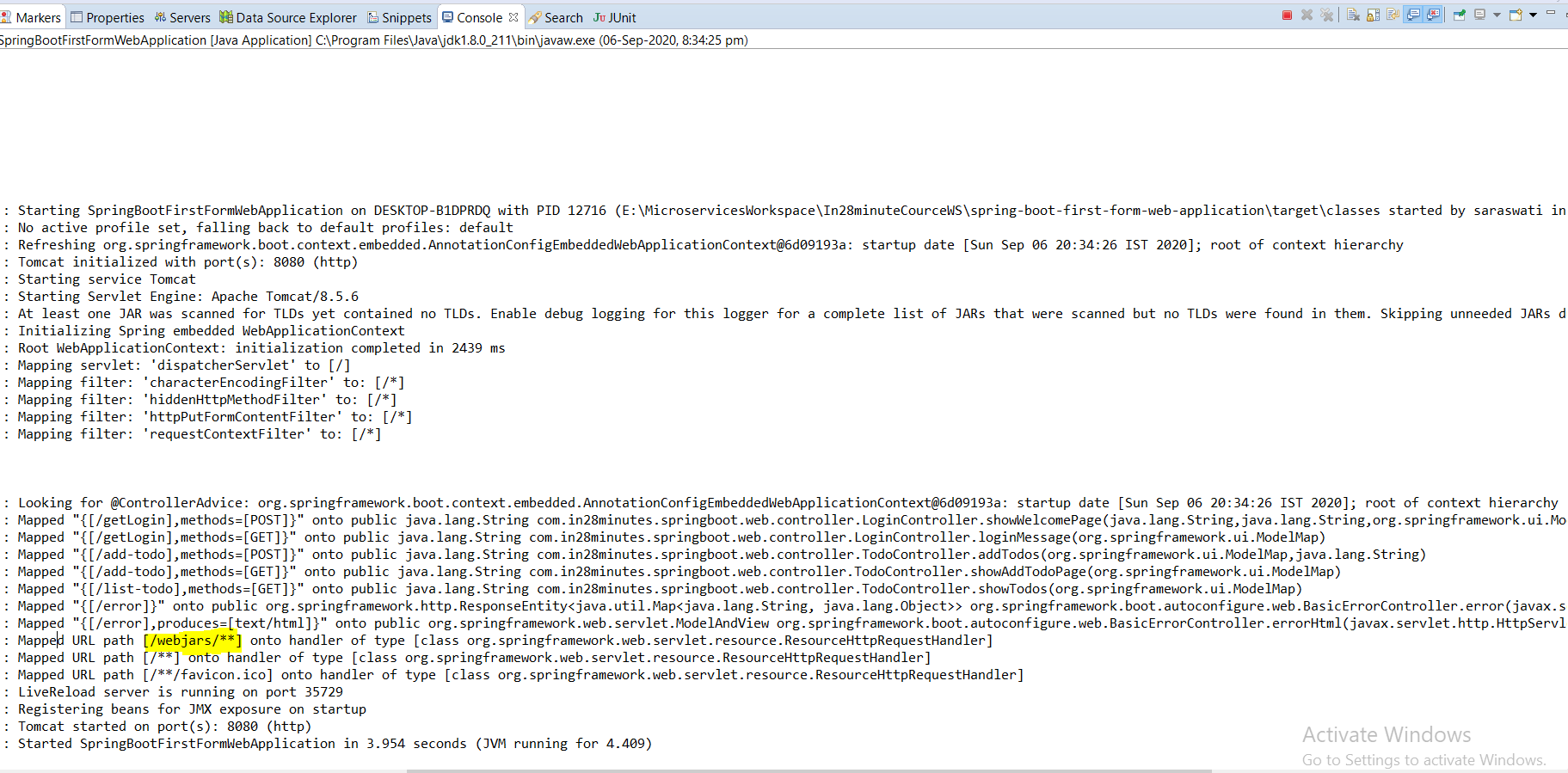
After adding the above two dependency as soon as we restart the server these two dependency gets added in the maven dependency as shown in the below image.

**Go get CSS framework bootstrap** in the application, we need to have three files

1. jquery.min.**js**
2. bootstrap.min.**js**
3. bootstrap.min.**css**

WebJars basically static things which takes these three files and zipped up in the form of WebJars and made available in the maven repository. We can version then similar to other jars. So here we have WebJars for bootstrap and WebJars for JQuery. Since to be able to work with bootstrap, we need JQuery as well, that is why along with WebJars bootstrap we have includes WebJars for JQuery also.And to be able to use in JSP we have to include link in to jsp. So the best practice CSS will be inside <head> and javaScript will inside bottom of <body> tag.





**As here in console we can see that WebJars is auto configured which spring boot take care for this auto configuration.**

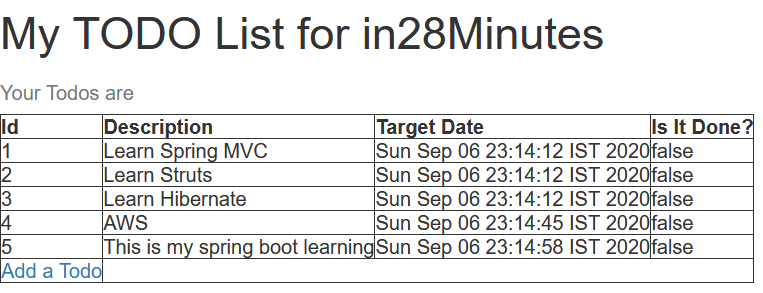
To be able to use bootstrap we need JQuery that is why we have added in the POM.xml and to be able to use these we need to link boot strap and JQuery in our JSP (todo-list.jsp). The best practice is that we will add these scripts at the bottom of the page. And CSS will be added inside the <head> tag of the JSP. So that during the page load CSS could be loaded ASAP and java script could be loaded as late as possible.

The path of the java script and CSS file can be seen in maven dependency by expanding the particular jar.

|  |  |
| --- | --- |
|  |  |

So to add boot strap we need to add these two java script and one CSS as mentioned in the JSP.

**Now after adding these two script and css file, once we reload the page then the outcome will look like as below.**



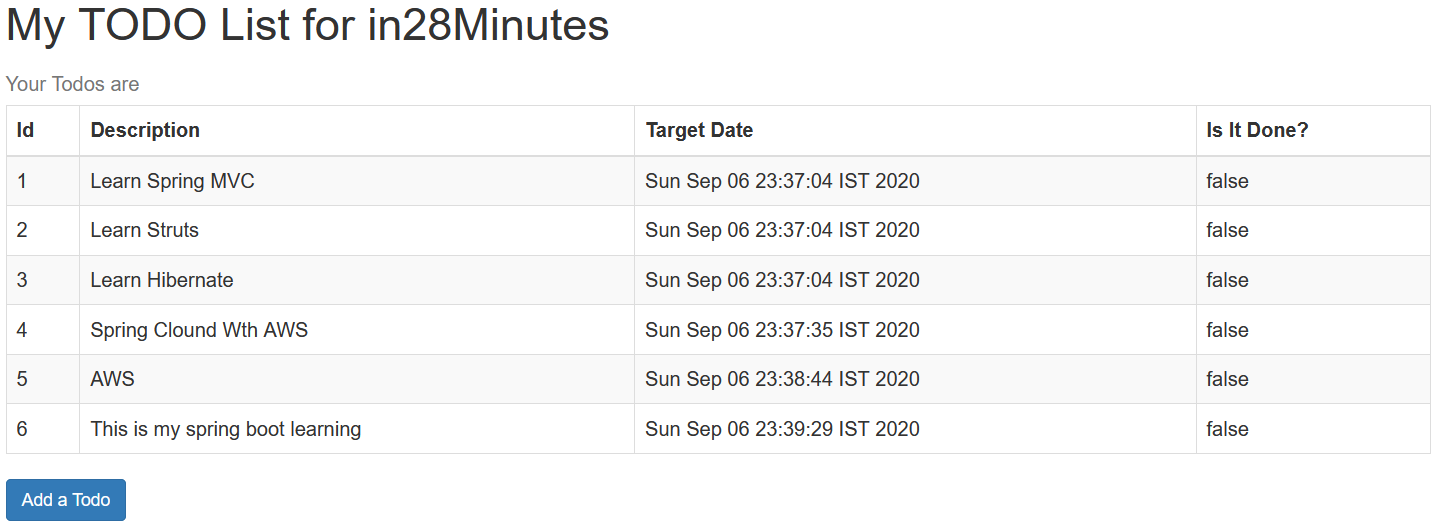
Now to make it more good looking let’s use some more boot strap classes.

* One of the important thing about boot strap is everything will put inside <div> tag and inside this tag we will use CSS class =”container” [<divclass=*"container"*>]
* The other thing inside <table> tag we can use the class=”table” i.e. <table class=”table”>. To make it better we can use class=”table-striped” and for table border we can use class=”*table-bordered”*

So all the table class can be mentioned as <tableclass=*"table table-striped table-bordered"*>

* For button <div><aclass=*"btn btn-primary"*href=*"/add-todo"*>Add a Todo</a></div>

|  |
| --- |
| <%@tagliburi=*"http://java.sun.com/jsp/jstl/core"*prefix=*"c"*%>  <html>  <head>  <title>First Web Application TODO for ${nameKey}</title>  <linkhref=*"webjars/bootstrap/3.3.6/css/bootstrap.min.css"*  rel=*"stylesheet"*>  </head>  <body>  <divclass=*"container"*>  <h1class=*"text-justify"*>My TODO List for ${nameKey}</h1>  <tableclass=*"table table-striped table-bordered"*>  <caption>Your Todos are</caption>  <thead>  <tr>  <th>Id</th>  <th>Description</th>  <th>Target Date</th>  <th>Is It Done?</th>  </tr>  </thead>  <tbody>  **<!-- use JStl for loop todo list. We will use forEach loop-->**  **<!-- forEach items in todo list create one variable todoItem -->**  **<!-- using var name access the model class variable just like calling getter method to get values –>**  <c:forEachitems=*"*${todo}*"*var=*"todoItem"*>  <tr>  <td>${todoItem.id}</td>  <td>${todoItem.desc}</td>  <td>${todoItem.targetDate}</td>  <td>${todoItem.done}</td>  </tr>  </c:forEach>  </tbody>  </table>  <div>  <aclass=*"btn btn-primary"*href=*"/add-todo"*>Add a Todo</a>  </div>  <scriptsrc=*"webjars/jquery/1.9.1/jquery.min.js"*></script>  <scriptsrc=*"webjars/bootstrap/3.3.6/js/bootstrap.min.js"*></script>  </div>  </body>  </html> |



**Hence using Bootstrap CSS framework we have formatted our page very well.**

**Step16.md**

[**https://github.com/in28minutes/spring-boot-master-class/blob/master/02.Spring-Boot-Web-Application/Step16.md**](https://github.com/in28minutes/spring-boot-master-class/blob/master/02.Spring-Boot-Web-Application/Step16.md)

yet to write code for delete

Note: COURSE UPDATE: Use modelAttribute instead of commandName

**With recent changes in Spring Boot, you might face this error in next step**

1. org.apache.jasper.JasperException: /WEB-INF/views/todo.jsp (line: [20], column: [4]) Unable to find setter method for attribute: [commandName]
2. at org.apache.jasper.compiler.DefaultErrorHandler.jspError(DefaultErrorHandler.java:42) ~[tomcat-embed-jasper-9.0.26.jar:9.0.26]

The solution is to use modelAttribute instead of commandName.

<form:form method="post" modelAttribute="todo">

For more - <https://docs.spring.io/spring/docs/current/javadoc-api/org/springframework/web/bind/annotation/ModelAttribute.html>

Fullscreen

**Step17.md**

[**https://github.com/in28minutes/spring-boot-master-class/blob/master/02.Spring-Boot-Web-Application/Step17.md**](https://github.com/in28minutes/spring-boot-master-class/blob/master/02.Spring-Boot-Web-Application/Step17.md)

**What we will do:**

In this short step:

* Format Add Todo Page
* Add Html5 Form Validations

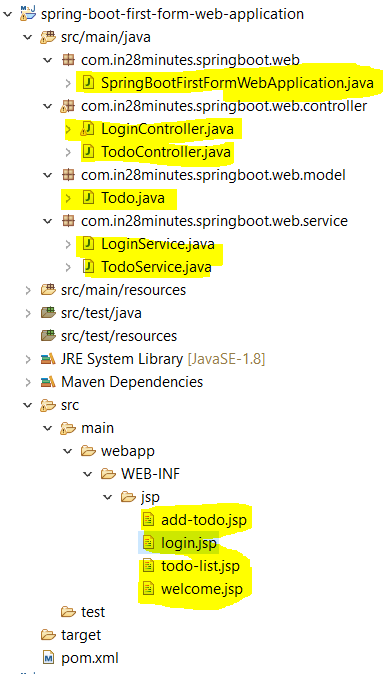
**Useful Snippets**

<fieldset class="form-group">

<label>Description</label>

<input name="desc" type="text" class="form-control" required="required"/>

</fieldset>



Files List

1. pom.xml: No change, nothing added or deleted
2. src/main/java/com/in28minutes/springboot/web/controller/LoginController.java
3. src/main/java/com/in28minutes/springboot/web/controller/TodoController.java
4. src/main/java/com/in28minutes/springboot/web/model/Todo.java
5. src/main/java/com/in28minutes/springboot/web/service/LoginService.java
6. src/main/java/com/in28minutes/springboot/web/service/TodoService.java
7. src/main/java/com/in28minutes/springboot/web/SpringBootFirstWebApplication.java
8. src/main/resources/application.properties
9. src/main/webapp/WEB-INF/jsp/list-todos.jsp
10. src/main/webapp/WEB-INF/jsp/login.jsp
11. src/main/webapp/WEB-INF/jsp/todo.jsp
12. src/main/webapp/WEB-INF/jsp/welcome.jsp
13. src/test/java/com/in28minutes/springboot/web/SpringBootFirstWebApplicationTests.java : No changes

|  |
| --- |
| src/main/java/com/in28minutes/springboot/web/controller/LoginController.java  **package com.in28minutes.springboot.web.controller;**  **import org.springframework.beans.factory.annotation.Autowired;**  **import org.springframework.stereotype.Controller;**  **import org.springframework.ui.ModelMap;**  **import org.springframework.web.bind.annotation.RequestMapping;**  **import org.springframework.web.bind.annotation.RequestMethod;**  **import org.springframework.web.bind.annotation.RequestParam;**  **import org.springframework.web.bind.annotation.SessionAttributes;**  **import com.in28minutes.springboot.web.service.LoginService;**  **@Controller**  **@SessionAttributes("name")**  **public class LoginController {**    **@Autowired**  **LoginService service;**    **@RequestMapping(value="/login", method = RequestMethod.GET)**  **public String showLoginPage(ModelMap model){**  **return "login";**  **}**    **@RequestMapping(value="/login", method = RequestMethod.POST)**  **public String showWelcomePage(ModelMap model, @RequestParam String name, @RequestParam String password){**    **boolean isValidUser = service.validateUser(name, password);**    **if (!isValidUser) {**  **model.put("errorMessage", "Invalid Credentials");**  **return "login";**  **}**  **model.put("name", name);**  **model.put("password", password);**  **return "welcome";**  **}**  **}** |

|  |
| --- |
| src/main/java/com/in28minutes/springboot/web/controller/TodoController.java  package com.in28minutes.springboot.web.controller;  import java.util.Date;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.stereotype.Controller;  import org.springframework.ui.ModelMap;  import org.springframework.web.bind.annotation.RequestMapping;  import org.springframework.web.bind.annotation.RequestMethod;  import org.springframework.web.bind.annotation.RequestParam;  import org.springframework.web.bind.annotation.SessionAttributes;  import com.in28minutes.springboot.web.service.LoginService;  import com.in28minutes.springboot.web.service.TodoService;  @Controller  @SessionAttributes("name")  public class TodoController {    @Autowired  TodoService service;    @RequestMapping(value="/list-todos", method = RequestMethod.GET)  public String showTodos(ModelMap model){  String name = (String) model.get("name");  model.put("todos", service.retrieveTodos(name));  return "list-todos";  }    @RequestMapping(value="/add-todo", method = RequestMethod.GET)  public String showAddTodoPage(ModelMap model){  return "todo";  }  @RequestMapping(value="/delete-todo", method = RequestMethod.GET)  public String deleteTodo(@RequestParam int id){  service.deleteTodo(id);  return "redirect:/list-todos";  }  @RequestMapping(value="/add-todo", method = RequestMethod.POST)  public String addTodo(ModelMap model, @RequestParam String desc){  service.addTodo((String) model.get("name"), desc, new Date(), false);  return "redirect:/list-todos";  }  } |

|  |
| --- |
| **src/main/java/com/in28minutes/springboot/web/model/Todo.java**  package com.in28minutes.springboot.web.model;  import java.util.Date;  public class Todo {  private int id;  private String user;  private String desc;  private Date targetDate;  private boolean isDone;  public Todo(int id, String user, String desc, Date targetDate,  boolean isDone) {  super();  this.id = id;  this.user = user;  this.desc = desc;  this.targetDate = targetDate;  this.isDone = isDone;  }  // Getters and setters and hashCode() and equals() and toString()  @Override  public String toString() {  return String.format(  "Todo [id=%s, user=%s, desc=%s, targetDate=%s, isDone=%s]", id,  user, desc, targetDate, isDone);  }  } |
| **src/main/java/com/in28minutes/springboot/web/service/LoginService.java**  **package com.in28minutes.springboot.web.service;**  **import org.springframework.stereotype.Component;**  **import org.springframework.stereotype.Service;**  **@Service**  **public class LoginService {**  **public boolean validateUser(String userid, String password) {**  **// in28minutes, dummy**  **return userid.equalsIgnoreCase("in28minutes")**  **&& password.equalsIgnoreCase("dummy");**  **}**  **}** |
| **src/main/java/com/in28minutes/springboot/web/service/TodoService.java**  **package com.in28minutes.springboot.web.service;**  **import java.util.ArrayList;**  **import java.util.Date;**  **import java.util.Iterator;**  **import java.util.List;**  **import org.springframework.stereotype.Service;**  **import com.in28minutes.springboot.web.model.Todo;**  **@Service**  **public class TodoService {**  **private static List<Todo> todos = new ArrayList<Todo>();**  **private static int todoCount = 3;**  **static {**  **todos.add(new Todo(1, "in28Minutes", "Learn Spring MVC", new Date(),**  **false));**  **todos.add(new Todo(2, "in28Minutes", "Learn Struts", new Date(), false));**  **todos.add(new Todo(3, "in28Minutes", "Learn Hibernate", new Date(),**  **false));**  **}**  **public List<Todo> retrieveTodos(String user) {**  **List<Todo> filteredTodos = new ArrayList<Todo>();**  **for (Todo todo : todos) {**  **if (todo.getUser().equals(user)) {**  **filteredTodos.add(todo);**  **}**  **}**  **return filteredTodos;**  **}**  **public void addTodo(String name, String desc, Date targetDate,**  **boolean isDone) {**  **todos.add(new Todo(++todoCount, name, desc, targetDate, isDone));**  **}**  **public void deleteTodo(int id) {**  **Iterator<Todo> iterator = todos.iterator();**  **while (iterator.hasNext()) {**  **Todo todo = iterator.next();**  **if (todo.getId() == id) {**  **iterator.remove();**  **}**  **}**  **}**  **}** |
| **src/main/java/com/in28minutes/springboot/web/SpringBootFirstWebApplication.java**  **package com.in28minutes.springboot.web;**  **import org.springframework.boot.SpringApplication;**  **import org.springframework.boot.autoconfigure.SpringBootApplication;**  **import org.springframework.context.annotation.ComponentScan;**  **@SpringBootApplication**  **@ComponentScan("com.in28minutes.springboot.web")**  **public class SpringBootFirstWebApplication {**  **public static void main(String[] args) {**  **SpringApplication.run(SpringBootFirstWebApplication.class, args);**  **}**  **}** |
| **src/main/resources/application.properties**  **spring.mvc.view.prefix=/WEB-INF/jsp/**  **spring.mvc.view.suffix=.jsp**  **logging.level.org.springframework.web=INFO** |
| **src/main/webapp/WEB-INF/jsp/list-todos.jsp**  **<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>**  **<html>**  **<head>**  **<title>Todo's for ${name}</title>**  **<link href="webjars/bootstrap/3.3.6/css/bootstrap.min.css" rel="stylesheet">**  **</head>**  **<body>**  **<div class="container">**  **<table class="table table-striped">**  **<caption>Your todos are</caption>**  **<thead>**  **<tr>**  **<th>Description</th>**  **<th>Target Date</th>**  **<th>Is it Done?</th>**  **<th></th>**  **</tr>**  **</thead>**  **<tbody>**  **<c:forEach items="${todos}" var="todo">**  **<tr>**  **<td>${todo.desc}</td>**  **<td>${todo.targetDate}</td>**  **<td>${todo.done}</td>**  **<td><a type="button" class="btn btn-warning"**  **href="/delete-todo?id=${todo.id}">Delete</a></td>**  **</tr>**  **</c:forEach>**  **</tbody>**  **</table>**  **<div>**  **<a class="button" href="/add-todo">Add a Todo</a>**  **</div>**  **</div>**  **<script src="webjars/jquery/1.9.1/jquery.min.js"></script>**  **<script src="webjars/bootstrap/3.3.6/js/bootstrap.min.js"></script>**  **</body>**  **</html>** |
| **src/main/webapp/WEB-INF/jsp/login.jsp**  **<html>**  **<head>**  **<title>First Web Application</title>**  **</head>**  **<body>**  **<font color="red">${errorMessage}</font>**  **<form method="post">**  **Name : <input type="text" name="name" />**  **Password : <input type="password" name="password" />**  **<input type="submit" />**  **</form>**  **</body>**  **</html>** |
| **src/main/webapp/WEB-INF/jsp/todo.jsp**  **<html>**  **<head>**  **<title>First Web Application</title>**  **<link href="webjars/bootstrap/3.3.6/css/bootstrap.min.css" rel="stylesheet">**  **</head>**  **<body>**  **<div class="container">**  **<form method="post">**  **<fieldset class="form-group">**  **<label>Description</label>**  **<input name="desc" type="text"**  **class="form-control" required="required"/>**  **</fieldset>**  **<button type="submit" class="btn btn-success">Add</button>**  **</form>**  **</div>**  **<script src="webjars/jquery/1.9.1/jquery.min.js"></script>**  **<script src="webjars/bootstrap/3.3.6/js/bootstrap.min.js"></script>**  **</body>**  **</html>** |
| **src/main/webapp/WEB-INF/jsp/welcome.jsp**  **<html>**  **<head>**  **<title>First Web Application</title>**  **</head>**  **<body>**  **Welcome ${name}!! <a href="/list-todos">Click here</a> to manage your todo's.**  **</body>**  **</html>** |
|  |

todo.txt

LoginController -> adds name to model

welcome.jsp -> shows ${name}

TodoController -> redirects to list-todos.jsp

${name} is empty

Component, Service, Repository, Controller

Autowired

ComponentScan

Field dummyService in com.in28minutes.springboot.web.controller.LoginController

required a bean of type 'com.in28minutes.dummy.DummyService'

that could not be found.

