**To Do App (web application)**

* Spring initializer
* Spring boot dev tools

Create a package(**package** com.supre.springboot.myfirstwebapp.hello) with the class (SayHelloController)

When we hit the url (say-hello) it will return Hello! What are you learning today?

**SayHelloController.java**

**package** com.supre.springboot.myfirstwebapp.hello;

**import** org.springframework.stereotype.Controller;

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

**import** org.springframework.web.bind.annotation.ResponseBody;

@Controller

**public** **class** SayHelloController {

@RequestMapping("say-hello")

@ResponseBody

**public** String sayHello() {

**return** "Hello! What are you learning today?";

}

@RequestMapping("say-hello-html")

@ResponseBody

**public** String sayHelloHtml() {

StringBuffer sb=**new** StringBuffer();

sb.append("<html>");

sb.append("<head>");

sb.append("<title>My first page</title>");

sb.append("</head>");

sb.append("<body>");

sb.append("My first html page");

sb.append("</body>");

sb.append("</html>");

**return** sb.toString();

}

//"say-hello-jsp" =>sayHello.jsp

@RequestMapping("say-hello-jsp")

**public** String sayHelloJsp() {

**return** "sayHello";

}

}

* We have to create the JSP in the specific folder
* (/src/main/resources/META-INF/resources/WEB-INF/jsp/sayHello.jsp)

Write the following in application.properties

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

spring.mvc.view.suffix=.jsp

Note: write the following dependency in pom.xml

<dependency>

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

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

<scope>provided</scope>

</dependency>

Now, the folder structure is:

com.supre.springboot.myfirstwebapp.hello (package)

SayHelloController(class)

com.supre.springboot.myfirstwebapp.login (package)

LoginController(class)

**LoginController.java**

**package** com.supre.springboot.myfirstwebapp.login;

**import** org.springframework.stereotype.Controller;

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

@Controller

**public** **class** LoginController {

@RequestMapping("login")

**public** String gotoLoginPage() {

**return** "login";

}

}

* Now write the following in the url (<http://localhost:8080/login?name=Supreet>)

**LoginController.java**

**package** com.supre.springboot.myfirstwebapp.login;

**import** org.springframework.stereotype.Controller;

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

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

@Controller

**public** **class** LoginController {

@RequestMapping("login")

**public** String gotoLoginPage(@RequestParam String name) {

System.***out***.println("Request param is "+name);

**return** "login";

}

}

Make the login.jsp in the specified location

We get the output (Request param is Supreet) in the springboot console.

* To put the name in the login.jsp to display it in browser we have to use model.

**LoginController.java**

**package** com.supre.springboot.myfirstwebapp.login;

**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 {

@RequestMapping("login")

**public** String gotoLoginPage(@RequestParam String name, ModelMap model) {

model.put("name", name);

System.***out***.println("Request param is "+name);

**return** "login";

}

}

**Login.jsp**

<html>

    <head>

        <title>Login Page</title>

        </head>

        <body>

           Welcome to the login page

           ${name}

        </body>

        </html>

* We put <http://localhost:8080/login?name=Supreet> in the browser url. And get the name in the browser.
* **LOGGING LEVELS:**
* logging.level.org.springframework=INFO
* logging.level.com.supre.springboot.myfirstwebapp.hello=debug (debugging for a particular package)

Note: it is not recommended to write syso …so use logger

**In application.properties**

logging.level.org.springframework=debug

**LoginController.java**

**package** com.supre.springboot.myfirstwebapp.login;

**import** org.slf4j.Logger;

**import** org.slf4j.LoggerFactory;

**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 {

**private** Logger logger =LoggerFactory.*getLogger*(getClass());

@RequestMapping("login")

**public** String gotoLoginPage(@RequestParam String name, ModelMap model) {

model.put("name", name);

logger.debug("Request paraam is {}", name);

logger.info("I waant this to printed at info level");