Spring Boot Starter Parent

Spring Boot Starter Web

@SpringBootApplication

Auto Configuration

Dispatcher Servlet

/login => "login"

"login" => src/main/webapp/WEB-INF/jsp/login.jsp

Search for a view named "login"

/login => LoginController

**In html 5, all the fields should be inside <filedset> and for fields to be mandatory we can put required attribute in the input box.**

**Important Note:**

COURSE UPDATE : Add dependency spring-boot-starter-validation

If you are making use of Spring Boot Release (> 2.3.0) make sure to add the following dependency to your pom.xml (We are going to implement validations in next step):

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-validation</artifactId>

</dependency>

**RECOMMENDATION**

Stop and start your server again after adding the dependency.

**If you are of the curious kind:**

Here's the reason you need to add the dependency - <https://github.com/spring-projects/spring-boot/issues/19550>

**Step18.md(part-1)**

[**https://github.com/in28minutes/spring-boot-master-class/blob/master/02.Spring-Boot-Web-Application/Step18.md**](https://github.com/in28minutes/spring-boot-master-class/blob/master/02.Spring-Boot-Web-Application/Step18.md)

**What we will do:**

* Lets use a command bean or modelAttribute for Todo
* Add Validations
* The JSR 303 and JSR 349 defines specification for the Bean Validation API (version 1.0 and 1.1, respectively), and Hibernate Validator is the reference implementation.
* org.hibernate:hibernate-validator

In the previous step we have formatted and validate few of the input fields of jsp page using HTML 5. But apart from client side validation we have to depend on server side validation also. Basic validation in the client side we can do using java script but apart from client side validation we have to do server side validation also. In this step we will learn how to do validation in the server side.

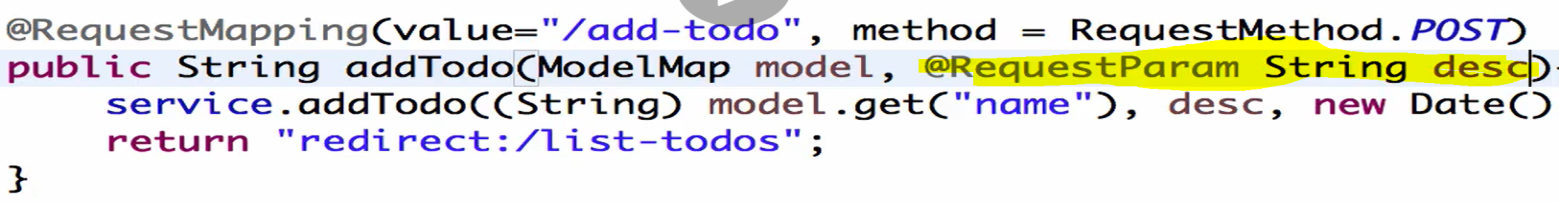
We will use the JSR 303 and JSR 349 defines specification for the Bean Validation API (version 1.0 and 1.1, respectively), and Hibernate Validator is the reference implementation. So we will use validation on top of our beans and we would use the reference implementation Hibernate Validator.

Note: We are not going to use Hibernate which is most popular implementation of JPA, here we are using Hibernate Validator just to validate stuffs. For example if want to validate bean length (at least 4 character needs to enter) then these kind of validation we can do from hibernate Validator.

Implementation server side validation using Bean Validation API involves few steps, one of them concept called **Command Bean or Form backing bean**.

Until now we have the description entered by the user

|  |  |
| --- | --- |
| So what are we doing here? | As shown below in the controller, we are assigning a RequestParam of type string. But if we have multiple fields to enter then, in this case we will have to define several RequestParam, which will be little difficult to manage. This kind of request is good for one or two RequestParam, but for multiple fields this is not suitable. |



|  |  |
| --- | --- |
|  |  |

So here we can directly map Todo model class bean directly to the form fields. This concept is called form backing object or command bean.

Once we setup the command bean, we can **add validation** on the command beans using bean validation API. Once we add the validation on command beans, we can use them in the controller and if there any validation error then we can show those errors on the View.

So Implementation of server side validation using Bean Validation API involves few steps.

1. Setup Command Bean or Form Backing Bean in the controller class
2. Add validation on command bean
3. Use validation in the Controller class
4. Display errors in the view

To handle multiple input fields in the form, spring brings one concept called [CommandBean or Form Backing Bean]. So instead of using multiple @RequestParam in the RequestMapping method we can use Model class object as shown below in the screen.

If we want to use command bean then two things we have to do.

1. One thing is implement Model class object it in the controller instead of using @RequestParam
2. Second thing, we need to use spring form tag in the view (jsp). Spring MVC has its own form tag library.

So we need to use this command bean in the Controller as well as in the View so that both side binding would happen. So the values from the form (view) will be bound bean (Controller) and the values from the bean (Controller) will be bound to the form (view) – Vice-versa.

Let’s start updating with controller (TodoController.java).

|  |  |
| --- | --- |
| **Without command bean** | **With command bean** |

Now let’s go to view Todo.jsp As shown in the Snippet.

1. First of all add the spring framework tab library on top of the jsp page as shown below in the snippet.
2. Add the prefix (form) in all the <form> tab libraries like [<form:form> ,<form:input> , <from:label>]
3. Now we have to bind Todo todo bean (which got created in controller class as shown in above screen with view (jsp) form label and its input tag. So as shown in the below code inside form tag after method =POST we have used **commandName="todo"** or **modelAttribute=”todo”** which binds model class bean to JSP.
4. Now if we want to add default Todo object in the in the view(jsp page) input. So we will add this object into Model in the method which renders the todo jsp view is defined. For example here we have written one method ShowAddTodoPage(), here only we will add todo in model attribute.

|  |
| --- |
| @RequestMapping(value = "/add-todo", method = RequestMethod.GET)  public String showAddTodoPage(ModelMap model) {  model.addAttribute("**todo**", new Todo(0, (String) model.get("name"), "Default Desc",new Date(), false));  return "todo";  } |

Here the highlighted line set the default value into the view(jsp page), so when this todo.jsp page will displayed, “Default Desc” will be shown as default input. So using model.addAttribute () we can bind the model class object and this todo object set above in the model addAttribute will be available in the view (jsp page).

1. Now to bind bean object with form tag instead of name attribute here in the spring tag library, **path** attribute is being used, which gets converted in name tag in the run time.
2. Note: In the latest version of spring instead of commandName , now modelAttribute is being used. So commandName and modelAttribute both are same only now just name is changed.

**Useful Snippets(**Todo.jsp)

|  |
| --- |
| **<%@taglib uri="http://www.springframework.org/tags/form" prefix="form"%>**  **<!-- <form:form method="post" commandName="todo"> -->**  <**form:form** method="post" **modelAttribute="todo**">  <fieldset class="form-group">  <**form:label** **path="desc"**>Description</**form:label**>  <**form:input** **path="desc"** type="text" class="form-control" required="required"/>  </fieldset>  </**form:form**>  @Size(min = 10, message = "Enter atleast 10 Characters.")  @Valid Todo todo, BindingResult result  if (result.hasErrors())  return "todo";  <form:errors path="desc" cssClass="text-warning" /> |

**Note:** Now after making these changes if we run the above code then we will get will get below error. Because of the thing in spring binding it expect one no argument default constructor.



So we will have to add no argument default constructor.

public Todo() {

super ();

}

Now let’s try to below code run in the eclipse:

**pom.xml**

<?xml version="1.0" encoding="UTF-8"?>

<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">

<modelVersion>4.0.0</modelVersion>

<groupId>com.in28minutes.springboot.web</groupId>

<artifactId>spring-boot-first-web-application</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>spring-boot-first-web-application</name>

<description>Demo project for Spring Boot</description>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>1.4.3.RELEASE</version>

<relativePath /> <!-- lookup parent from repository -->

</parent>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>

<java.version>1.8</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>jstl</artifactId>

</dependency>

<dependency>

<groupId>org.webjars</groupId>

<artifactId>bootstrap</artifactId>

<version>3.3.6</version>

</dependency>

<dependency>

<groupId>org.webjars</groupId>

<artifactId>jquery</artifactId>

<version>1.9.1</version>

</dependency>

<dependency>

<groupId>org.apache.tomcat.embed</groupId>

<artifactId>tomcat-embed-jasper</artifactId>

<scope>provided</scope>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-devtools</artifactId>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

src/main/java/com/in28minutes/springboot/web/controller/LoginController.java

package com.in28minutes.springboot.web.controller;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Controller;

import org.springframework.ui.ModelMap;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestMethod;

import org.springframework.web.bind.annotation.RequestParam;

import org.springframework.web.bind.annotation.SessionAttributes;

import com.in28minutes.springboot.web.service.LoginService;

@Controller

@SessionAttributes("name")

public class LoginController {

@Autowired

LoginService service;

@RequestMapping(value="/login", method = RequestMethod.GET)

public String showLoginPage(ModelMap model){

return "login";

}

@RequestMapping(value="/login", method = RequestMethod.POST)

public String showWelcomePage(ModelMap model, @RequestParam String name, @RequestParam String password){

boolean isValidUser = service.validateUser(name, password);

if (!isValidUser) {

model.put("errorMessage", "Invalid Credentials");

return "login";

}

model.put("name", name);

model.put("password", password);

return "welcome";

}

}

src/main/java/com/in28minutes/springboot/web/controller/TodoController.java

package com.in28minutes.springboot.web.controller;

import java.util.Date;

import javax.validation.Valid;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Controller;

import org.springframework.ui.ModelMap;

import org.springframework.validation.BindingResult;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestMethod;

import org.springframework.web.bind.annotation.RequestParam;

import org.springframework.web.bind.annotation.SessionAttributes;

import com.in28minutes.springboot.web.model.Todo;

import com.in28minutes.springboot.web.service.LoginService;

import com.in28minutes.springboot.web.service.TodoService;

@Controller

@SessionAttributes("name")

public class TodoController {

@Autowired

TodoService service;

@RequestMapping(value = "/list-todos", method = RequestMethod.GET)

public String showTodos(ModelMap model) {

String name = (String) model.get("name");

model.put("todos", service.retrieveTodos(name));

return "list-todos";

}

**@RequestMapping(value = "/add-todo", method = RequestMethod.GET)**

**public String showAddTodoPage(ModelMap model) {**

**model.addAttribute("todo", new Todo(0, (String) model.get("name"), "Default Desc",new Date(), false));**

**return "todo";**

**}**

@RequestMapping(value = "/delete-todo", method = RequestMethod.GET)

public String deleteTodo(@RequestParam int id) {

service.deleteTodo(id);

return "redirect:/list-todos";

}

@RequestMapping(value = "/add-todo", method = RequestMethod.POST)

**public String addTodo(ModelMap model, @Valid Todo todo, BindingResult result) {**

**if(result.hasErrors()){**

**return "todo";**

**}**

**service.addTodo((String) model.get("name"), todo.getDesc(), new Date(),**

**false);**

return "redirect:/list-todos";

}

}

src/main/java/com/in28minutes/springboot/web/model/Todo.java

package com.in28minutes.springboot.web.model;

import java.util.Date;

import javax.validation.constraints.Size;

public class Todo {

private int id;

private String user;

@Size(min=10, message="Enter at least 10 Characters...")

private String desc;

private Date targetDate;

private boolean isDone;

public Todo() {

super();

}

public Todo(int id, String user, String desc, Date targetDate,

boolean isDone) {

super();

this.id = id;

this.user = user;

this.desc = desc;

this.targetDate = targetDate;

this.isDone = isDone;

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getUser() {

return user;

}

public void setUser(String user) {

this.user = user;

}

public String getDesc() {

return desc;

}

public void setDesc(String desc) {

this.desc = desc;

}

public Date getTargetDate() {

return targetDate;

}

public void setTargetDate(Date targetDate) {

this.targetDate = targetDate;

}

public boolean isDone() {

return isDone;

}

public void setDone(boolean isDone) {

this.isDone = isDone;

}

@Override

public int hashCode() {

final int prime = 31;

int result = 1;

result = prime \* result + id;

return result;

}

@Override

public boolean equals(Object obj) {

if (this == obj) {

return true;

}

if (obj == null) {

return false;

}

if (getClass() != obj.getClass()) {

return false;

}

Todo other = (Todo) obj;

if (id != other.id) {

return false;

}

return true;

}

@Override

public String toString() {

return String.format(

"Todo [id=%s, user=%s, desc=%s, targetDate=%s, isDone=%s]", id,

user, desc, targetDate, isDone);

}

}

src/main/java/com/in28minutes/springboot/web/service/LoginService.java

package com.in28minutes.springboot.web.service;

import org.springframework.stereotype.Component;

import org.springframework.stereotype.Service;

@Service

public class LoginService {

public boolean validateUser(String userid, String password) {

// in28minutes, dummy

return userid.equalsIgnoreCase("in28minutes")

&& password.equalsIgnoreCase("dummy");

}

}

src/main/java/com/in28minutes/springboot/web/service/TodoService.java

package com.in28minutes.springboot.web.service;

import java.util.ArrayList;

import java.util.Date;

import java.util.Iterator;

import java.util.List;

import org.springframework.stereotype.Service;

import com.in28minutes.springboot.web.model.Todo;

@Service

public class TodoService {

private static List<Todo> todos = new ArrayList<Todo>();

private static int todoCount = 3;

static {

todos.add(new Todo(1, "in28Minutes", "Learn Spring MVC", new Date(),

false));

todos.add(new Todo(2, "in28Minutes", "Learn Struts", new Date(), false));

todos.add(new Todo(3, "in28Minutes", "Learn Hibernate", new Date(),

false));

}

public List<Todo> retrieveTodos(String user) {

List<Todo> filteredTodos = new ArrayList<Todo>();

for (Todo todo : todos) {

if (todo.getUser().equals(user)) {

filteredTodos.add(todo);

}

}

return filteredTodos;

}

public void addTodo(String name, String desc, Date targetDate,

boolean isDone) {

todos.add(new Todo(++todoCount, name, desc, targetDate, isDone));

}

public void deleteTodo(int id) {

Iterator<Todo> iterator = todos.iterator();

while (iterator.hasNext()) {

Todo todo = iterator.next();

if (todo.getId() == id) {

iterator.remove();

}

}

}

}

src/main/java/com/in28minutes/springboot/web/SpringBootFirstWebApplication.java

package com.in28minutes.springboot.web;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.annotation.ComponentScan;

@SpringBootApplication

@ComponentScan("com.in28minutes.springboot.web")

public class SpringBootFirstWebApplication {

public static void main(String[] args) {

SpringApplication.run(SpringBootFirstWebApplication.class, args);

}

}

**src/main/resources/application.properties**

spring.mvc.view.prefix=/WEB-INF/jsp/

spring.mvc.view.suffix=.jsp

logging.level.org.springframework.web=INFO

**src/main/webapp/WEB-INF/jsp/list-todos.jsp**

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>

<html>

<head>

<title>Todo's for ${name}</title>

<link href="webjars/bootstrap/3.3.6/css/bootstrap.min.css"

rel="stylesheet">

</head>

<body>

<div class="container">

<table class="table table-striped">

<caption>Your todos are</caption>

<thead>

<tr>

<th>Description</th>

<th>Target Date</th>

<th>Is it Done?</th>

<th></th>

</tr>

</thead>

<tbody>

<c:forEach items="${todos}" var="todo">

<tr>

<td>${todo.desc}</td>

<td>${todo.targetDate}</td>

<td>${todo.done}</td>

<td><a type="button" class="btn btn-warning"

href="/delete-todo?id=${todo.id}">Delete</a></td>

</tr>

</c:forEach>

</tbody>

</table>

<div>

<a class="button" href="/add-todo">Add a Todo</a>

</div>

</div>

<script src="webjars/jquery/1.9.1/jquery.min.js"></script>

<script src="webjars/bootstrap/3.3.6/js/bootstrap.min.js"></script>

</body>

</html>

**src/main/webapp/WEB-INF/jsp/login.jsp**

<html>

<head>

<title>First Web Application</title>

</head>

<body>

<font color="red">${errorMessage}</font>

<form method="post">

Name : <input type="text" name="name" />

Password : <input type="password" name="password" />

<input type="submit" />

</form>

</body>

</html>

**src/main/webapp/WEB-INF/jsp/todo.jsp**

<%@taglib uri="http://www.springframework.org/tags/form" prefix="form"%>

<html>

<head>

<title>First Web Application</title>

<link href="webjars/bootstrap/3.3.6/css/bootstrap.min.css" rel="stylesheet">

</head>

<body>

<div class="container">

<form:form method="post" commandName="todo">

<fieldset class="form-group">

<form:label path="desc">Description</form:label>

<form:input path="desc" type="text" class="form-control" required="required"/>

<form:errors path="desc" cssClass="text-warning"/>

</fieldset>

<button type="submit" class="btn btn-success">Add</button>

</form:form>

</div>

<script src="webjars/jquery/1.9.1/jquery.min.js"></script>

<script src="webjars/bootstrap/3.3.6/js/bootstrap.min.js"></script>

</body>

</html>

**src/main/webapp/WEB-INF/jsp/welcome.jsp**

<html>

<head>

<title>First Web Application</title>

</head>

<body>

Welcome ${name}!! <a href="/list-todos">Click here</a> to manage your todo's.

</body>

</html>

**src/test/java/com/in28minutes/springboot/web/SpringBootFirstWebApplicationTests.java**

package com.in28minutes.springboot.web;

import org.junit.Test;

import org.junit.runner.RunWith;

import org.springframework.boot.test.context.SpringBootTest;

import org.springframework.test.context.junit4.SpringRunner;

@RunWith(SpringRunner.class)

@SpringBootTest

public class SpringBootFirstWebApplicationTests {

@Test

public void contextLoads() {

}

}

**todo.txt**

Implementing Server Side Validation

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Command Bean or Form Backing Bean

Add Validation

Use Validation on Controller

Display Errors in View

Command Bean

~~~~~~~~~~~~

Controller

View - Spring Form Tags

LoginController -> adds name to model

welcome.jsp -> shows ${name}

TodoController -> redirects to list-todos.jsp

${name} is empty

Component, Service, Repository, Controller

Autowired

ComponentScan

Field dummyService in com.in28minutes.springboot.web.controller.LoginController

required a bean of type 'com.in28minutes.dummy.DummyService'

that could not be found.

Spring Boot Starter Parent

Spring Boot Starter Web

@SpringBootApplication

Auto Configuration

Dispatcher Servlet

/login => "login"

"login" => src/main/webapp/WEB-INF/jsp/login.jsp

Search for a view named "login"

/login => LoginController

**Here in this step we have just used commandBean or modelAttribute to bind with controller and view. Now in the next step we will use Hibernate Validator API.**

**Step18.md(part-2)**

**In the part-1 we have created commandBean or modelAttribute. The main advantage of this commandBean or form backing bean or modelAttribute is that , now we have double (two way binding), means we have bean which we are mapping to form(Bean->form) and we are form which we are mapping to bean(Form->Bean).**

|  |  |
| --- | --- |
|  |  |

**So here when we click on [Add a Todo] then then showAddTodoPage() method gets called and in this method we are setting todo bean with its default value and same todo bean we have mapped or bound in view with commandName todo as shown below. So what happens , todo object set in the showAddTodoPage will be available in the view with its default value and the default value will be mapped automatically to the corresponding input tag because in the both the places we are using same name (same object name(todo) and same variable name in model class and jsp page.**

|  |  |
| --- | --- |
|  |  |

**Means here the values are getting mapped from Bean to form (Bean-> Form), this is the first stage. Now the default set value will be shown in the input text as shown in the below image. Now we overwrite this default value (default Desc) with the new value(Learn Hibernate2) and click add so what would happen in the background?**

|  |  |
| --- | --- |
|  |  |

**In this case the form value will be bound to todo object defined in the showAddTodoPage() method. So this is nothing but form to bean (Form-> Bean) binding.**

**So in this way in the first step we have enabled the double two way binding.**

**Now let’s add validation to the bean.**

So far we have not added any validation to form and bean, so we can anything. Let’s say we have to add value at least of 6 length character. So here we will do the validation on the Model class Bean using Bean validation API.

Let’s go to Todo.java model class and add validation on description (desc) field.

|  |  |
| --- | --- |
|  |  |
| Here we have used @Size(min=10 , message=”Enter at least 10 Characters…). This, @Size is the part of java validation API in which we can specify min , max length of char and validation message to be displayed. | To work this validation completely we have to enable this validation in the controller along with the model class bean field.  So let’s go to controller and enable this validation in controller class also because the actual binding takes place in controller. To enable the validation in the controller on the Todo bean, we will add @Valid annotation followed by Todo todo in the method as shown in the image. |

So now we have added validation in the Model class field and enable the validation in the controller and to check whether the validation is working correct or not. To verify this we will use (**BindingResult result)**.

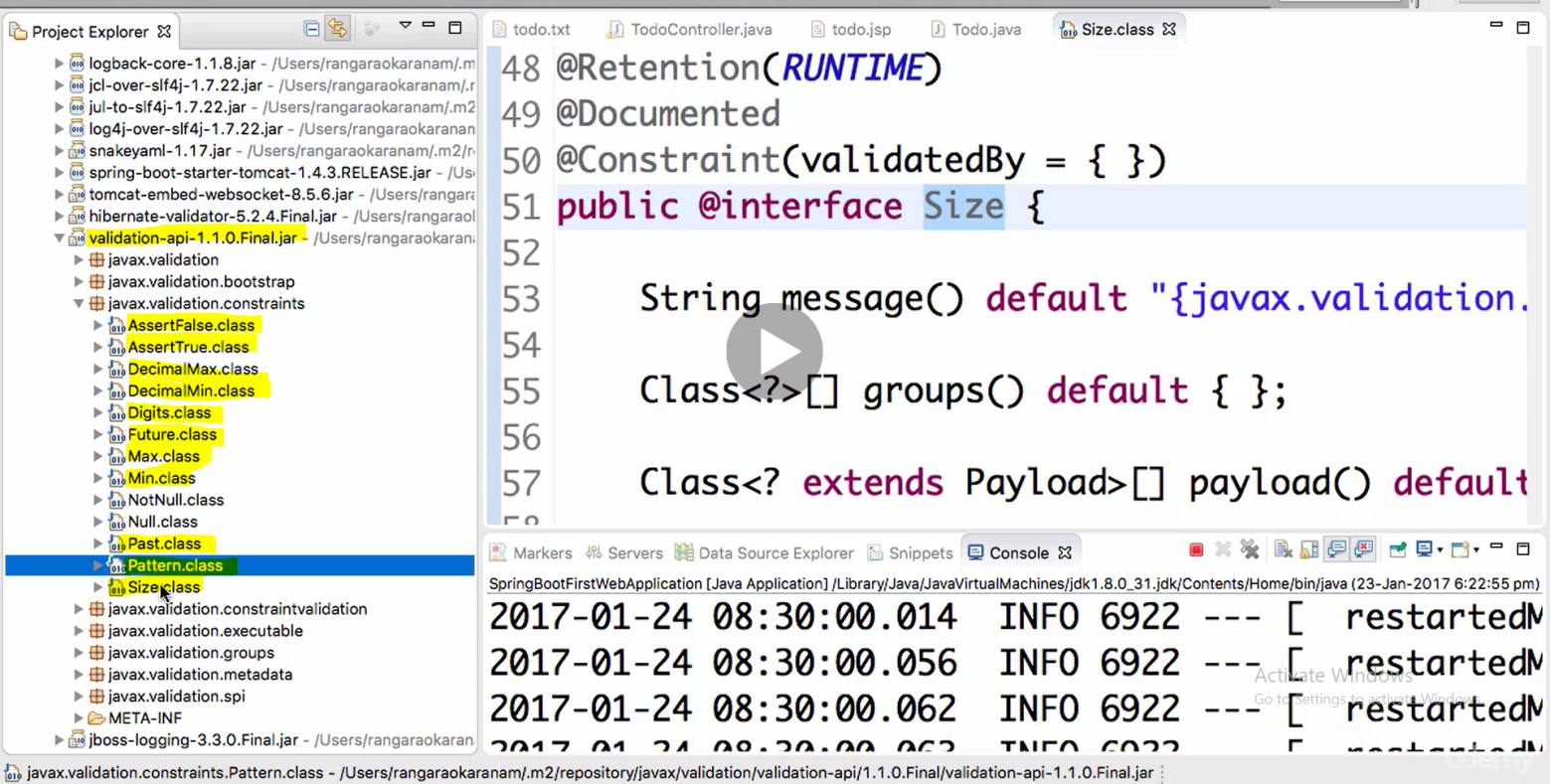
|  |  |
| --- | --- |
| So whenever we will use @Valid then along with this annotation we will have to use BindingResult result. So final code in the controller class is as given below. | @RequestMapping(value = "/add-todo", method = RequestMethod.POST)  **public String addTodo(ModelMap model, @Valid Todo todo, BindingResult result) {**  **if(result.hasErrors()){**  **return "todo";**  **}**  **service.addTodo((String) model.get("name"), todo.getDesc(), new Date(), false);**  return "redirect:/list-todos";  } |

|  |  |
| --- | --- |
| Now the next step is to show this error validation message in the view (jsp page).So spring provides tag  <form:errors **path =”desc”**  cssClass=”text-warning” / >.  Here text-warning is the bootstrap CSS class which is being to display error message as warning. | **<%@taglib uri="http://www.springframework.org/tags/form" prefix="form"%>**  **<html>**  **<head>**  **<title>First Web Application</title>**  **<link href="webjars/bootstrap/3.3.6/css/bootstrap.min.css" rel="stylesheet">**  **</head>**  **<body>**  **<div class="container">**  **<form:form method="post" commandName="todo">**  **<fieldset class="form-group">**  **<form:label path="desc">Description</form:label>**  **<form:input path="desc" type="text" class="form-control" required="required"/>**  **<form:errors path="desc" cssClass="text-warning"/>**  **</fieldset>**  **<button type="submit" class="btn btn-success">Add</button>**  **</form:form>**  **</div>**  **<script src="webjars/jquery/1.9.1/jquery.min.js"></script>**  **<script src="webjars/bootstrap/3.3.6/js/bootstrap.min.js"></script>**  **</body>**  **</html>** |

Now if enter the character less than 10 then we will get the error validation message like given below screen



We can use some other validation classes as per the given screen shot.



Like @Pattern for particular regular expression pattern, @Future & @Past for date, @Null & @NotNull , @DecimalMax , @DecimalMin etc.

So here we have follow the given step.

1. Add validation in model class bean field
2. Enable validation in the controller and use BindingResult to verify the validation
3. Add <form: errors> spring tag to display the error message for the particular field of the form.

**Step19.md**

[**https://github.com/in28minutes/spring-boot-master-class/blob/master/02.Spring-Boot-Web-Application/Step19.md**](https://github.com/in28minutes/spring-boot-master-class/blob/master/02.Spring-Boot-Web-Application/Step19.md)

1. **Add the update button in todo-list.jsp**
2. **Create a controller method to handle the update to show the update screen just like add desc screen. It means when we click on update button, a screen will be populated with current description field name which we have selected to update that and once update the todo we will see the updated todo list. So here in this step we will write one get request method to populate the current description in the update from.**
3. **Next we will write POST method to update the current description with new description and finally render the list-todo page with the updated description**

|  |
| --- |
| Added update button or link in list-todo.jsp |

What we will do:

* Add Update Functionality
* Lets Use the Same JSP as earlier.

Useful Snippets

public Todo retrieveTodo(int id) {

for (Todo todo : todos) {

if (todo.getId() == id)

return todo;

}

return null;

}

public void updateTodo(Todo todo) {

todos.remove(todo);

todos.add(todo);

}

todo.setUser("in28Minutes"); //TODO:Remove Hardcoding Later

service.updateTodo(todo);

<form:hidden path="id"/>

**Files List**

pom.xml

<?xml version="1.0" encoding="UTF-8"?>

<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">

<modelVersion>4.0.0</modelVersion>

<groupId>com.in28minutes.springboot.web</groupId>

<artifactId>spring-boot-first-web-application</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>spring-boot-first-web-application</name>

<description>Demo project for Spring Boot</description>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>1.4.3.RELEASE</version>

<relativePath /> <!-- lookup parent from repository -->

</parent>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>

<java.version>1.8</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>jstl</artifactId>

</dependency>

<dependency>

<groupId>org.webjars</groupId>

<artifactId>bootstrap</artifactId>

<version>3.3.6</version>

</dependency>

<dependency>

<groupId>org.webjars</groupId>

<artifactId>jquery</artifactId>

<version>1.9.1</version>

</dependency>

<dependency>

<groupId>org.apache.tomcat.embed</groupId>

<artifactId>tomcat-embed-jasper</artifactId>

<scope>provided</scope>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-devtools</artifactId>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

src/main/java/com/in28minutes/springboot/web/controller/LoginController.java

package com.in28minutes.springboot.web.controller;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Controller;

import org.springframework.ui.ModelMap;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestMethod;

import org.springframework.web.bind.annotation.RequestParam;

import org.springframework.web.bind.annotation.SessionAttributes;

import com.in28minutes.springboot.web.service.LoginService;

@Controller

@SessionAttributes("name")

public class LoginController {

@Autowired

LoginService service;

@RequestMapping(value="/login", method = RequestMethod.GET)

public String showLoginPage(ModelMap model){

return "login";

}

@RequestMapping(value="/login", method = RequestMethod.POST)

public String showWelcomePage(ModelMap model, @RequestParam String name, @RequestParam String password){

boolean isValidUser = service.validateUser(name, password);

if (!isValidUser) {

model.put("errorMessage", "Invalid Credentials");

return "login";

}

model.put("name", name);

model.put("password", password);

return "welcome";

}

}

src/main/java/com/in28minutes/springboot/web/controller/TodoController.java

package com.in28minutes.springboot.web.controller;

import java.util.Date;

import javax.validation.Valid;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Controller;

import org.springframework.ui.ModelMap;

import org.springframework.validation.BindingResult;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestMethod;

import org.springframework.web.bind.annotation.RequestParam;

import org.springframework.web.bind.annotation.SessionAttributes;

import com.in28minutes.springboot.web.model.Todo;

import com.in28minutes.springboot.web.service.LoginService;

import com.in28minutes.springboot.web.service.TodoService;

@Controller

@SessionAttributes("name")

public class TodoController {

@Autowired

TodoService service;

@RequestMapping(value = "/list-todos", method = RequestMethod.GET)

public String showTodos(ModelMap model) {

String name = (String) model.get("name");

model.put("todos", service.retrieveTodos(name));

return "list-todos";

}

@RequestMapping(value = "/add-todo", method = RequestMethod.GET)

public String showAddTodoPage(ModelMap model) {

model.addAttribute("todo", new Todo(0, (String) model.get("name"),

"Default Desc", new Date(), false));

return "todo";

}

@RequestMapping(value = "/delete-todo", method = RequestMethod.GET)

public String deleteTodo(@RequestParam int id) {

service.deleteTodo(id);

return "redirect:/list-todos";

}

@RequestMapping(value = "/update-todo", method = RequestMethod.GET)

public String showUpdateTodoPage(@RequestParam int id, ModelMap model) {

Todo todo = service.retrieveTodo(id);

model.put("todo", todo);

return "todo";

}

@RequestMapping(value = "/update-todo", method = RequestMethod.POST)

public String updateTodo(ModelMap model, @Valid Todo todo, BindingResult result) {

if (result.hasErrors()) {

return "todo";

}

todo.setUser((String) model.get("name"));

service.updateTodo(todo);

return "redirect:/list-todos";

}

@RequestMapping(value = "/add-todo", method = RequestMethod.POST)

public String addTodo(ModelMap model, @Valid Todo todo, BindingResult result) {

if (result.hasErrors()) {

return "todo";

}

service.addTodo((String) model.get("name"), todo.getDesc(), new Date(),

false);

return "redirect:/list-todos";

}

}

src/main/java/com/in28minutes/springboot/web/model/Todo.java

package com.in28minutes.springboot.web.model;

import java.util.Date;

import javax.validation.constraints.Size;

public class Todo {

private int id;

private String user;

@Size(min=10, message="Enter at least 10 Characters...")

private String desc;

private Date targetDate;

private boolean isDone;

public Todo() {

super();

}

public Todo(int id, String user, String desc, Date targetDate,

boolean isDone) {

super();

this.id = id;

this.user = user;

this.desc = desc;

this.targetDate = targetDate;

this.isDone = isDone;

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getUser() {

return user;

}

public void setUser(String user) {

this.user = user;

}

public String getDesc() {

return desc;

}

public void setDesc(String desc) {

this.desc = desc;

}

public Date getTargetDate() {

return targetDate;

}

public void setTargetDate(Date targetDate) {

this.targetDate = targetDate;

}

public boolean isDone() {

return isDone;

}

public void setDone(boolean isDone) {

this.isDone = isDone;

}

@Override

public int hashCode() {

final int prime = 31;

int result = 1;

result = prime \* result + id;

return result;

}

@Override

public boolean equals(Object obj) {

if (this == obj) {

return true;

}

if (obj == null) {

return false;

}

if (getClass() != obj.getClass()) {

return false;

}

Todo other = (Todo) obj;

if (id != other.id) {

return false;

}

return true;

}

@Override

public String toString() {

return String.format(

"Todo [id=%s, user=%s, desc=%s, targetDate=%s, isDone=%s]", id,

user, desc, targetDate, isDone);

}

}

src/main/java/com/in28minutes/springboot/web/service/LoginService.java

package com.in28minutes.springboot.web.service;

import org.springframework.stereotype.Component;

import org.springframework.stereotype.Service;

@Service

public class LoginService {

public boolean validateUser(String userid, String password) {

// in28minutes, dummy

return userid.equalsIgnoreCase("in28minutes")

&& password.equalsIgnoreCase("dummy");

}

}

src/main/java/com/in28minutes/springboot/web/service/TodoService.java

package com.in28minutes.springboot.web.service;

import java.util.ArrayList;

import java.util.Date;

import java.util.Iterator;

import java.util.List;

import org.springframework.stereotype.Service;

import com.in28minutes.springboot.web.model.Todo;

@Service

public class TodoService {

private static List<Todo> todos = new ArrayList<Todo>();

private static int todoCount = 3;

static {

todos.add(new Todo(1, "in28Minutes", "Learn Spring MVC", new Date(),

false));

todos.add(new Todo(2, "in28Minutes", "Learn Struts", new Date(), false));

todos.add(new Todo(3, "in28Minutes", "Learn Hibernate", new Date(),

false));

}

public List<Todo> retrieveTodos(String user) {

List<Todo> filteredTodos = new ArrayList<Todo>();

for (Todo todo : todos) {

if (todo.getUser().equalsIgnoreCase(user)) {

filteredTodos.add(todo);

}

}

return filteredTodos;

}

public Todo retrieveTodo(int id) {

for (Todo todo : todos) {

if (todo.getId()==id) {

return todo;

}

}

return null;

}

public void updateTodo(Todo todo){

todos.remove(todo);

todos.add(todo);

}

public void addTodo(String name, String desc, Date targetDate,

boolean isDone) {

todos.add(new Todo(++todoCount, name, desc, targetDate, isDone));

}

public void deleteTodo(int id) {

Iterator<Todo> iterator = todos.iterator();

while (iterator.hasNext()) {

Todo todo = iterator.next();

if (todo.getId() == id) {

iterator.remove();

}

}

}

}

src/main/java/com/in28minutes/springboot/web/SpringBootFirstWebApplication.java

package com.in28minutes.springboot.web;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.annotation.ComponentScan;

@SpringBootApplication

@ComponentScan("com.in28minutes.springboot.web")

public class SpringBootFirstWebApplication {

public static void main(String[] args) {

SpringApplication.run(SpringBootFirstWebApplication.class, args);

}

}

src/main/resources/application.properties

spring.mvc.view.prefix=/WEB-INF/jsp/

spring.mvc.view.suffix=.jsp

logging.level.org.springframework.web=INFO

src/main/webapp/WEB-INF/jsp/list-todos.jsp

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>

<html>

<head>

<title>Todo's for ${name}</title>

<link href="webjars/bootstrap/3.3.6/css/bootstrap.min.css"

rel="stylesheet">

</head>

<body>

<div class="container">

<table class="table table-striped">

<caption>Your todos are</caption>

<thead>

<tr>

<th>Description</th>

<th>Target Date</th>

<th>Is it Done?</th>

<th></th>

<th></th>

</tr>

</thead>

<tbody>

<c:forEach items="${todos}" var="todo">

<tr>

<td>${todo.desc}</td>

<td>${todo.targetDate}</td>

<td>${todo.done}</td>

<td><a type="button" class="btn btn-success"

href="/update-todo?id=${todo.id}">Update</a></td>

<td><a type="button" class="btn btn-warning"

href="/delete-todo?id=${todo.id}">Delete</a></td>

</tr>

</c:forEach>

</tbody>

</table>

<div>

<a class="button" href="/add-todo">Add a Todo</a>

</div>

</div>

<script src="webjars/jquery/1.9.1/jquery.min.js"></script>

<script src="webjars/bootstrap/3.3.6/js/bootstrap.min.js"></script>

</body>

</html>

src/main/webapp/WEB-INF/jsp/login.jsp

<html>

<head>

<title>First Web Application</title>

</head>

<body>

<font color="red">${errorMessage}</font>

<form method="post">

Name : <input type="text" name="name" />

Password : <input type="password" name="password" />

<input type="submit" />

</form>

</body>

</html>

src/main/webapp/WEB-INF/jsp/todo.jsp

<%@taglib uri="http://www.springframework.org/tags/form" prefix="form"%>

<html>

<head>

<title>First Web Application</title>

<link href="webjars/bootstrap/3.3.6/css/bootstrap.min.css"

rel="stylesheet">

</head>

<body>

<div class="container">

<form:form method="post" commandName="todo">

<form:hidden path="id"/>

<fieldset class="form-group">

<form:label path="desc">Description</form:label>

<form:input path="desc" type="text"

class="form-control" required="required"/>

<form:errors path="desc" cssClass="text-warning"/>

</fieldset>

<button type="submit" class="btn btn-success">Add</button>

</form:form>

</div>

<script src="webjars/jquery/1.9.1/jquery.min.js"></script>

<script src="webjars/bootstrap/3.3.6/js/bootstrap.min.js"></script>

</body>

</html>

src/main/webapp/WEB-INF/jsp/welcome.jsp

<html>

<head>

<title>First Web Application</title>

</head>

<body>

Welcome ${name}!! <a href="/list-todos">Click here</a> to manage your todo's.

</body>

</html>

src/test/java/com/in28minutes/springboot/web/SpringBootFirstWebApplicationTests.java

package com.in28minutes.springboot.web;

import org.junit.Test;

import org.junit.runner.RunWith;

import org.springframework.boot.test.context.SpringBootTest;

import org.springframework.test.context.junit4.SpringRunner;

@RunWith(SpringRunner.class)

@SpringBootTest

public class SpringBootFirstWebApplicationTests {

@Test

public void contextLoads() {

}

}

**todo.txt**

Implementing Server Side Validation

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Command Bean or Form Backing Bean

Add Validation

Use Validation on Controller

Display Errors in View

Command Bean

~~~~~~~~~~~~

Controller

View - Spring Form Tags

LoginController -> adds name to model

welcome.jsp -> shows ${name}

TodoController -> redirects to list-todos.jsp

${name} is empty

Component, Service, Repository, Controller

Autowired

ComponentScan

Field dummyService in com.in28minutes.springboot.web.controller.LoginController

required a bean of type 'com.in28minutes.dummy.DummyService'

that could not be found.