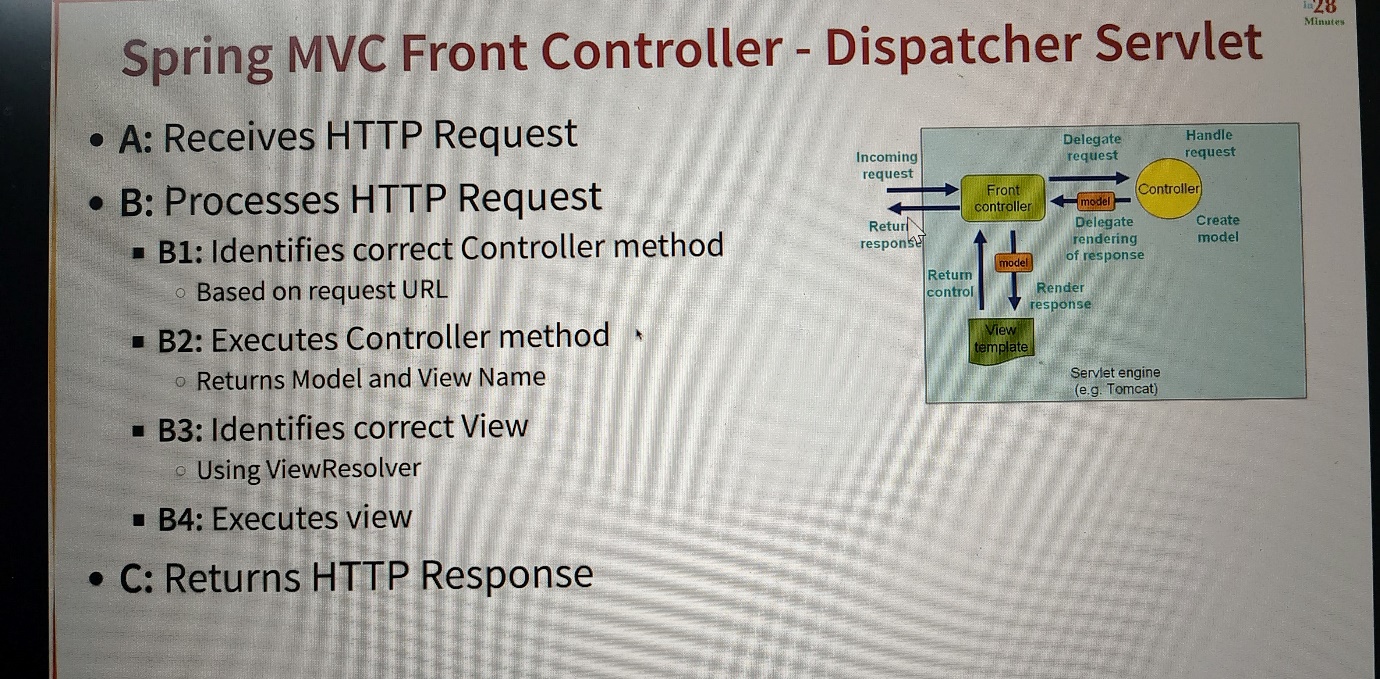
logger.warn("I waant this to printed at warn level");

//System.out.println("Request param is "+name);

**return** "login";

}

}



Now, we are building a simple login form :

**Login.jsp**

<html>

    <head>

        <title>Login Page</title>

        </head>

        <body>

           Welcome to the login page !

           <form method="post" >

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

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

            <input type="submit">

           </form>

        </body>

</html>

**Welcome.jsp**

<html>

    <head>

        <title>Welcome Page</title>

        </head>

        <body>

          <div>Welcome to your page</div>

        </body>

</html>

**LoginController.java**

**package** com.supre.springboot.myfirstwebapp.login;

**import** org.slf4j.Logger;

**import** org.slf4j.LoggerFactory;

**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** LoginController {

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

**public** String gotoLoginPage() {

**return** "login";

}

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

**public** String gotoWelcomePage() {

**return** "welcome";

}

}

**Note: Network -> doc->headers**

Now to pass the data to the welcome.jsp, we are using @RequestParam and ModelMap

**LoginController.java**

**package** com.supre.springboot.myfirstwebapp.login;

**import** org.slf4j.Logger;

**import** org.slf4j.LoggerFactory;

**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 gotoLoginPage() {

**return** "login";

}

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

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

model.put("name", name);

model.put("password", password);

**return** "welcome";

}

}

**Welcome.jsp**

<html>

    <head>

        <title>Welcome Page</title>

        </head>

        <body>

          <div>Welcome to your page</div>

          <div>NAME: ${name}</div>

          <div>password: ${password}</div>

        </body>

</html>

**NOTE:** NOW we are building a login page with authentication. If authentication is correct ,user will direct to welcome page

**LoginController.java**

**package** com.supre.springboot.myfirstwebapp.login;

**import** org.slf4j.Logger;

**import** org.slf4j.LoggerFactory;

**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;

@Controller

**public** **class** LoginController {

**private** AuthenticationService authenticationService ;

**public** LoginController(AuthenticationService authenticationService) {

**super**();

**this**.authenticationService = authenticationService;

}

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

**public** String gotoLoginPage() {

**return** "login";

}

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

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

**if**(authenticationService.authenticate(name, password)) {

model.put("name", name);

model.put("password", password);