Spring Boot Starter Parent

Spring Boot Starter Web

@SpringBootApplication

Auto Configuration

Dispatcher Servlet

/login => "login"

"login" => src/main/webapp/WEB-INF/jsp/login.jsp

Search for a view named "login"

/login => LoginController

**Step20.md: Let’s add aTarget Date for Todo – use initBinder to handle Date Field**

[**https://github.com/in28minutes/spring-boot-master-class/blob/master/02.Spring-Boot-Web-Application/Step20.md**](https://github.com/in28minutes/spring-boot-master-class/blob/master/02.Spring-Boot-Web-Application/Step20.md)

What we will do:

* Make real use of the Target Date Field
* initBinder method

As of now in the update functionality to update only description, now here in this step we will add target date as well to update. So that we could be able edit and update description and target date.

Here the first thing is to remember that target date is a date (it’s not a string field like description), so there are few things that we need to setup before create and update target date.

Let’s add target date in the todo.jsp

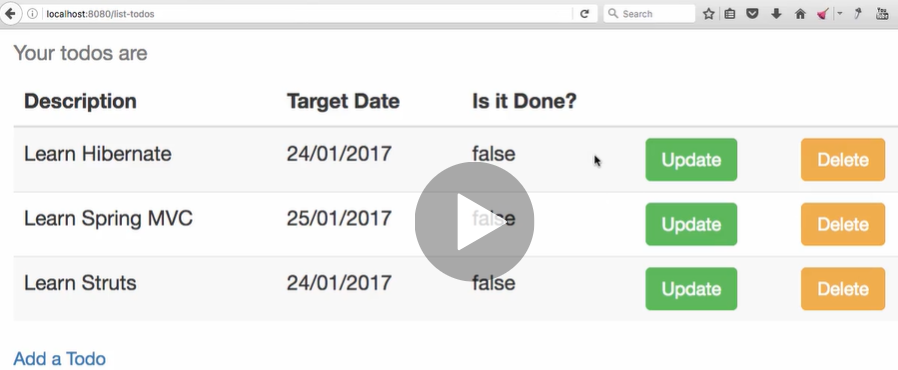
|  |
| --- |
| **src/main/webapp/WEB-INF/jsp/todo.jsp**  <%@taglib uri="http://www.springframework.org/tags/form" prefix="form"%>  <html>  <head>  <title>First Web Application</title>  <link href="webjars/bootstrap/3.3.6/css/bootstrap.min.css" rel="stylesheet">  </head>  <body>  <div class="container">  <form:form method="post" commandName="todo">  <form:hidden path="id" />  <fieldset class="form-group">  <form:label path="desc">Description</form:label>  <form:input path="desc" type="text" class="form-control"required="required" />  <form:errors path="desc" cssClass="text-warning" />  </fieldset>  **<fieldset class="form-group">**  **<form:label path="targetDate">Target Date</form:label>**  **<form:input path="targetDate" type="text" class="form-control"required="required" />**  **<form:errors path="targetDate" cssClass="text-warning" />**  **</fieldset>**  <button type="submit" class="btn btn-success">Add</button>  </form:form>  </div>  <script src="webjars/jquery/1.9.1/jquery.min.js"></script>  <script src="webjars/bootstrap/3.3.6/js/bootstrap.min.js"></script>  **<script**  **src="webjars/bootstrap-datepicker/1.0.1/js/bootstrap-datepicker.js"></script>**  **<script>**  **$('#targetDate').datepicker({format : 'dd/mm/yyyy'});**  **</script>**  </body>  </html> |
| Here we have added target date field set and java script date time piker. Now if we run the above code just after adding target field only then we will get the below error.    Once we click to add Description and target date then we got the above error ‘failed to convert date property’ because this is not in proper date format and in the model class it is defined as Date data type so we will have to format this date using the concept called initBinder , we have to tell to spring that , this is the format we are using across the application |

So the next step let’s add initBinder in the TodoContorller.java. @InitBinder basically initialize web data binder. So how does the InitBinder works? Like incase of String (like description) we can set string type data of any format only but in case of date, we have to set a particular format to the entire TodoControler, and this format setting is done by InitBinder. So here to set the particular format we have used SimpleDateFormat class to set date format and then registering this format to WebDataBinder as shown in the below yellow marked code.

|  |
| --- |
| **@InitBinder**  **public void initBinder(WebDataBinder binder) {**  **// Date - dd/MM/yyyy**  **SimpleDateFormat dateFormat = new SimpleDateFormat("dd/MM/yyyy");**  **binder.registerCustomEditor(Date.class, new CustomDateEditor(dateFormat, false));**  **}** |

|  |
| --- |
| src/main/java/com/in28minutes/springboot/web/controller/TodoController.java  package com.in28minutes.springboot.web.controller;  import java.text.SimpleDateFormat;  import java.util.Date;  import javax.validation.Valid;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.beans.propertyeditors.CustomDateEditor;  import org.springframework.stereotype.Controller;  import org.springframework.ui.ModelMap;  import org.springframework.validation.BindingResult;  import org.springframework.web.bind.WebDataBinder;  import org.springframework.web.bind.annotation.InitBinder;  import org.springframework.web.bind.annotation.RequestMapping;  import org.springframework.web.bind.annotation.RequestMethod;  import org.springframework.web.bind.annotation.RequestParam;  import org.springframework.web.bind.annotation.SessionAttributes;  import com.in28minutes.springboot.web.model.Todo;  import com.in28minutes.springboot.web.service.LoginService;  import com.in28minutes.springboot.web.service.TodoService;  @Controller  @SessionAttributes("name")  public class TodoController {  @Autowired  TodoService service;  **@InitBinder**  **public void initBinder(WebDataBinder binder) {**  **// Date - dd/MM/yyyy**  **SimpleDateFormat dateFormat = new SimpleDateFormat("dd/MM/yyyy");**  **binder.registerCustomEditor(Date.class, new CustomDateEditor(dateFormat, false));**  **}**  @RequestMapping(value = "/list-todos", method = RequestMethod.GET)  public String showTodos(ModelMap model) {  String name = (String) model.get("name");  model.put("todos", service.retrieveTodos(name));  return "list-todos";  }  @RequestMapping(value = "/add-todo", method = RequestMethod.GET)  public String showAddTodoPage(ModelMap model) {  model.addAttribute("todo", new Todo(0, (String) model.get("name"),  "Default Desc", new Date(), false));  return "todo";  }  @RequestMapping(value = "/delete-todo", method = RequestMethod.GET)  public String deleteTodo(@RequestParam int id) {  service.deleteTodo(id);  return "redirect:/list-todos";  }  @RequestMapping(value = "/update-todo", method = RequestMethod.GET)  public String showUpdateTodoPage(@RequestParam int id, ModelMap model) {  Todo todo = service.retrieveTodo(id);  model.put("todo", todo);  return "todo";  }  @RequestMapping(value = "/update-todo", method = RequestMethod.POST)  public String updateTodo(ModelMap model, @Valid Todo todo,  BindingResult result) {  if (result.hasErrors()) {  return "todo";  }  todo.setUser((String) model.get("name"));  service.updateTodo(todo);  return "redirect:/list-todos";  }  @RequestMapping(value = "/add-todo", method = RequestMethod.POST)  public String addTodo(ModelMap model, @Valid Todo todo, BindingResult result) {  if (result.hasErrors()) {  return "todo";  }  service.addTodo((String) model.get("name"), todo.getDesc(), **todo.getTargetDate(),**false);  return "redirect:/list-todos";  }  } |
| So now for Date.class use the above format (dd/mm/yyyy). Now if run the application again then we will get the date format like given below. (dd/mm/yyyy)   |  |  | | --- | --- | |  | Now let’s change the date and update it | | And now the date got updated but here in list-todo still date is not coming in dd/mm/yyyy format, so let’s display here also in dd/mm/yyyy format. Let’s go to list.todo.jsp  Here we will have to add jstl format tag library. **<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt"%>**  And then format the target date using **<td><fmt:formatDate value="${todo.targetDate}" pattern="dd/MM/yyyy"/></td>** | |   src/main/webapp/WEB-INF/jsp/list-todos.jsp  <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>  **<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt"%>**  <html>  <head>  <title>Todo's for ${name}</title>  <link href="webjars/bootstrap/3.3.6/css/bootstrap.min.css" rel="stylesheet">  </head>  <body>  <div class="container">  <table class="table table-striped">  <caption>Your todos are</caption>  <thead>  <tr>  <th>Description</th>  <th>Target Date</th>  <th>Is it Done?</th>  <th></th>  <th></th>  </tr>  </thead>  <tbody>  <c:forEach items="${todos}" var="todo">  <tr>  <td>${todo.desc}</td>  **<td><fmt:formatDate value="${todo.targetDate}" pattern="dd/MM/yyyy"/></td>**  <td>${todo.done}</td>  <td><a type="button" class="btn btn-success"  href="/update-todo?id=${todo.id}">Update</a></td>  <td><a type="button" class="btn btn-warning"  href="/delete-todo?id=${todo.id}">Delete</a></td>  </tr>  </c:forEach>  </tbody>  </table>  <div>  <a class="button" href="/add-todo">Add a Todo</a>  </div>  </div>  <script src="webjars/jquery/1.9.1/jquery.min.js"></script>  <script src="webjars/bootstrap/3.3.6/js/bootstrap.min.js"></script>  </body>  </html> |

Now if we run the above code then we will get the date in dd/mm/yyyy format only. One thing that we have to keep remember that here we have just jstl formatting tag library to format date, we have not initBinider to format date.



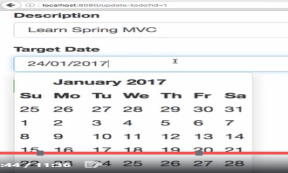
Useful Snippets

|  |
| --- |
| **JSP- snippet:**  <%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt"%>  <fmt:formatDate pattern="dd/MM/yyyy" value="${todo.targetDate}" />    **TodoController java snippet:**    @InitBinder  protected void initBinder(WebDataBinder binder) {  SimpleDateFormat dateFormat = new SimpleDateFormat("dd/MM/yyyy");  binder.registerCustomEditor(Date.class, new CustomDateEditor(  dateFormat, false));  }  **Pom.xml dependency added for datetimepiker snippet**:  <dependency>  <groupId>org.webjars</groupId>  <artifactId>bootstrap-datepicker</artifactId>  <version>1.0.1</version>  </dependency>  **Java Script snippet used in jsp where we have want date to be selected**  <script  src="webjars/bootstrap-datepicker/1.0.1/js/bootstrap-datepicker.js"></script>  <script>  $('#targetDate').datepicker({format : 'dd/mm/yyyy'});  </script> |

So far we have format the input date type as well as the date which being shown in the page after update. Now instead of typing date into date input box, if we want to select the date, then for that we have to use date piker to pick the date. Datepicker is part of java script which comes with the bootstrap, so to include datepiker we have to do two things.

|  |  |
| --- | --- |
| **First one is add dependency in Pom.xml**  **dependency>**  **<groupId>org.webjars</groupId>**  **<artifactId>bootstrap-datepicker</artifactId>**  **<version>1.0.1</version>**  **</dependency>**  This dependency is the part of webjars only. Once include this in pom, restart the server and then we will get this dependency in application and then we need to refer this file into jsp file as ginve into another cell **src=”…”** | Another one is use java script for the date input which we have to take as date picker to pick the date.  <script  **src="webjars/bootstrap-datepicker/1.0.1/js/bootstrap-**  **datepicker.js**"></script>  <script>  $('#**targetDate**').datepicker({format : 'dd/mm/yyyy'});  </script> |

Now the datepicker will look like as given below.



**Files List**

pom.xml

<?xml version="1.0" encoding="UTF-8"?>

<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">

<modelVersion>4.0.0</modelVersion>

<groupId>com.in28minutes.springboot.web</groupId>

<artifactId>spring-boot-first-web-application</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>spring-boot-first-web-application</name>

<description>Demo project for Spring Boot</description>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>1.4.3.RELEASE</version>

<relativePath /> <!-- lookup parent from repository -->

</parent>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>

<java.version>1.8</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>jstl</artifactId>

</dependency>

<dependency>

<groupId>org.webjars</groupId>

<artifactId>bootstrap</artifactId>

<version>3.3.6</version>

</dependency>

**<dependency>**

**<groupId>org.webjars</groupId>**

**<artifactId>bootstrap-datepicker</artifactId>**

**<version>1.0.1</version>**

**</dependency>**

<dependency>

<groupId>org.webjars</groupId>

<artifactId>jquery</artifactId>

<version>1.9.1</version>

</dependency>

<dependency>

<groupId>org.apache.tomcat.embed</groupId>

<artifactId>tomcat-embed-jasper</artifactId>

<scope>provided</scope>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-devtools</artifactId>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

src/main/java/com/in28minutes/springboot/web/controller/LoginController.java

package com.in28minutes.springboot.web.controller;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Controller;

import org.springframework.ui.ModelMap;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestMethod;

import org.springframework.web.bind.annotation.RequestParam;

import org.springframework.web.bind.annotation.SessionAttributes;

import com.in28minutes.springboot.web.service.LoginService;

@Controller

@SessionAttributes("name")

public class LoginController {

@Autowired

LoginService service;

@RequestMapping(value="/login", method = RequestMethod.GET)

public String showLoginPage(ModelMap model){

return "login";

}

@RequestMapping(value="/login", method = RequestMethod.POST)

public String showWelcomePage(ModelMap model, @RequestParam String name, @RequestParam String password){

boolean isValidUser = service.validateUser(name, password);

if (!isValidUser) {

model.put("errorMessage", "Invalid Credentials");

return "login";

}

model.put("name", name);

model.put("password", password);

return "welcome";

}

}

src/main/java/com/in28minutes/springboot/web/model/Todo.java

package com.in28minutes.springboot.web.model;

import java.util.Date;

import javax.validation.constraints.Size;

public class Todo {

private int id;

private String user;

@Size(min=10, message="Enter at least 10 Characters...")

private String desc;

private Date targetDate;

private boolean isDone;

public Todo() {

super();

}

public Todo(int id, String user, String desc, Date targetDate,

boolean isDone) {

super();

this.id = id;

this.user = user;

this.desc = desc;

this.targetDate = targetDate;

this.isDone = isDone;

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getUser() {

return user;

}

public void setUser(String user) {

this.user = user;

}

public String getDesc() {

return desc;

}

public void setDesc(String desc) {

this.desc = desc;

}

public Date getTargetDate() {

return targetDate;

}

public void setTargetDate(Date targetDate) {

this.targetDate = targetDate;

}

public boolean isDone() {

return isDone;

}

public void setDone(boolean isDone) {

this.isDone = isDone;

}

@Override

public int hashCode() {

final int prime = 31;

int result = 1;

result = prime \* result + id;

return result;

}

@Override

public boolean equals(Object obj) {

if (this == obj) {

return true;

}

if (obj == null) {

return false;

}

if (getClass() != obj.getClass()) {

return false;

}

Todo other = (Todo) obj;

if (id != other.id) {

return false;

}

return true;

}

@Override

public String toString() {

return String.format(

"Todo [id=%s, user=%s, desc=%s, targetDate=%s, isDone=%s]", id,

user, desc, targetDate, isDone);

}

}

src/main/java/com/in28minutes/springboot/web/service/LoginService.java

package com.in28minutes.springboot.web.service;

import org.springframework.stereotype.Component;

import org.springframework.stereotype.Service;

@Service

public class LoginService {

public boolean validateUser(String userid, String password) {

// in28minutes, dummy

return userid.equalsIgnoreCase("in28minutes")

&& password.equalsIgnoreCase("dummy");

}

}

src/main/java/com/in28minutes/springboot/web/service/TodoService.java

package com.in28minutes.springboot.web.service;

import java.util.ArrayList;

import java.util.Date;

import java.util.Iterator;

import java.util.List;

import org.springframework.stereotype.Service;

import com.in28minutes.springboot.web.model.Todo;

@Service

public class TodoService {

private static List<Todo> todos = new ArrayList<Todo>();

private static int todoCount = 3;

static {

todos.add(new Todo(1, "in28Minutes", "Learn Spring MVC", new Date(),

false));

todos.add(new Todo(2, "in28Minutes", "Learn Struts", new Date(), false));

todos.add(new Todo(3, "in28Minutes", "Learn Hibernate", new Date(),

false));

}

public List<Todo> retrieveTodos(String user) {

List<Todo> filteredTodos = new ArrayList<Todo>();

for (Todo todo : todos) {

if (todo.getUser().equalsIgnoreCase(user)) {

filteredTodos.add(todo);

}

}

return filteredTodos;

}

public Todo retrieveTodo(int id) {

for (Todo todo : todos) {

if (todo.getId()==id) {

return todo;

}

}

return null;

}

public void updateTodo(Todo todo){

todos.remove(todo);

todos.add(todo);

}

public void addTodo(String name, String desc, Date targetDate,

boolean isDone) {

todos.add(new Todo(++todoCount, name, desc, targetDate, isDone));

}

public void deleteTodo(int id) {

Iterator<Todo> iterator = todos.iterator();

while (iterator.hasNext()) {

Todo todo = iterator.next();

if (todo.getId() == id) {

iterator.remove();

}

}

}

}

src/main/java/com/in28minutes/springboot/web/SpringBootFirstWebApplication.java

package com.in28minutes.springboot.web;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.annotation.ComponentScan;

@SpringBootApplication

@ComponentScan("com.in28minutes.springboot.web")

public class SpringBootFirstWebApplication {

public static void main(String[] args) {

SpringApplication.run(SpringBootFirstWebApplication.class, args);

}

}

**src/main/resources/application.properties**

spring.mvc.view.prefix=/WEB-INF/jsp/

spring.mvc.view.suffix=.jsp

logging.level.org.springframework.web=INFO

src/main/webapp/WEB-INF/jsp/login.jsp

<html>

<head>

<title>First Web Application</title>

</head>

<body>

<font color="red">${errorMessage}</font>

<form method="post">

Name : <input type="text" name="name" />

Password : <input type="password" name="password" />

<input type="submit" />

</form>

</body>

</html>

src/main/webapp/WEB-INF/jsp/welcome.jsp

<html>

<head>

<title>First Web Application</title>

</head>

<body>

Welcome ${name}!! <a href="/list-todos">Click here</a> to manage your todo's.

</body>

</html>

src/test/java/com/in28minutes/springboot/web/SpringBootFirstWebApplicationTests.java

package com.in28minutes.springboot.web;

import org.junit.Test;

import org.junit.runner.RunWith;

import org.springframework.boot.test.context.SpringBootTest;

import org.springframework.test.context.junit4.SpringRunner;

@RunWith(SpringRunner.class)

@SpringBootTest

public class SpringBootFirstWebApplicationTests {

@Test

public void contextLoads() {

}

}

**todo.txt**

Implementing Server Side Validation

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Command Bean or Form Backing Bean

Add Validation

Use Validation on Controller

Display Errors in View

Command Bean

~~~~~~~~~~~~

Controller

View - Spring Form Tags

LoginController -> adds name to model

welcome.jsp -> shows ${name}

TodoController -> redirects to list-todos.jsp

${name} is empty

Component, Service, Repository, Controller

Autowired

ComponentScan

Field dummyService in com.in28minutes.springboot.web.controller.LoginController

required a bean of type 'com.in28minutes.dummy.DummyService'

that could not be found.

Spring Boot Starter Parent

Spring Boot Starter Web

@SpringBootApplication

Auto Configuration

Dispatcher Servlet

/login => "login"

"login" => src/main/webapp/WEB-INF/jsp/login.jsp

Search for a view named "login"

/login => LoginController

**Step21.md- jsp fragments and Navigation bar**

[**https://github.com/in28minutes/spring-boot-master-class/blob/master/02.Spring-Boot-Web-Application/Step21.md**](https://github.com/in28minutes/spring-boot-master-class/blob/master/02.Spring-Boot-Web-Application/Step21.md)

What we will do:

* Add a navigation bar
* Use JSP Fragments
* Exercise : Align the login & welcome pages.
* Exercise : Highlight the correct menu item.

Useful Snippets

<nav role="navigation" class="navbar navbar-default">

<div class="">

<a href="http://www.in28minutes.com" class="navbar-brand">in28Minutes</a>

</div>

<div class="navbar-collapse">

<ul class="nav navbar-nav">

<li class="active"><a href="/login">Home</a></li>

<li><a href="/list-todos">Todos</a></li>

</ul>

</div>

</nav>

**Files List**

**pom.xml**

<?xml version="1.0" encoding="UTF-8"?>

<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">

<modelVersion>4.0.0</modelVersion>

<groupId>com.in28minutes.springboot.web</groupId>

<artifactId>spring-boot-first-web-application</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>spring-boot-first-web-application</name>

<description>Demo project for Spring Boot</description>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>1.4.3.RELEASE</version>

<relativePath /> <!-- lookup parent from repository -->

</parent>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>

<java.version>1.8</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>jstl</artifactId>

</dependency>

<dependency>

<groupId>org.webjars</groupId>

<artifactId>bootstrap</artifactId>

<version>3.3.6</version>

</dependency>

<dependency>

<groupId>org.webjars</groupId>

<artifactId>bootstrap-datepicker</artifactId>

<version>1.0.1</version>

</dependency>

<dependency>

<groupId>org.webjars</groupId>

<artifactId>jquery</artifactId>

<version>1.9.1</version>

</dependency>

<dependency>

<groupId>org.apache.tomcat.embed</groupId>

<artifactId>tomcat-embed-jasper</artifactId>

<scope>provided</scope>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-devtools</artifactId>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

src/main/java/com/in28minutes/springboot/web/controller/LoginController.java

package com.in28minutes.springboot.web.controller;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Controller;

import org.springframework.ui.ModelMap;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestMethod;

import org.springframework.web.bind.annotation.RequestParam;

import org.springframework.web.bind.annotation.SessionAttributes;

import com.in28minutes.springboot.web.service.LoginService;

@Controller

@SessionAttributes("name")

public class LoginController {

@Autowired

LoginService service;

@RequestMapping(value="/login", method = RequestMethod.GET)

public String showLoginPage(ModelMap model){

return "login";

}

@RequestMapping(value="/login", method = RequestMethod.POST)

public String showWelcomePage(ModelMap model, @RequestParam String name, @RequestParam String password){

boolean isValidUser = service.validateUser(name, password);

if (!isValidUser) {

model.put("errorMessage", "Invalid Credentials");

return "login";

}

model.put("name", name);

model.put("password", password);

return "welcome";

}

}

src/main/java/com/in28minutes/springboot/web/controller/TodoController.java

package com.in28minutes.springboot.web.controller;

import java.text.SimpleDateFormat;

import java.util.Date;

import javax.validation.Valid;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.beans.propertyeditors.CustomDateEditor;

import org.springframework.stereotype.Controller;

import org.springframework.ui.ModelMap;

import org.springframework.validation.BindingResult;

import org.springframework.web.bind.WebDataBinder;

import org.springframework.web.bind.annotation.InitBinder;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestMethod;

import org.springframework.web.bind.annotation.RequestParam;

import org.springframework.web.bind.annotation.SessionAttributes;

import com.in28minutes.springboot.web.model.Todo;

import com.in28minutes.springboot.web.service.LoginService;

import com.in28minutes.springboot.web.service.TodoService;

@Controller

@SessionAttributes("name")

public class TodoController {

@Autowired

TodoService service;

@InitBinder

public void initBinder(WebDataBinder binder) {

// Date - dd/MM/yyyy

SimpleDateFormat dateFormat = new SimpleDateFormat("dd/MM/yyyy");

binder.registerCustomEditor(Date.class, new CustomDateEditor(

dateFormat, false));

}

@RequestMapping(value = "/list-todos", method = RequestMethod.GET)

public String showTodos(ModelMap model) {

String name = (String) model.get("name");

model.put("todos", service.retrieveTodos(name));

return "list-todos";

}

@RequestMapping(value = "/add-todo", method = RequestMethod.GET)

public String showAddTodoPage(ModelMap model) {

model.addAttribute("todo", new Todo(0, (String) model.get("name"),

"Default Desc", new Date(), false));

return "todo";

}

@RequestMapping(value = "/delete-todo", method = RequestMethod.GET)

public String deleteTodo(@RequestParam int id) {

service.deleteTodo(id);

return "redirect:/list-todos";

}

@RequestMapping(value = "/update-todo", method = RequestMethod.GET)

public String showUpdateTodoPage(@RequestParam int id, ModelMap model) {

Todo todo = service.retrieveTodo(id);

model.put("todo", todo);

return "todo";

}

@RequestMapping(value = "/update-todo", method = RequestMethod.POST)

public String updateTodo(ModelMap model, @Valid Todo todo,

BindingResult result) {

if (result.hasErrors()) {

return "todo";

}

todo.setUser((String) model.get("name"));

service.updateTodo(todo);

return "redirect:/list-todos";

}

@RequestMapping(value = "/add-todo", method = RequestMethod.POST)

public String addTodo(ModelMap model, @Valid Todo todo, BindingResult result) {

if (result.hasErrors()) {

return "todo";

}

service.addTodo((String) model.get("name"), todo.getDesc(), todo.getTargetDate(),

false);

return "redirect:/list-todos";

}

}

src/main/java/com/in28minutes/springboot/web/model/Todo.java

package com.in28minutes.springboot.web.model;

import java.util.Date;

import javax.validation.constraints.Size;

public class Todo {

private int id;

private String user;

@Size(min=10, message="Enter at least 10 Characters...")

private String desc;

private Date targetDate;

private boolean isDone;

public Todo() {

super();

}

public Todo(int id, String user, String desc, Date targetDate,

boolean isDone) {

super();

this.id = id;

this.user = user;

this.desc = desc;

this.targetDate = targetDate;

this.isDone = isDone;

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getUser() {

return user;

}

public void setUser(String user) {

this.user = user;

}

public String getDesc() {

return desc;

}

public void setDesc(String desc) {

this.desc = desc;

}

public Date getTargetDate() {

return targetDate;

}

public void setTargetDate(Date targetDate) {

this.targetDate = targetDate;

}

public boolean isDone() {

return isDone;

}

public void setDone(boolean isDone) {

this.isDone = isDone;

}

@Override

public int hashCode() {

final int prime = 31;

int result = 1;

result = prime \* result + id;

return result;

}

@Override

public boolean equals(Object obj) {

if (this == obj) {

return true;

}

if (obj == null) {

return false;

}

if (getClass() != obj.getClass()) {

return false;

}

Todo other = (Todo) obj;

if (id != other.id) {

return false;

}

return true;

}

@Override

public String toString() {

return String.format(

"Todo [id=%s, user=%s, desc=%s, targetDate=%s, isDone=%s]", id,

user, desc, targetDate, isDone);

}

}

src/main/java/com/in28minutes/springboot/web/service/LoginService.java

package com.in28minutes.springboot.web.service;

import org.springframework.stereotype.Component;

import org.springframework.stereotype.Service;

@Service

public class LoginService {

public boolean validateUser(String userid, String password) {

// in28minutes, dummy

return userid.equalsIgnoreCase("in28minutes")

&& password.equalsIgnoreCase("dummy");

}

}

src/main/java/com/in28minutes/springboot/web/service/TodoService.java

package com.in28minutes.springboot.web.service;

import java.util.ArrayList;

import java.util.Date;

import java.util.Iterator;

import java.util.List;

import org.springframework.stereotype.Service;

import com.in28minutes.springboot.web.model.Todo;

@Service

public class TodoService {

private static List<Todo> todos = new ArrayList<Todo>();

private static int todoCount = 3;

static {

todos.add(new Todo(1, "in28Minutes", "Learn Spring MVC", new Date(),

false));

todos.add(new Todo(2, "in28Minutes", "Learn Struts", new Date(), false));

todos.add(new Todo(3, "in28Minutes", "Learn Hibernate", new Date(),

false));

}

public List<Todo> retrieveTodos(String user) {

List<Todo> filteredTodos = new ArrayList<Todo>();

for (Todo todo : todos) {

if (todo.getUser().equalsIgnoreCase(user)) {

filteredTodos.add(todo);

}

}

return filteredTodos;

}

public Todo retrieveTodo(int id) {

for (Todo todo : todos) {

if (todo.getId()==id) {

return todo;

}

}

return null;

}

public void updateTodo(Todo todo){

todos.remove(todo);

todos.add(todo);

}

public void addTodo(String name, String desc, Date targetDate,

boolean isDone) {

todos.add(new Todo(++todoCount, name, desc, targetDate, isDone));

}

public void deleteTodo(int id) {

Iterator<Todo> iterator = todos.iterator();

while (iterator.hasNext()) {

Todo todo = iterator.next();

if (todo.getId() == id) {

iterator.remove();

}

}

}

}

src/main/java/com/in28minutes/springboot/web/SpringBootFirstWebApplication.java

package com.in28minutes.springboot.web;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.annotation.ComponentScan;

@SpringBootApplication

@ComponentScan("com.in28minutes.springboot.web")

public class SpringBootFirstWebApplication {

public static void main(String[] args) {

SpringApplication.run(SpringBootFirstWebApplication.class, args);

}

}

src/main/resources/application.properties

spring.mvc.view.prefix=/WEB-INF/jsp/

spring.mvc.view.suffix=.jsp

logging.level.org.springframework.web=INFO

src/main/webapp/WEB-INF/jsp/common/**footer.jspf**

<script src="webjars/jquery/1.9.1/jquery.min.js"></script>

<script src="webjars/bootstrap/3.3.6/js/bootstrap.min.js"></script>

<script

src="webjars/bootstrap-datepicker/1.0.1/js/bootstrap-datepicker.js"></script>

<script>

$('#targetDate').datepicker({

format : 'dd/mm/yyyy'

});

</script>

</body>

</html>

src/main/webapp/WEB-INF/jsp/common/header.jspf

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>

<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt"%>

<%@taglib uri="http://www.springframework.org/tags/form" prefix="form"%>

<html>

<head>

<title>First Web Application</title>

<link href="webjars/bootstrap/3.3.6/css/bootstrap.min.css"

rel="stylesheet">

</head>

<body>

src/main/webapp/WEB-INF/jsp/common/navigation.jspf

<nav role="navigation" class="navbar navbar-default">

<div class="">

<a href="http://www.in28minutes.com" class="navbar-brand">in28Minutes</a>

</div>

<div class="navbar-collapse">

<ul class="nav navbar-nav">

<li class="active"><a href="/login">Home</a></li>

<li><a href="/list-todos">Todos</a></li>

</ul>

</div>

</nav>

src/main/webapp/WEB-INF/jsp/list-todos.jsp

**<%@ include file="common/header.jspf" %>**

**<%@ include file="common/navigation.jspf" %>**

<div class="container">

<table class="table table-striped">

<caption>Your todos are</caption>

<thead>

<tr>

<th>Description</th>

<th>Target Date</th>

<th>Is it Done?</th>

<th></th>

<th></th>

</tr>

</thead>

<tbody>

<c:forEach items="${todos}" var="todo">

<tr>

<td>${todo.desc}</td>

<td><fmt:formatDate value="${todo.targetDate}" pattern="dd/MM/yyyy"/></td>

<td>${todo.done}</td>

<td><a type="button" class="btn btn-success"

href="/update-todo?id=${todo.id}">Update</a></td>

<td><a type="button" class="btn btn-warning"

href="/delete-todo?id=${todo.id}">Delete</a></td>

</tr>

</c:forEach>

</tbody>

</table>

<div>

<a class="button" href="/add-todo">Add a Todo</a>

</div>

</div>

**<%@ include file="common/footer.jspf" %>**

src/main/webapp/WEB-INF/jsp/login.jsp

**<%@ include file="common/header.jspf" %>**

**<%@ include file="common/navigation.jspf" %>**

<div class="container">

<font color="red">${errorMessage}</font>

<form method="post">

Name : <input type="text" name="name" />

Password : <input type="password" name="password" />

<input type="submit" />

</form>

</div>

**<%@ include file="common/footer.jspf" %>**

**src/main/webapp/WEB-INF/jsp/todo.jsp**

**<%@ include file="common/header.jspf" %>**

**<%@ include file="common/navigation.jspf" %>**

<div class="container">

<form:form method="post" commandName="todo">

<form:hidden path="id" />

<fieldset class="form-group">

<form:label path="desc">Description</form:label>

<form:input path="desc" type="text" class="form-control"

required="required" />

<form:errors path="desc" cssClass="text-warning" />

</fieldset>

<fieldset class="form-group">

<form:label path="targetDate">Target Date</form:label>

<form:input path="targetDate" type="text" class="form-control"

required="required" />

<form:errors path="targetDate" cssClass="text-warning" />

</fieldset>

<button type="submit" class="btn btn-success">Add</button>

</form:form>

</div>

**<%@ include file="common/footer.jspf" %>**

src/main/webapp/WEB-INF/jsp/welcome.jsp

**<%@ include file="common/header.jspf"%>**

**<%@ include file="common/navigation.jspf"%>**

<div class="container">

Welcome ${name}!! <a href="/list-todos">Click here</a> to manage your

todo's.

</div>

**<%@ include file="common/footer.jspf"%>**

src/test/java/com/in28minutes/springboot/web/SpringBootFirstWebApplicationTests.java

package com.in28minutes.springboot.web;

import org.junit.Test;

import org.junit.runner.RunWith;

import org.springframework.boot.test.context.SpringBootTest;

import org.springframework.test.context.junit4.SpringRunner;

@RunWith(SpringRunner.class)

@SpringBootTest

public class SpringBootFirstWebApplicationTests {

@Test

public void contextLoads() {

}

}

todo.txt

Implementing Server Side Validation

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Command Bean or Form Backing Bean

Add Validation

Use Validation on Controller

Display Errors in View

Command Bean

~~~~~~~~~~~~

Controller

View - Spring Form Tags

LoginController -> adds name to model

welcome.jsp -> shows ${name}

TodoController -> redirects to list-todos.jsp

${name} is empty

Component, Service, Repository, Controller

Autowired

ComponentScan

Field dummyService in com.in28minutes.springboot.web.controller.LoginController

required a bean of type 'com.in28minutes.dummy.DummyService'

that could not be found.

Spring Boot Starter Parent

Spring Boot Starter Web

@SpringBootApplication

Auto Configuration

Dispatcher Servlet

/login => "login"

"login" => src/main/webapp/WEB-INF/jsp/login.jsp

Search for a view named "login"

/login => LoginController

**Step22.md: Preparing for Spring Security**

<https://github.com/in28minutes/spring-boot-master-class/blob/master/02.Spring-Boot-Web-Application/Step22.md>

With spring security we can add secured features in entire application. If we look at the biggest problem that we have right now, someone can type <http://localhost:8080/add-todo> and can open this page or any of the directly without login into application. This is not secure way of doing these things.

So if we try to open any page directly, it should direct to automatically to login page and ask userid and password to login and then we can open any of the pages of the application.

The framework which allows this kind of functionality is called Spring Security. Using spring security we can secure entire application and the only persone who logged in can only see the entire application.

Generally in typical web application only logged in user only can see or do other functionality of the application, same thing the spring security does.

One of the things that spring security provides us is the secured default login page. Here we will use that default login page to login and in that case we don’t need our own created login page at all. So we will delete our login page.

Note: Here we are retrieving the name from session in different places, so instead of retrieving using Model, we will convert this particular reubale line of code into one reuabale method, and where ever required, we will just call this particular method. To convert this particular reusable lines of code into reusable method follow the steps:

1. We will have to select that particular line of code
2. Right click on the selected reusable line of code
3. Go to Refactor - > Click on Extract Method
4. Give the method name and select the number of occurances > click Ok

|  |  |
| --- | --- |
|  | |
|  | And now we have got the corresponding reusable Refactor method. |

And now this particular method will be replaced with the particular reusable line of code in all the places (occuurances).

**What we will do:**

* Prepare for Using Spring Security
* Remove All the Login Related Functionality
* Make Welcome the default page - with some hardcoding to start with.
* Refactor getLoggedInUserName
* Update Home Page Link in navigation

**Files List**

**pom.xml**

<?xml version="1.0" encoding="UTF-8"?>

<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">

<modelVersion>4.0.0</modelVersion>

<groupId>com.in28minutes.springboot.web</groupId>

<artifactId>spring-boot-first-web-application</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>spring-boot-first-web-application</name>

<description>Demo project for Spring Boot</description>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>1.4.3.RELEASE</version>

<relativePath /> <!-- lookup parent from repository -->

</parent>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>

<java.version>1.8</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>jstl</artifactId>

</dependency>

<dependency>

<groupId>org.webjars</groupId>

<artifactId>bootstrap</artifactId>

<version>3.3.6</version>

</dependency>

<dependency>

<groupId>org.webjars</groupId>

<artifactId>bootstrap-datepicker</artifactId>

<version>1.0.1</version>

</dependency>

<dependency>

<groupId>org.webjars</groupId>

<artifactId>jquery</artifactId>

<version>1.9.1</version>

</dependency>

<dependency>

<groupId>org.apache.tomcat.embed</groupId>

<artifactId>tomcat-embed-jasper</artifactId>

<scope>provided</scope>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-devtools</artifactId>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

**src/main/java/com/in28minutes/springboot/web/controller/LoginController.java**

package com.in28minutes.springboot.web.controller;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Controller;

import org.springframework.ui.ModelMap;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestMethod;

import org.springframework.web.bind.annotation.SessionAttributes;

import com.in28minutes.springboot.web.service.LoginService;

@Controller

@SessionAttributes("name")

public class LoginController {

@Autowired

LoginService service;

@RequestMapping(value = "/", method = RequestMethod.GET)

public String showLoginPage(ModelMap model) {

model.put("name", "in28Minutes");

return "welcome";

}

}

**src/main/java/com/in28minutes/springboot/web/controller/TodoController.java**

package com.in28minutes.springboot.web.controller;

import java.text.SimpleDateFormat;

import java.util.Date;

import javax.validation.Valid;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.beans.propertyeditors.CustomDateEditor;

import org.springframework.stereotype.Controller;

import org.springframework.ui.ModelMap;

import org.springframework.validation.BindingResult;

import org.springframework.web.bind.WebDataBinder;

import org.springframework.web.bind.annotation.InitBinder;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestMethod;

import org.springframework.web.bind.annotation.RequestParam;

import org.springframework.web.bind.annotation.SessionAttributes;

import com.in28minutes.springboot.web.model.Todo;

import com.in28minutes.springboot.web.service.LoginService;

import com.in28minutes.springboot.web.service.TodoService;

@Controller

@SessionAttributes("name")

public class TodoController {

@Autowired

TodoService service;

@InitBinder

public void initBinder(WebDataBinder binder) {

// Date - dd/MM/yyyy

SimpleDateFormat dateFormat = new SimpleDateFormat("dd/MM/yyyy");

binder.registerCustomEditor(Date.class, new CustomDateEditor(

dateFormat, false));

}

@RequestMapping(value = "/list-todos", method = RequestMethod.GET)

public String showTodos(ModelMap model) {

String name = getLoggedInUserName(model);

model.put("todos", service.retrieveTodos(name));

return "list-todos";

}

private String getLoggedInUserName(ModelMap model) {

return (String) model.get("name");

}

@RequestMapping(value = "/add-todo", method = RequestMethod.GET)

public String showAddTodoPage(ModelMap model) {

model.addAttribute("todo", new Todo(0, getLoggedInUserName(model),

"Default Desc", new Date(), false));

return "todo";

}

@RequestMapping(value = "/delete-todo", method = RequestMethod.GET)

public String deleteTodo(@RequestParam int id) {

service.deleteTodo(id);

return "redirect:/list-todos";

}

@RequestMapping(value = "/update-todo", method = RequestMethod.GET)

public String showUpdateTodoPage(@RequestParam int id, ModelMap model) {

Todo todo = service.retrieveTodo(id);

model.put("todo", todo);

return "todo";

}

@RequestMapping(value = "/update-todo", method = RequestMethod.POST)

public String updateTodo(ModelMap model, @Valid Todo todo,

BindingResult result) {

if (result.hasErrors()) {

return "todo";

}

todo.setUser(getLoggedInUserName(model));

service.updateTodo(todo);

return "redirect:/list-todos";

}

@RequestMapping(value = "/add-todo", method = RequestMethod.POST)

public String addTodo(ModelMap model, @Valid Todo todo, BindingResult result) {

if (result.hasErrors()) {

return "todo";

}

service.addTodo(getLoggedInUserName(model), todo.getDesc(), todo.getTargetDate(),

false);

return "redirect:/list-todos";

}

}

**src/main/java/com/in28minutes/springboot/web/model/Todo.java**

package com.in28minutes.springboot.web.model;

import java.util.Date;

import javax.validation.constraints.Size;

public class Todo {

private int id;

private String user;

@Size(min=10, message="Enter at least 10 Characters...")

private String desc;

private Date targetDate;

private boolean isDone;

public Todo() {

super();

}

public Todo(int id, String user, String desc, Date targetDate,

boolean isDone) {

super();

this.id = id;

this.user = user;

this.desc = desc;

this.targetDate = targetDate;

this.isDone = isDone;

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getUser() {

return user;

}

public void setUser(String user) {

this.user = user;

}

public String getDesc() {

return desc;

}

public void setDesc(String desc) {

this.desc = desc;

}

public Date getTargetDate() {

return targetDate;

}

public void setTargetDate(Date targetDate) {

this.targetDate = targetDate;

}

public boolean isDone() {

return isDone;

}

public void setDone(boolean isDone) {

this.isDone = isDone;

}

@Override

public int hashCode() {

final int prime = 31;

int result = 1;

result = prime \* result + id;

return result;

}

@Override

public boolean equals(Object obj) {

if (this == obj) {

return true;

}

if (obj == null) {

return false;

}

if (getClass() != obj.getClass()) {

return false;

}

Todo other = (Todo) obj;

if (id != other.id) {

return false;

}

return true;

}

@Override

public String toString() {

return String.format(

"Todo [id=%s, user=%s, desc=%s, targetDate=%s, isDone=%s]", id,

user, desc, targetDate, isDone);

}

}

**src/main/java/com/in28minutes/springboot/web/service/LoginService.java**

package com.in28minutes.springboot.web.service;

import org.springframework.stereotype.Component;

import org.springframework.stereotype.Service;

@Service

public class LoginService {

public boolean validateUser(String userid, String password) {

// in28minutes, dummy

return userid.equalsIgnoreCase("in28minutes")

&& password.equalsIgnoreCase("dummy");

}

}

**src/main/java/com/in28minutes/springboot/web/service/TodoService.java**

package com.in28minutes.springboot.web.service;

import java.util.ArrayList;

import java.util.Date;

import java.util.Iterator;

import java.util.List;

import org.springframework.stereotype.Service;

import com.in28minutes.springboot.web.model.Todo;

@Service

public class TodoService {

private static List<Todo> todos = new ArrayList<Todo>();

private static int todoCount = 3;

static {

todos.add(new Todo(1, "in28Minutes", "Learn Spring MVC", new Date(),

false));

todos.add(new Todo(2, "in28Minutes", "Learn Struts", new Date(), false));

todos.add(new Todo(3, "in28Minutes", "Learn Hibernate", new Date(),

false));

}

public List<Todo> retrieveTodos(String user) {

List<Todo> filteredTodos = new ArrayList<Todo>();

for (Todo todo : todos) {

if (todo.getUser().equalsIgnoreCase(user)) {

filteredTodos.add(todo);

}

}

return filteredTodos;

}

public Todo retrieveTodo(int id) {

for (Todo todo : todos) {

if (todo.getId()==id) {

return todo;

}

}

return null;

}

public void updateTodo(Todo todo){

todos.remove(todo);

todos.add(todo);

}

public void addTodo(String name, String desc, Date targetDate,

boolean isDone) {

todos.add(new Todo(++todoCount, name, desc, targetDate, isDone));

}

public void deleteTodo(int id) {

Iterator<Todo> iterator = todos.iterator();

while (iterator.hasNext()) {

Todo todo = iterator.next();

if (todo.getId() == id) {

iterator.remove();

}

}

}

}

**src/main/java/com/in28minutes/springboot/web/SpringBootFirstWebApplication.java**

package com.in28minutes.springboot.web;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.annotation.ComponentScan;

@SpringBootApplication

@ComponentScan("com.in28minutes.springboot.web")

public class SpringBootFirstWebApplication {

public static void main(String[] args) {

SpringApplication.run(SpringBootFirstWebApplication.class, args);

}

}

**src/main/resources/application.properties**

spring.mvc.view.prefix=/WEB-INF/jsp/

spring.mvc.view.suffix=.jsp

logging.level.org.springframework.web=INFO

**src/main/webapp/WEB-INF/jsp/common/footer.jspf**

<script src="webjars/jquery/1.9.1/jquery.min.js"></script>

<script src="webjars/bootstrap/3.3.6/js/bootstrap.min.js"></script>

<script

src="webjars/bootstrap-datepicker/1.0.1/js/bootstrap-datepicker.js"></script>

<script>

$('#targetDate').datepicker({

format : 'dd/mm/yyyy'

});

</script>

</body>

</html>

**src/main/webapp/WEB-INF/jsp/common/header.jspf**

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>

<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt"%>

<%@taglib uri="http://www.springframework.org/tags/form" prefix="form"%>

<html>

<head>

<title>First Web Application</title>

<link href="webjars/bootstrap/3.3.6/css/bootstrap.min.css"

rel="stylesheet">

</head>

<body>

**src/main/webapp/WEB-INF/jsp/common/navigation.jspf**

<nav role="navigation" class="navbar navbar-default">

<div class="">

<a href="http://www.in28minutes.com" class="navbar-brand">in28Minutes</a>

</div>

<div class="navbar-collapse">

<ul class="nav navbar-nav">

<li class="active"><a href="/">Home</a></li>

<li><a href="/list-todos">Todos</a></li>

</ul>

</div>

</nav>

**src/main/webapp/WEB-INF/jsp/list-todos.jsp**

<%@ include file="common/header.jspf" %>

<%@ include file="common/navigation.jspf" %>

<div class="container">

<table class="table table-striped">

<caption>Your todos are</caption>

<thead>

<tr>

<th>Description</th>

<th>Target Date</th>

<th>Is it Done?</th>

<th></th>

<th></th>

</tr>

</thead>

<tbody>

<c:forEach items="${todos}" var="todo">

<tr>

<td>${todo.desc}</td>

<td><fmt:formatDate value="${todo.targetDate}" pattern="dd/MM/yyyy"/></td>

<td>${todo.done}</td>

<td><a type="button" class="btn btn-success"

href="/update-todo?id=${todo.id}">Update</a></td>

<td><a type="button" class="btn btn-warning"

href="/delete-todo?id=${todo.id}">Delete</a></td>

</tr>

</c:forEach>

</tbody>

</table>

<div>

<a class="button" href="/add-todo">Add a Todo</a>

</div>

</div>

<%@ include file="common/footer.jspf" %>

**src/main/webapp/WEB-INF/jsp/todo.jsp**

<%@ include file="common/header.jspf" %>

<%@ include file="common/navigation.jspf" %>

<div class="container">

<form:form method="post" commandName="todo">

<form:hidden path="id" />

<fieldset class="form-group">

<form:label path="desc">Description</form:label>

<form:input path="desc" type="text" class="form-control"

required="required" />

<form:errors path="desc" cssClass="text-warning" />

</fieldset>

<fieldset class="form-group">

<form:label path="targetDate">Target Date</form:label>

<form:input path="targetDate" type="text" class="form-control"

required="required" />

<form:errors path="targetDate" cssClass="text-warning" />

</fieldset>

<button type="submit" class="btn btn-success">Add</button>

</form:form>

</div>

<%@ include file="common/footer.jspf" %>

**src/main/webapp/WEB-INF/jsp/welcome.jsp**

<%@ include file="common/header.jspf"%>

<%@ include file="common/navigation.jspf"%>

<div class="container">

Welcome ${name}!! <a href="/list-todos">Click here</a> to manage your

todo's.

</div>

<%@ include file="common/footer.jspf"%>

**src/test/java/com/in28minutes/springboot/web/SpringBootFirstWebApplicationTests.java**

package com.in28minutes.springboot.web;

import org.junit.Test;

import org.junit.runner.RunWith;

import org.springframework.boot.test.context.SpringBootTest;

import org.springframework.test.context.junit4.SpringRunner;

@RunWith(SpringRunner.class)

@SpringBootTest

public class SpringBootFirstWebApplicationTests {

@Test

public void contextLoads() {

}

}

**todo.txt**

Implementing Server Side Validation

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Command Bean or Form Backing Bean

Add Validation

Use Validation on Controller

Display Errors in View

Command Bean

~~~~~~~~~~~~

Controller

View - Spring Form Tags

LoginController -> adds name to model

welcome.jsp -> shows ${name}

TodoController -> redirects to list-todos.jsp

${name} is empty

Component, Service, Repository, Controller

Autowired

ComponentScan

Field dummyService in com.in28minutes.springboot.web.controller.LoginController

required a bean of type 'com.in28minutes.dummy.DummyService'

that could not be found.

Spring Boot Starter Parent

Spring Boot Starter Web

@SpringBootApplication

Auto Configuration

Dispatcher Servlet

/login => "login"

"login" => src/main/webapp/WEB-INF/jsp/login.jsp

Search for a view named "login"

/login => LoginController

Note: COURSE UPDATE : Spring Security

For the recent versions of Spring Boot, You need to use NoOpPasswordEncoder.getInstance. Remember this if you face any errors in the next step.

**New Code**

**auth.inMemoryAuthentication().passwordEncoder(NoOpPasswordEncoder.getInstance()).withUser("in28Minutes").password("dummy")**

**Old Code**

**auth.inMemoryAuthentication().withUser("in28Minutes").password("dummy")**

**Step23.md: Initial Spring Security Setup**

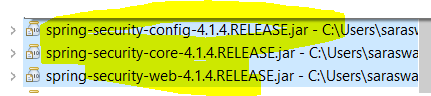
<https://github.com/in28minutes/spring-boot-master-class/blob/master/02.Spring-Boot-Web-Application/Step23.md>

Here in this step we will do all the stuff which is requerd to setup for security.

As we already know that in spring boot we have one of the starter dependency for the web (spring-boot-starter-web) for spring MVC and rest based application.

Similar to that we have another starter dependency for security (spring-boot-starter-security)

|  |  |
| --- | --- |
| <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-web</artifactId>  </dependency> | <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-security</artifactId>  </dependency> |



As soon as we include security dependency in the pom.xml we get these three jar file inside the maven dependency.

Apart from getting these dependencies in maven dependency it does some auto configuration. For example after adding this dependency, once we restart the server, if we type <http://localhost:8080/> then we will get one login popup on the browser.

|  |  |
| --- | --- |
|  |  |

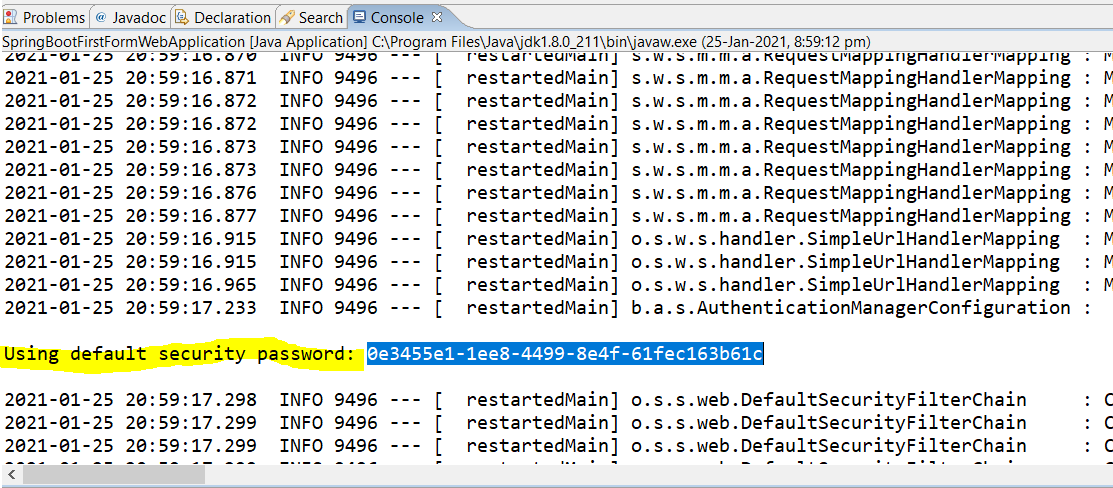
So what happens we add security dependency in the pom.xml, the spring boot security starter would automatically configure spring security in entire spring boot application. So now every URL in the spring boot application is protected. Now if click on cancel the login screen then we get whitelabel error page saying “Full authentication is required to access this resource”.

So now if we try to access <http://localhost:8080/add-todo> or <http://localhost:8080/list-todo> then we can not access those pages without login and if try to access the above URL then spring security will automatically redirect to login page to login first to access those pages.

Since this login has been automatically configured by spring security, so it provides the default username and password .

The default username is – user

The password we will get from console which will be generated by spring security to login.



Now if give username= user and password = as printed in the console then we will be able to login and will get home page.

|  |  |
| --- | --- |
|  |  |

Hence in this way we can add spring security in our application. But here in default configuration we don’t have any control on login page and username and password, so we would like to take control in our hand. Typically the we have LDAP to manage username and password. But here to take control of username password, let’s create in-memory storage for username and password for login.

Let’s create a small spring security configuration, so that add the username and password.

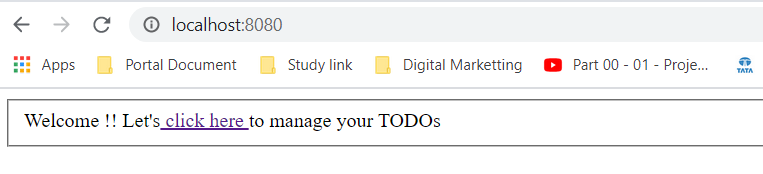
* Create a new class inside (let’s say in package com.web.security) SecurityConfiguration.java
* To be picked up it as configuration file we will use annotation (@Configuration) at the top of SecurityConfiguration.java class. This @Configuration is the spring annotation which will help us to add more configurations in the spring boot application, so here in this class we will add little bit more configurations. Basically this annotation defines a class having one or more beans or method to be processed by spring container to generate bean or method definition. So here in this class we would like to configure bean processes by the spring container
* To enable this class as configuration class we will have to extends a class called [WebSecurityConfigurerAdapter]
* We will override few of the method available in the WebSecurityConfigurerAdapter to be able to add our own security configuration
* There are two type of configuration that we would like to do here.
* First one is create a user (let’s say we have create one user having credential (in28Minutes/dummy)
* Second thing is when an un-authorized or un-authenticated user comes to login then we can ask to put correct username and password, so we will create a login form when user is un-autorized
* So first let’s create a user and also assigne the role to the user marked in above of yellow marked comments. Now after wirting this method (configureGlobalSecurity) if we run the application, then the default username=user and password=(provided by spring security in the console) will be override with the give username = “in28Minuts” and password=”dummy” and now instead of login with username = “user” we will be able to login by (in28Minuts/dummy)
* Now the second thing is to get proper login page to login instead of getting login popup in the browser, this will achive by overriding a specific method available in WebSecurityConfigurerAdapter called [configure], the second yellow marked line is used for this only

|  |
| --- |
| **package com.in28minutes.security;**  **import org.springframework.beans.factory.annotation.Autowired;**  **import org.springframework.context.annotation.Configuration;**  **import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;**  **import org.springframework.security.config.annotation.web.builders.HttpSecurity;**  **import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;**  **import org.springframework.security.crypto.password.NoOpPasswordEncoder;**  **@Configuration**  **public class SecurityConfiguration extends WebSecurityConfigurerAdapter{**  **//Create User - in28Minutes/dummy**  **@Autowired**  **public void configureGlobalSecurity(AuthenticationManagerBuilder auth) throws Exception {**  **auth.inMemoryAuthentication()**  **.passwordEncoder(NoOpPasswordEncoder.getInstance())**  **.withUser("in28Minutes").password("dummy")**  **.roles("USER", "ADMIN");**  **}**  **// Here in the above method we are creating in-memory authentication and we are creating username and password and giving this user roles is as USER & ADMIN both.**    **@Override**  **protected void configure(HttpSecurity http) throws Exception {**  **http.authorizeRequests().antMatchers("/login", "/h2-console/\*\*").permitAll()**  **.antMatchers("/", "/\*todo\*/\*\*").access("hasRole('USER')").and()**  **.formLogin();**    **http.csrf().disable();**  **http.headers().frameOptions().disable();**  **}**  **// Here in the above method for (/login) > permitAll and if anyone want to go (/) root page or todo page then that should have role as USER and if he/she does not have USER role then shown him/her login form.**  **}** |

Now after adding the above two methods, if we run the application then we will get poper login page instead of getting login popup to login.

|  |  |
| --- | --- |
|  |  |

And now here we go:



We have logged in successfully using spring security.

Below find all the code and snipptet

What we will do:

* Get Setup for Spring Security

Useful Snippets

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-security</artifactId>

</dependency>

package com.in28minutes.security;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;

import org.springframework.security.config.annotation.web.builders.HttpSecurity;

import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;

import org.springframework.security.crypto.password.NoOpPasswordEncoder;

@Configuration

public class SecurityConfiguration extends WebSecurityConfigurerAdapter{

//Create User - in28Minutes/dummy

@Autowired

public void configureGlobalSecurity(AuthenticationManagerBuilder auth)

throws Exception {

auth.inMemoryAuthentication()

.passwordEncoder(NoOpPasswordEncoder.getInstance())

.withUser("in28Minutes").password("dummy")

.roles("USER", "ADMIN");

}

@Override

protected void configure(HttpSecurity http) throws Exception {

http.authorizeRequests().antMatchers("/login", "/h2-console/\*\*").permitAll()

.antMatchers("/", "/\*todo\*/\*\*").access("hasRole('USER')").and()

.formLogin();

http.csrf().disable();

http.headers().frameOptions().disable();

}

}

Not Needed anymore with Spring Boot Auto Configuration

<filter>

<filter-name>springSecurityFilterChain</filter-name>

<filter-class>org.springframework.web.filter.DelegatingFilterProxy</filter-class>

</filter>

<filter-mapping>

<filter-name>springSecurityFilterChain</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

Files List

pom.xml

<?xml version="1.0" encoding="UTF-8"?>

<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">

<modelVersion>4.0.0</modelVersion>

<groupId>com.in28minutes.springboot.web</groupId>

<artifactId>spring-boot-first-web-application</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>spring-boot-first-web-application</name>

<description>Demo project for Spring Boot</description>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>1.4.3.RELEASE</version>

<relativePath /> <!-- lookup parent from repository -->

</parent>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>

<java.version>1.8</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-security</artifactId>

</dependency>

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>jstl</artifactId>

</dependency>

<dependency>

<groupId>org.webjars</groupId>

<artifactId>bootstrap</artifactId>

<version>3.3.6</version>

</dependency>

<dependency>

<groupId>org.webjars</groupId>

<artifactId>bootstrap-datepicker</artifactId>

<version>1.0.1</version>

</dependency>

<dependency>

<groupId>org.webjars</groupId>

<artifactId>jquery</artifactId>

<version>1.9.1</version>

</dependency>

<dependency>

<groupId>org.apache.tomcat.embed</groupId>

<artifactId>tomcat-embed-jasper</artifactId>

<scope>provided</scope>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-devtools</artifactId>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

src/main/java/com/in28minutes/springboot/web/controller/LoginController.java

package com.in28minutes.springboot.web.controller;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Controller;

import org.springframework.ui.ModelMap;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestMethod;

import org.springframework.web.bind.annotation.SessionAttributes;

import com.in28minutes.springboot.web.service.LoginService;

@Controller

@SessionAttributes("name")

public class LoginController {

@Autowired

LoginService service;

@RequestMapping(value = "/", method = RequestMethod.GET)

public String showLoginPage(ModelMap model) {

model.put("name", "in28Minutes");

return "welcome";

}

}

src/main/java/com/in28minutes/springboot/web/controller/TodoController.java

package com.in28minutes.springboot.web.controller;

import java.text.SimpleDateFormat;

import java.util.Date;

import javax.validation.Valid;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.beans.propertyeditors.CustomDateEditor;

import org.springframework.stereotype.Controller;

import org.springframework.ui.ModelMap;

import org.springframework.validation.BindingResult;

import org.springframework.web.bind.WebDataBinder;

import org.springframework.web.bind.annotation.InitBinder;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestMethod;

import org.springframework.web.bind.annotation.RequestParam;

import org.springframework.web.bind.annotation.SessionAttributes;

import com.in28minutes.springboot.web.model.Todo;

import com.in28minutes.springboot.web.service.LoginService;

import com.in28minutes.springboot.web.service.TodoService;

@Controller

@SessionAttributes("name")

public class TodoController {

@Autowired

TodoService service;

@InitBinder

public void initBinder(WebDataBinder binder) {

// Date - dd/MM/yyyy

SimpleDateFormat dateFormat = new SimpleDateFormat("dd/MM/yyyy");

binder.registerCustomEditor(Date.class, new CustomDateEditor(

dateFormat, false));

}

@RequestMapping(value = "/list-todos", method = RequestMethod.GET)

public String showTodos(ModelMap model) {

String name = getLoggedInUserName(model);

model.put("todos", service.retrieveTodos(name));

return "list-todos";

}

private String getLoggedInUserName(ModelMap model) {

return (String) model.get("name");

}

@RequestMapping(value = "/add-todo", method = RequestMethod.GET)

public String showAddTodoPage(ModelMap model) {

model.addAttribute("todo", new Todo(0, getLoggedInUserName(model),

"Default Desc", new Date(), false));

return "todo";

}

@RequestMapping(value = "/delete-todo", method = RequestMethod.GET)

public String deleteTodo(@RequestParam int id) {

service.deleteTodo(id);

return "redirect:/list-todos";

}

@RequestMapping(value = "/update-todo", method = RequestMethod.GET)

public String showUpdateTodoPage(@RequestParam int id, ModelMap model) {

Todo todo = service.retrieveTodo(id);

model.put("todo", todo);

return "todo";

}

@RequestMapping(value = "/update-todo", method = RequestMethod.POST)

public String updateTodo(ModelMap model, @Valid Todo todo,

BindingResult result) {

if (result.hasErrors()) {

return "todo";

}

todo.setUser(getLoggedInUserName(model));

service.updateTodo(todo);

return "redirect:/list-todos";

}

@RequestMapping(value = "/add-todo", method = RequestMethod.POST)

public String addTodo(ModelMap model, @Valid Todo todo, BindingResult result) {

if (result.hasErrors()) {

return "todo";

}

service.addTodo(getLoggedInUserName(model), todo.getDesc(), todo.getTargetDate(),

false);

return "redirect:/list-todos";

}

}

src/main/java/com/in28minutes/springboot/web/model/Todo.java

package com.in28minutes.springboot.web.model;

import java.util.Date;

import javax.validation.constraints.Size;

public class Todo {

private int id;

private String user;

@Size(min=10, message="Enter at least 10 Characters...")

private String desc;

private Date targetDate;

private boolean isDone;

public Todo() {

super();

}

public Todo(int id, String user, String desc, Date targetDate,

boolean isDone) {

super();

this.id = id;

this.user = user;

this.desc = desc;

this.targetDate = targetDate;

this.isDone = isDone;

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getUser() {

return user;

}

public void setUser(String user) {

this.user = user;

}

public String getDesc() {

return desc;

}

public void setDesc(String desc) {

this.desc = desc;

}

public Date getTargetDate() {

return targetDate;

}

public void setTargetDate(Date targetDate) {

this.targetDate = targetDate;

}

public boolean isDone() {

return isDone;

}

public void setDone(boolean isDone) {

this.isDone = isDone;

}

@Override

public int hashCode() {

final int prime = 31;

int result = 1;

result = prime \* result + id;

return result;

}

@Override

public boolean equals(Object obj) {

if (this == obj) {

return true;

}

if (obj == null) {

return false;

}

if (getClass() != obj.getClass()) {

return false;

}

Todo other = (Todo) obj;

if (id != other.id) {

return false;

}

return true;

}

@Override

public String toString() {

return String.format(

"Todo [id=%s, user=%s, desc=%s, targetDate=%s, isDone=%s]", id,

user, desc, targetDate, isDone);

}

}

src/main/java/com/in28minutes/springboot/web/security/SecurityConfiguration.java

package com.in28minutes.springboot.web.security;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;

import org.springframework.security.config.annotation.web.builders.HttpSecurity;

import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;

@Configuration

public class SecurityConfiguration extends WebSecurityConfigurerAdapter{

//Create User - in28Minutes/dummy

@Autowired

public void configureGlobalSecurity(AuthenticationManagerBuilder auth)

throws Exception {

auth.inMemoryAuthentication().withUser("in28Minutes").password("dummy")

.roles("USER", "ADMIN");

}

@Override

protected void configure(HttpSecurity http) throws Exception {

http.authorizeRequests().antMatchers("/login").permitAll()

.antMatchers("/", "/\*todo\*/\*\*").access("hasRole('USER')").and()

.formLogin();

}

}

src/main/java/com/in28minutes/springboot/web/service/LoginService.java

package com.in28minutes.springboot.web.service;

import org.springframework.stereotype.Component;

import org.springframework.stereotype.Service;

@Service

public class LoginService {

public boolean validateUser(String userid, String password) {

// in28minutes, dummy

return userid.equalsIgnoreCase("in28minutes")

&& password.equalsIgnoreCase("dummy");

}

}

src/main/java/com/in28minutes/springboot/web/service/TodoService.java

package com.in28minutes.springboot.web.service;

import java.util.ArrayList;

import java.util.Date;

import java.util.Iterator;

import java.util.List;

import org.springframework.stereotype.Service;

import com.in28minutes.springboot.web.model.Todo;

@Service

public class TodoService {

private static List<Todo> todos = new ArrayList<Todo>();

private static int todoCount = 3;

static {

todos.add(new Todo(1, "in28Minutes", "Learn Spring MVC", new Date(),

false));

todos.add(new Todo(2, "in28Minutes", "Learn Struts", new Date(), false));

todos.add(new Todo(3, "in28Minutes", "Learn Hibernate", new Date(),

false));

}

public List<Todo> retrieveTodos(String user) {

List<Todo> filteredTodos = new ArrayList<Todo>();

for (Todo todo : todos) {

if (todo.getUser().equalsIgnoreCase(user)) {

filteredTodos.add(todo);

}

}

return filteredTodos;

}

public Todo retrieveTodo(int id) {

for (Todo todo : todos) {

if (todo.getId()==id) {

return todo;

}

}

return null;

}

public void updateTodo(Todo todo){

todos.remove(todo);

todos.add(todo);

}

public void addTodo(String name, String desc, Date targetDate,

boolean isDone) {

todos.add(new Todo(++todoCount, name, desc, targetDate, isDone));

}

public void deleteTodo(int id) {

Iterator<Todo> iterator = todos.iterator();

while (iterator.hasNext()) {

Todo todo = iterator.next();

if (todo.getId() == id) {

iterator.remove();

}

}

}

}

src/main/java/com/in28minutes/springboot/web/SpringBootFirstWebApplication.java

package com.in28minutes.springboot.web;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.annotation.ComponentScan;

@SpringBootApplication

@ComponentScan("com.in28minutes.springboot.web")

public class SpringBootFirstWebApplication {

public static void main(String[] args) {

SpringApplication.run(SpringBootFirstWebApplication.class, args);

}

}

src/main/resources/application.properties

spring.mvc.view.prefix=/WEB-INF/jsp/

spring.mvc.view.suffix=.jsp

logging.level.org.springframework.web=INFO

src/main/webapp/WEB-INF/jsp/common/footer.jspf

<script src="webjars/jquery/1.9.1/jquery.min.js"></script>

<script src="webjars/bootstrap/3.3.6/js/bootstrap.min.js"></script>

<script

src="webjars/bootstrap-datepicker/1.0.1/js/bootstrap-datepicker.js"></script>

<script>

$('#targetDate').datepicker({

format : 'dd/mm/yyyy'

});

</script>

</body>

</html>

src/main/webapp/WEB-INF/jsp/common/header.jspf

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>

<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt"%>

<%@taglib uri="http://www.springframework.org/tags/form" prefix="form"%>

<html>

<head>

<title>First Web Application</title>

<link href="webjars/bootstrap/3.3.6/css/bootstrap.min.css"

rel="stylesheet">

</head>

<body>

src/main/webapp/WEB-INF/jsp/common/navigation.jspf

<nav role="navigation" class="navbar navbar-default">

<div class="">

<a href="http://www.in28minutes.com" class="navbar-brand">in28Minutes</a>

</div>

<div class="navbar-collapse">

<ul class="nav navbar-nav">

<li class="active"><a href="/">Home</a></li>

<li><a href="/list-todos">Todos</a></li>

</ul>

</div>

</nav>

src/main/webapp/WEB-INF/jsp/list-todos.jsp

<%@ include file="common/header.jspf" %>

<%@ include file="common/navigation.jspf" %>

<div class="container">

<table class="table table-striped">

<caption>Your todos are</caption>

<thead>

<tr>

<th>Description</th>

<th>Target Date</th>

<th>Is it Done?</th>

<th></th>

<th></th>

</tr>

</thead>

<tbody>

<c:forEach items="${todos}" var="todo">

<tr>

<td>${todo.desc}</td>

<td><fmt:formatDate value="${todo.targetDate}" pattern="dd/MM/yyyy"/></td>

<td>${todo.done}</td>

<td><a type="button" class="btn btn-success"

href="/update-todo?id=${todo.id}">Update</a></td>

<td><a type="button" class="btn btn-warning"

href="/delete-todo?id=${todo.id}">Delete</a></td>

</tr>

</c:forEach>

</tbody>

</table>

<div>

<a class="button" href="/add-todo">Add a Todo</a>

</div>

</div>

<%@ include file="common/footer.jspf" %>

src/main/webapp/WEB-INF/jsp/todo.jsp

<%@ include file="common/header.jspf" %>

<%@ include file="common/navigation.jspf" %>

<div class="container">

<form:form method="post" commandName="todo">

<form:hidden path="id" />

<fieldset class="form-group">

<form:label path="desc">Description</form:label>

<form:input path="desc" type="text" class="form-control"

required="required" />

<form:errors path="desc" cssClass="text-warning" />

</fieldset>

<fieldset class="form-group">

<form:label path="targetDate">Target Date</form:label>

<form:input path="targetDate" type="text" class="form-control"

required="required" />

<form:errors path="targetDate" cssClass="text-warning" />

</fieldset>

<button type="submit" class="btn btn-success">Add</button>

</form:form>

</div>

<%@ include file="common/footer.jspf" %>

src/main/webapp/WEB-INF/jsp/welcome.jsp

<%@ include file="common/header.jspf"%>

<%@ include file="common/navigation.jspf"%>

<div class="container">

Welcome ${name}!! <a href="/list-todos">Click here</a> to manage your

todo's.

</div>

<%@ include file="common/footer.jspf"%>

src/test/java/com/in28minutes/springboot/web/SpringBootFirstWebApplicationTests.java

package com.in28minutes.springboot.web;

import org.junit.Test;

import org.junit.runner.RunWith;

import org.springframework.boot.test.context.SpringBootTest;

import org.springframework.test.context.junit4.SpringRunner;

@RunWith(SpringRunner.class)

@SpringBootTest

public class SpringBootFirstWebApplicationTests {

@Test

public void contextLoads() {

}

}

todo.txt

Implementing Server Side Validation

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Command Bean or Form Backing Bean

Add Validation

Use Validation on Controller

Display Errors in View

Command Bean

~~~~~~~~~~~~

Controller

View - Spring Form Tags

LoginController -> adds name to model

welcome.jsp -> shows ${name}

TodoController -> redirects to list-todos.jsp

${name} is empty

Component, Service, Repository, Controller

Autowired

ComponentScan

Field dummyService in com.in28minutes.springboot.web.controller.LoginController

required a bean of type 'com.in28minutes.dummy.DummyService'

that could not be found.

Spring Boot Starter Parent

Spring Boot Starter Web

@SpringBootApplication

Auto Configuration

Dispatcher Servlet

/login => "login"

"login" => src/main/webapp/WEB-INF/jsp/login.jsp

Search for a view named "login"

/login => LoginController

**Step24.md: Refactor and add Logout functionality using spring security.**

<https://github.com/in28minutes/spring-boot-master-class/blob/master/02.Spring-Boot-Web-Application/Step24.md>

As in the previous step we have hard coded the username and password to get logged in. So here in this step we will perform the following things.

What we will do:

* Remove Hardcoding of User Name
* Remove LoginService
* Rename LoginController to WelcomeController
* Add Logout Functionality

|  |
| --- |
| Useful Snippets  **Mthod to get loggedinusername and password from spring security.**  private String getLoggedInUserName(ModelMap model) {  Object principal = SecurityContextHolder.getContext().getAuthentication().getPrincipal();  if (principal instanceof UserDetails){  return ((UserDetails) principal).getUsername();  }else  return principal.toString();  }  // Here UserDetails is the spring security class which stores the user credential, so will the username from UserDetails class. If it found username then return ussernam from UserDetails otherwise return principal object.  **Below code will used in nav.jspf**  <ul class="nav navbar-nav navbar-right">  <li><a href="/logout">Logout</a></li>  </ul>  **LogoutController.java** -----------------------------------------------------------------------------------  @RequestMapping(value = "/logout", method = RequestMethod.GET)  public String logout(HttpServletRequest request,HttpServletResponse response) {  Authentication auth = SecurityContextHolder.getContext().getAuthentication();  if (auth != null) {  new SecurityContextLogoutHandler().logout(request, response, auth);  }  return "redirect:/";  } |

**Remove Hardcoding of User Name**

Since in the previous step the username was hardcoded in the model object, so now instead of hardcoded we will get the user name from getLoggedinUserName () method. So first will write the method to get username and password from spring security and then this method will be used to get loggedin username anywhery wherever required. The loggedinUser in spring security terminology is called principal, so first thing we nee to get principal. The method is wrttien above inside useful snippet.

|  |
| --- |
| @RequestMapping(value="/", method=RequestMethod.***GET***)  **public** String loginMessage( ModelMap model) {  System.***out***.println("inside login controll");  //model.put("nameKey", "in28Minutes"); Comment this line as username was hardcode  model.put("nameKey", **getLoggedInUserName(model)**); Getting the username from loggedinUsername(model)  //return"login";  **return**"welcome";  }  **private** String **getLoggedInUserName(ModelMap model)** {  Object principal = SecurityContextHolder.*getContext*().getAuthentication().getPrincipal();  **if** (principal **instanceof** UserDetails){  **return** ((UserDetails) principal).getUsername();  }**else**  **return** principal.toString();  } |

In TodoController.java we have wrttien method getLoggedInUserName (model) where we are getting user name from session. Now let’s take the username here also from the spring security UserDetails. So there are two ways to get loggedInuserName but the best way is to get getLoggedInUserName from spring security.

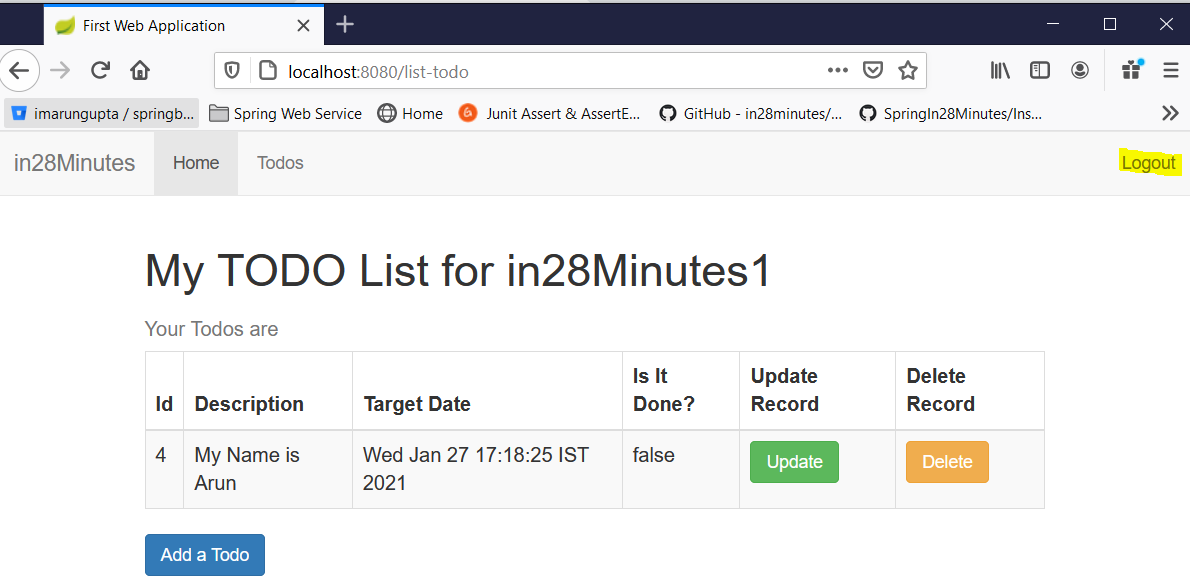
|  |
| --- |
| Old Code: Getting user name from session  @Controller  @SessionAttributes("nameKey")  **public** **class** TodoController {    **private** String **getLoggedInUserName**(ModelMap model) {  **return** (String) model.get("nameKey"); // Username getting from session  } |
| New code: Username getting from userDetails class  **private** String **getLoggedInUserName**(ModelMap model) {  Object principal = SecurityContextHolder.*getContext*().getAuthentication().getPrincipal();  **if** (principal **instanceof** UserDetails){  **return** ((UserDetails) principal).getUsername();  }**else**  **return** principal.toString();  } |

**Now let’s add logout functionality:**

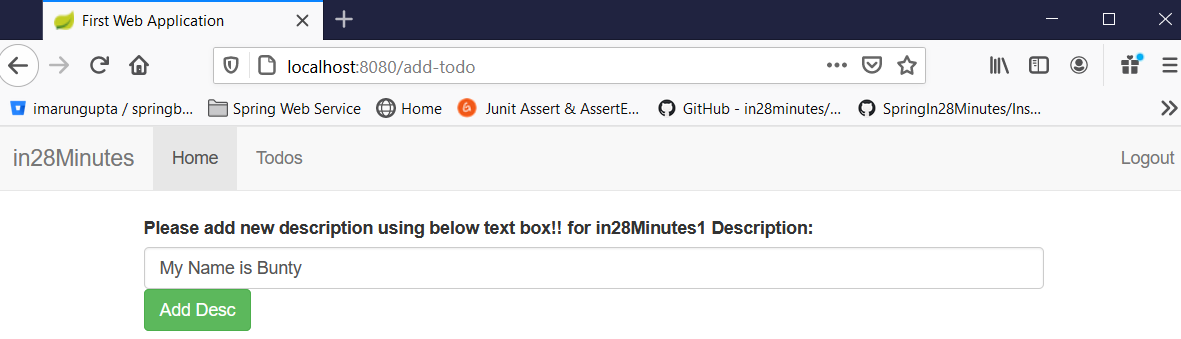
1. First thing will add Logout link in nav.jspf ( as shown in snippet above)
2. Crate a LogoutController.java and use the above code mentioned in snippet

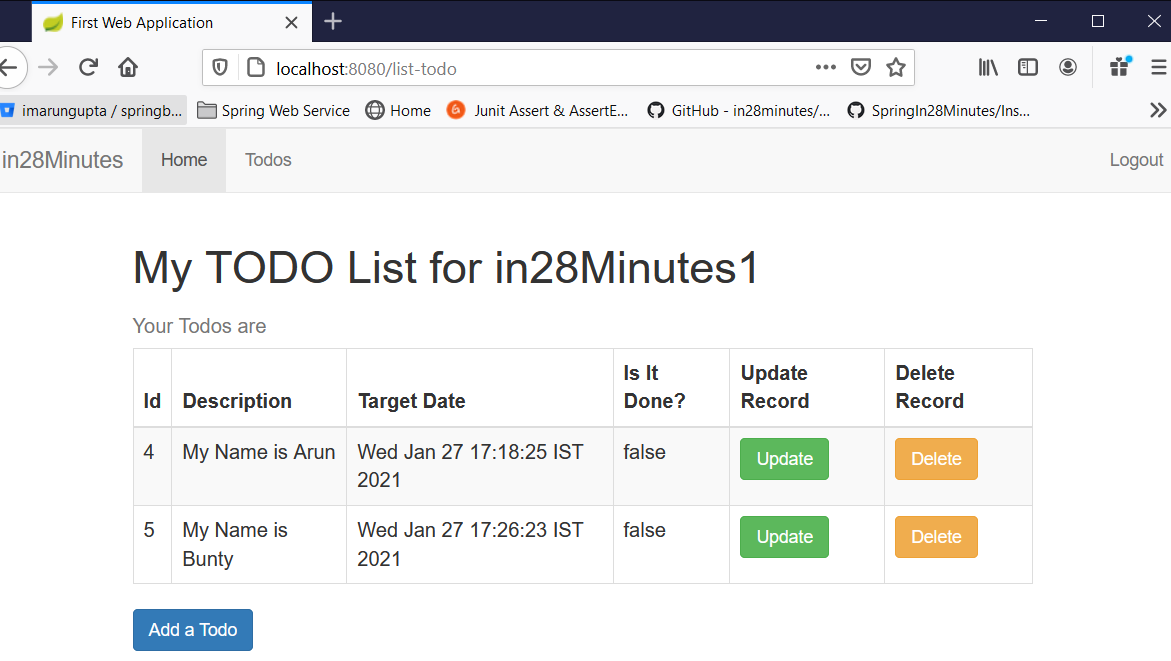
|  |  |
| --- | --- |
|  |  |
|  |  |

Untill we have create a typical web application login, logout , create, update , delete functionality.

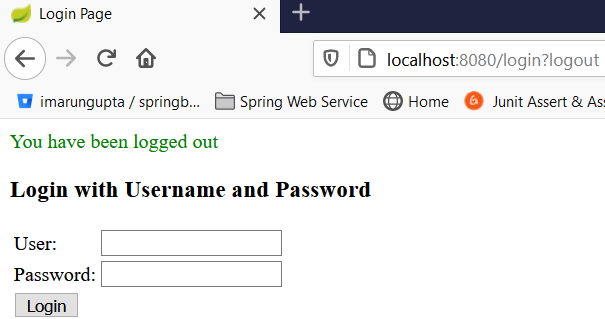


Add:





Click on Logout:



**Exception Handaling in spring:**

It is one of the most important topics in the spring boot. Exception handling is nothing but, show that, how our application reacts if something goes unexpected in the application. Exception Handling is a cross cutting concern, irespectinv it is weblayer or data layer, every layer has to do exception handling.

One of the great things about spring is that we don’t really need to put all your exception handling logic in multiple places. In the old application we were having lot of try-catch block across application (wheter it data layer, business layer, every layer were having try-catch block to catch exception). But with the spring it is possible to have centralized exception handling at one place and use it across application.

As we already have seen in so many places that if spring boot provide one default whitelabel Error Page for all the exception. For example the correct URL to get login page is: <http://localhost:8080/login> and now instead of this if we give the URL like <http://localhost:8080/dummy> the please see the difference below

|  |  |
| --- | --- |
| <http://localhost:8080/login> | <http://localhost:8080/dummy> |
| Here in case of correct url we got login page | Here in case worng url we got a default exception (whitelabel error page) with 404. No message available |

So this something provided by default by the spring boot. We can override the error message with our own exception message. For example let’s try to throw our own user defined exception while deleting the record for specific id.

|  |  |
| --- | --- |
| @RequestMapping(value="/delete-todo", method=RequestMethod.***GET***)  **public** String deleteTodo(@RequestParam **int** id) {    **if**(id==6) {  **throw** **new** RuntimeException("Something went wrong");  }  service.deleteTodo(id);  **return** "redirect:/list-todo";  } |  |

So we can see that this is the default centralized exception provided by spring boot itself and we can override the runtime exception according to our need.

What we will do:

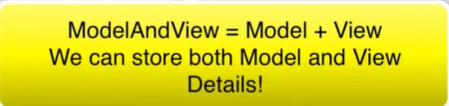
* Basic Exception Handling
* Exception Handling is a cross cutting concern
* Do not handle exceptions in Controllers or Services, if you cannot add value to them.
* Bit of refactoring on the controllers
* Whitelabel Error Page provided by default by Spring Boot
* You can see a few details of the errors
* We can customize if we would want to
* @ControllerAdvice and Controller Specific Exception Handling
* Handling Errors thrown from Views

Let’s create our own customized exception handlaing and error page. Follow the steps below.

1. Create a controller class with the name ErrorController.
2. Here we can add handler for specific exceptions by using annotation @ExceptionHandler(provide the exception type we want to handle).
3. Create a method of type ModelAndView and add exception and request URL object into ModelAndView object and send this object into the view by setting the view as shown in the below snippet code.
4. Create one view error.jsp

Useful Snippets

|  |
| --- |
| @Controller("error")  public class ExceptionController {  private Log logger = LogFactory.getLog(ExceptionController.class);  @ExceptionHandler(Exception.class)  public ModelAndView handleError(HttpServletRequest req, Exception ex) {  logger.error("Request: " + req.getRequestURL() + " raised " + ex);  ModelAndView mav = new ModelAndView();  mav.addObject("exception", ex);  mav.addObject("url", req.getRequestURL());  mav.setViewName("error");  return mav;  }  } |

****

Files List

pom.xml

<?xml version="1.0" encoding="UTF-8"?>

<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">

<modelVersion>4.0.0</modelVersion>

<groupId>com.in28minutes.springboot.web</groupId>

<artifactId>spring-boot-first-web-application</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>spring-boot-first-web-application</name>

<description>Demo project for Spring Boot</description>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>1.4.3.RELEASE</version>

<relativePath /> <!-- lookup parent from repository -->

</parent>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>

<java.version>1.8</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-security</artifactId>

</dependency>

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>jstl</artifactId>

</dependency>

<dependency>

<groupId>org.webjars</groupId>

<artifactId>bootstrap</artifactId>

<version>3.3.6</version>

</dependency>

<dependency>

<groupId>org.webjars</groupId>

<artifactId>bootstrap-datepicker</artifactId>

<version>1.0.1</version>

</dependency>

<dependency>

<groupId>org.webjars</groupId>

<artifactId>jquery</artifactId>

<version>1.9.1</version>

</dependency>

<dependency>

<groupId>org.apache.tomcat.embed</groupId>

<artifactId>tomcat-embed-jasper</artifactId>

<scope>provided</scope>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-devtools</artifactId>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

src/main/java/com/in28minutes/springboot/web/controller/ErrorController.java

package com.in28minutes.springboot.web.controller;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import org.springframework.stereotype.Controller;

import org.springframework.web.bind.annotation.ExceptionHandler;

import org.springframework.web.servlet.ModelAndView;

@Controller("error")

public class ErrorController {

@ExceptionHandler(Exception.class)

public ModelAndView handleException

(HttpServletRequest request, Exception ex){

ModelAndView mv = new ModelAndView();

mv.addObject("exception", ex.getLocalizedMessage());

mv.addObject("url", request.getRequestURL());

mv.setViewName("error");

return mv;

}

}

src/main/java/com/in28minutes/springboot/web/controller/LogoutController.java

package com.in28minutes.springboot.web.controller;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import org.springframework.security.core.Authentication;

import org.springframework.security.core.context.SecurityContextHolder;

import org.springframework.security.core.userdetails.UserDetails;

import org.springframework.security.web.authentication.logout.LogoutHandler;

import org.springframework.security.web.authentication.logout.SecurityContextLogoutHandler;

import org.springframework.stereotype.Controller;

import org.springframework.ui.ModelMap;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestMethod;

@Controller

public class LogoutController {

@RequestMapping(value = "/logout", method = RequestMethod.GET)

public String logout(HttpServletRequest request,

HttpServletResponse response) {

Authentication authentication = SecurityContextHolder.getContext()

.getAuthentication();

if (authentication != null) {

new SecurityContextLogoutHandler().logout(request, response,

authentication);

}

return "redirect:/";

}

}

src/main/java/com/in28minutes/springboot/web/controller/TodoController.java

package com.in28minutes.springboot.web.controller;

import java.text.SimpleDateFormat;

import java.util.Date;

import javax.validation.Valid;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.beans.propertyeditors.CustomDateEditor;

import org.springframework.security.core.context.SecurityContextHolder;

import org.springframework.security.core.userdetails.UserDetails;

import org.springframework.stereotype.Controller;

import org.springframework.ui.ModelMap;

import org.springframework.validation.BindingResult;

import org.springframework.web.bind.WebDataBinder;

import org.springframework.web.bind.annotation.InitBinder;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestMethod;

import org.springframework.web.bind.annotation.RequestParam;

import com.in28minutes.springboot.web.model.Todo;

import com.in28minutes.springboot.web.service.TodoService;

@Controller

public class TodoController {

@Autowired

TodoService service;

@InitBinder

public void initBinder(WebDataBinder binder) {

// Date - dd/MM/yyyy

SimpleDateFormat dateFormat = new SimpleDateFormat("dd/MM/yyyy");

binder.registerCustomEditor(Date.class, new CustomDateEditor(

dateFormat, false));

}

@RequestMapping(value = "/list-todos", method = RequestMethod.GET)

public String showTodos(ModelMap model) {

String name = getLoggedInUserName(model);

model.put("todos", service.retrieveTodos(name));

return "list-todos";

}

private String getLoggedInUserName(ModelMap model) {

Object principal = SecurityContextHolder.getContext()

.getAuthentication().getPrincipal();

if (principal instanceof UserDetails) {

return ((UserDetails) principal).getUsername();

}

return principal.toString();

}

@RequestMapping(value = "/add-todo", method = RequestMethod.GET)

public String showAddTodoPage(ModelMap model) {

model.addAttribute("todo", new Todo(0, getLoggedInUserName(model),

"Default Desc", new Date(), false));

return "todo";

}

@RequestMapping(value = "/delete-todo", method = RequestMethod.GET)

public String deleteTodo(@RequestParam int id) {

if(id==1)

throw new RuntimeException("Something went wrong");

service.deleteTodo(id);

return "redirect:/list-todos";

}

@RequestMapping(value = "/update-todo", method = RequestMethod.GET)

public String showUpdateTodoPage(@RequestParam int id, ModelMap model) {

Todo todo = service.retrieveTodo(id);

model.put("todo", todo);

return "todo";

}

@RequestMapping(value = "/update-todo", method = RequestMethod.POST)

public String updateTodo(ModelMap model, @Valid Todo todo,

BindingResult result) {

if (result.hasErrors()) {

return "todo";

}

todo.setUser(getLoggedInUserName(model));

service.updateTodo(todo);

return "redirect:/list-todos";

}

@RequestMapping(value = "/add-todo", method = RequestMethod.POST)

public String addTodo(ModelMap model, @Valid Todo todo, BindingResult result) {

if (result.hasErrors()) {

return "todo";

}

service.addTodo(getLoggedInUserName(model), todo.getDesc(), todo.getTargetDate(),

false);

return "redirect:/list-todos";

}

}

src/main/java/com/in28minutes/springboot/web/controller/WelcomeController.java

package com.in28minutes.springboot.web.controller;

import org.springframework.security.core.context.SecurityContextHolder;

import org.springframework.security.core.userdetails.UserDetails;

import org.springframework.stereotype.Controller;

import org.springframework.ui.ModelMap;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestMethod;

@Controller

public class WelcomeController {

@RequestMapping(value = "/", method = RequestMethod.GET)

public String showWelcomePage(ModelMap model) {

model.put("name", getLoggedinUserName());

return "welcome";

}

private String getLoggedinUserName() {

Object principal = SecurityContextHolder.getContext()

.getAuthentication().getPrincipal();

if (principal instanceof UserDetails) {

return ((UserDetails) principal).getUsername();

}

return principal.toString();

}

}

src/main/java/com/in28minutes/springboot/web/model/Todo.java

package com.in28minutes.springboot.web.model;

import java.util.Date;

import javax.validation.constraints.Size;

public class Todo {

private int id;

private String user;

@Size(min=10, message="Enter at least 10 Characters...")

private String desc;

private Date targetDate;

private boolean isDone;

public Todo() {

super();

}

public Todo(int id, String user, String desc, Date targetDate,

boolean isDone) {

super();

this.id = id;

this.user = user;

this.desc = desc;

this.targetDate = targetDate;

this.isDone = isDone;

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getUser() {

return user;

}

public void setUser(String user) {

this.user = user;

}

public String getDesc() {

return desc;

}

public void setDesc(String desc) {

this.desc = desc;

}

public Date getTargetDate() {

return targetDate;

}

public void setTargetDate(Date targetDate) {

this.targetDate = targetDate;

}

public boolean isDone() {

return isDone;

}

public void setDone(boolean isDone) {

this.isDone = isDone;

}

@Override

public int hashCode() {

final int prime = 31;

int result = 1;

result = prime \* result + id;

return result;

}

@Override

public boolean equals(Object obj) {

if (this == obj) {

return true;

}

if (obj == null) {

return false;

}

if (getClass() != obj.getClass()) {

return false;

}

Todo other = (Todo) obj;

if (id != other.id) {

return false;

}

return true;

}

@Override

public String toString() {

return String.format(

"Todo [id=%s, user=%s, desc=%s, targetDate=%s, isDone=%s]", id,

user, desc, targetDate, isDone);

}

}

**src/main/java/com/in28minutes/springboot/web/security/SecurityConfiguration.java**

package com.in28minutes.springboot.web.security;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;

import org.springframework.security.config.annotation.web.builders.HttpSecurity;

import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;

@Configuration

public class SecurityConfiguration extends WebSecurityConfigurerAdapter{

//Create User - in28Minutes/dummy

@Autowired

public void configureGlobalSecurity(AuthenticationManagerBuilder auth)

throws Exception {

auth.inMemoryAuthentication().withUser("in28Minutes").password("dummy")

.roles("USER", "ADMIN");

}

@Override

protected void configure(HttpSecurity http) throws Exception {

http.authorizeRequests().antMatchers("/login").permitAll()

.antMatchers("/", "/\*todo\*/\*\*").access("hasRole('USER')").and()

.formLogin();

}

}

**src/main/java/com/in28minutes/springboot/web/service/TodoService.java**

package com.in28minutes.springboot.web.service;

import java.util.ArrayList;

import java.util.Date;

import java.util.Iterator;

import java.util.List;

import org.springframework.stereotype.Service;

import com.in28minutes.springboot.web.model.Todo;

@Service

public class TodoService {

private static List<Todo> todos = new ArrayList<Todo>();

private static int todoCount = 3;

static {

todos.add(new Todo(1, "in28Minutes", "Learn Spring MVC", new Date(),

false));

todos.add(new Todo(2, "in28Minutes", "Learn Struts", new Date(), false));

todos.add(new Todo(3, "in28Minutes", "Learn Hibernate", new Date(),

false));

}

public List<Todo> retrieveTodos(String user) {

List<Todo> filteredTodos = new ArrayList<Todo>();

for (Todo todo : todos) {

if (todo.getUser().equalsIgnoreCase(user)) {

filteredTodos.add(todo);

}

}

return filteredTodos;

}

public Todo retrieveTodo(int id) {

for (Todo todo : todos) {

if (todo.getId()==id) {

return todo;

}

}

return null;

}

public void updateTodo(Todo todo){

todos.remove(todo);

todos.add(todo);

}

public void addTodo(String name, String desc, Date targetDate,

boolean isDone) {

todos.add(new Todo(++todoCount, name, desc, targetDate, isDone));

}

public void deleteTodo(int id) {

Iterator<Todo> iterator = todos.iterator();

while (iterator.hasNext()) {

Todo todo = iterator.next();

if (todo.getId() == id) {

iterator.remove();

}

}

}

}

**src/main/java/com/in28minutes/springboot/web/SpringBootFirstWebApplication.java**

package com.in28minutes.springboot.web;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.annotation.ComponentScan;

@SpringBootApplication

@ComponentScan("com.in28minutes.springboot.web")

public class SpringBootFirstWebApplication {

public static void main(String[] args) {

SpringApplication.run(SpringBootFirstWebApplication.class, args);

}

}

**src/main/resources/application.properties**

spring.mvc.view.prefix=/WEB-INF/jsp/

spring.mvc.view.suffix=.jsp

logging.level.org.springframework.web=INFO

**src/main/webapp/WEB-INF/jsp/common/footer.jspf**

<script src="webjars/jquery/1.9.1/jquery.min.js"></script>

<script src="webjars/bootstrap/3.3.6/js/bootstrap.min.js"></script>

<script

src="webjars/bootstrap-datepicker/1.0.1/js/bootstrap-datepicker.js"></script>

<script>

$('#targetDate').datepicker({

format : 'dd/mm/yyyy'

});

</script>

</body>

</html>

**src/main/webapp/WEB-INF/jsp/common/header.jspf**

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>

<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt"%>

<%@taglib uri="http://www.springframework.org/tags/form" prefix="form"%>

<html>

<head>

<title>First Web Application</title>

<link href="webjars/bootstrap/3.3.6/css/bootstrap.min.css"

rel="stylesheet">

</head>

<body>

**src/main/webapp/WEB-INF/jsp/common/navigation.jspf**

<nav role="navigation" class="navbar navbar-default">

<div class="">

<a href="http://www.in28minutes.com" class="navbar-brand">in28Minutes</a>

</div>

<div class="navbar-collapse">

<ul class="nav navbar-nav">

<li class="active"><a href="/">Home</a></li>

<li><a href="/list-todos">Todos</a></li>

</ul>

<ul class="nav navbar-nav navbar-right">

<li><a href="/logout">Logout</a></li>

</ul>

</div>

</nav>

**src/main/webapp/WEB-INF/jsp/error.jsp**

<%@ include file="common/header.jspf"%>

<%@ include file="common/navigation.jspf"%>

<div class="container">

An exception occurred! Please contact Support!

</div>

<%@ include file="common/footer.jspf"%>

**src/main/webapp/WEB-INF/jsp/list-todos.jsp**

<%@ include file="common/header.jspf" %>

<%@ include file="common/navigation.jspf" %>

<div class="container">

<table class="table table-striped">

<caption>Your todos are</caption>

<thead>

<tr>

<th>Description</th>

<th>Target Date</th>

<th>Is it Done?</th>

<th></th>

<th></th>

</tr>

</thead>

<tbody>

<c:forEach items="${todos}" var="todo">

<tr>

<td>${todo.desc}</td>

<td><fmt:formatDate value="${todo.targetDate}" pattern="dd/MM/yyyy"/></td>

<td>${todo.done}</td>

<td><a type="button" class="btn btn-success"

href="/update-todo?id=${todo.id}">Update</a></td>

<td><a type="button" class="btn btn-warning"

href="/delete-todo?id=${todo.id}">Delete</a></td>

</tr>

</c:forEach>

</tbody>

</table>

<div>

<a class="button" href="/add-todo">Add a Todo</a>

</div>

</div>

<%@ include file="common/footer.jspf" %>

**src/main/webapp/WEB-INF/jsp/todo.jsp**

<%@ include file="common/header.jspf" %>

<%@ include file="common/navigation.jspf" %>

<div class="container">

<form:form method="post" commandName="todo">

<form:hidden path="id" />

<fieldset class="form-group">

<form:label path="desc">Description</form:label>

<form:input path="desc" type="text" class="form-control"

required="required" />

<form:errors path="desc" cssClass="text-warning" />

</fieldset>

<fieldset class="form-group">

<form:label path="targetDate">Target Date</form:label>

<form:input path="targetDate" type="text" class="form-control"

required="required" />

<form:errors path="targetDate" cssClass="text-warning" />

</fieldset>

<button type="submit" class="btn btn-success">Add</button>

</form:form>

</div>

<%@ include file="common/footer.jspf" %>

**src/main/webapp/WEB-INF/jsp/welcome.jsp**

<%@ include file="common/header.jspf"%>

<%@ include file="common/navigation.jspf"%>

<div class="container">

Welcome ${name}!! <a href="/list-todos">Click here</a> to manage your

todo's.

</div>

<%@ include file="common/footer.jspf"%>

**src/test/java/com/in28minutes/springboot/web/SpringBootFirstWebApplicationTests.java**

package com.in28minutes.springboot.web;

import org.junit.Test;

import org.junit.runner.RunWith;

import org.springframework.boot.test.context.SpringBootTest;

import org.springframework.test.context.junit4.SpringRunner;

@RunWith(SpringRunner.class)

@SpringBootTest

public class SpringBootFirstWebApplicationTests {

@Test

public void contextLoads() {

}

}

**todo.txt**

Implementing Server Side Validation

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Command Bean or Form Backing Bean

Add Validation

Use Validation on Controller

Display Errors in View

Command Bean

~~~~~~~~~~~~

Controller

View - Spring Form Tags

LoginController -> adds name to model

welcome.jsp -> shows ${name}

TodoController -> redirects to list-todos.jsp

${name} is empty

Component, Service, Repository, Controller

Autowired

ComponentScan

Field dummyService in com.in28minutes.springboot.web.controller.LoginController

required a bean of type 'com.in28minutes.dummy.DummyService'

that could not be found.

Spring Boot Starter Parent

Spring Boot Starter Web

@SpringBootApplication

Auto Configuration

Dispatcher Servlet

/login => "login"

"login" => src/main/webapp/WEB-INF/jsp/login.jsp

Search for a view named "login"

/login => LoginController