//Authentication

//name-Supreet

//password- dummy

**return** "welcome";

}

model.put("errorMessage", "Invalid Credentials! Please try again");

**return** "login";

}

}

**AuthenticationService.java**

**package** com.supre.springboot.myfirstwebapp.login;

**import** org.springframework.stereotype.Component;

**import** org.springframework.stereotype.Service;

@Component

**public** **class** AuthenticationService {

**public** **boolean** authenticate(String username, String password) {

**boolean** isValidUserName = username.equalsIgnoreCase("Supreet");

**boolean** isValidPassword =password.equalsIgnoreCase("dummy");

**return** isValidUserName && isValidPassword;

}

}

**Welcome.jsp**

<html>

    <head>

        <title>Welcome Page</title>

        </head>

        <body>

           <pre>${errorMessage}</pre>

          <div>Welcome to your page</div>

          <div>NAME: ${name}</div>

          <div>password: ${password}</div>

        </body>

</html>

**com.supre.springboot.myfirstwebapp.todo -> Todo.java**

package com.supre.springboot.myfirstwebapp.todo;

import java.time.LocalDate;

public class Todo {

private int id;

private String username;

private String description;

private LocalDate targetDate;

private boolean done;

public Todo(int id, String username, String description, LocalDate targetDate, boolean done) {

super();

this.id = id;

this.username = username;

this.description = description;

this.targetDate = targetDate;

this.done = done;

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getUsername() {

return username;

}

public void setUsername(String username) {

this.username = username;

}

public String getDescription() {

return description;

}

public void setDescription(String description) {

this.description = description;

}

public LocalDate getTargetDate() {

return targetDate;

}

public void setTargetDate(LocalDate targetDate) {

this.targetDate = targetDate;

}

public boolean isDone() {

return done;

}

public void setDone(boolean done) {

this.done = done;

}

@Override

public String toString() {

return "Todo [id=" + id + ", username=" + username + ", description=" + description + ", targetDate="+ targetDate + ", done=" + done + "]";

}

}

**TodoController.java**

**package** com.supre.springboot.myfirstwebapp.todo;

**import** java.util.List;

**import** org.springframework.stereotype.Controller;

**import** org.springframework.ui.ModelMap;

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

@Controller

**public** **class** TodoController {

**private** TodoService todoService;

**public** TodoController(TodoService todoService) {

**super**();

**this**.todoService = todoService;

}

@RequestMapping("list-todos")

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

List<Todo> todos = todoService.findByUsername("in28minutes");

model.addAttribute("todos", todos);

**return** "listTodos";

}

}

**TodoService.java**

**package** com.supre.springboot.myfirstwebapp.todo;

**import** java.time.LocalDate;

**import** java.util.ArrayList;

**import** java.util.List;

**import** org.springframework.stereotype.Service;

@Service

**public** **class** TodoService {

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

**static** {

*todos*.add(**new** Todo(1, "in28minutes","Learn AWS", LocalDate.*now*().plusYears(1), **false** ));

*todos*.add(**new** Todo(2, "in28minutes","Learn DevOps",

LocalDate.*now*().plusYears(2), **false** ));

*todos*.add(**new** Todo(3, "in28minutes","Learn Full Stack Development",

LocalDate.*now*().plusYears(3), **false** ));

}

**public** List<Todo> findByUsername(String username){

**return** *todos*;

}

}

**Session vs Request Scope**

All requests from browser are handled by our web application deployed on a server .

**Request Scope:** Active for a single request ONLY

->Once the response is sent back, the request attributes will be removed from memory

->These cannot be used for future requests

->Recommended for most use cases

**Session Scope:** Details stored across multiple requests

-> Be careful about what you store in session (Takes additional memory as all details are stored on server) (@SessionAttributes(“name”) is used to use the name value across multiple requests)we have to write this on all the controllers where we want to use the particular value.

@Controller

@SessionAttributes("name")

**public** **class** LoginController {

//code

}

Now, we have to display the data of todos in table. Only ${todos} are not sufficient to do this. We have to use the jstl

<dependency>

<groupId>jakarta.servlet.jsp.jstl</groupId>

<artifactId>jakarta.servlet.jsp.jstl-api</artifactId>

</dependency>

<dependency>

<groupId>org.glassfish.web</groupId>

<artifactId>jakarta.servlet.jsp.jstl</artifactId>

</dependency>

**listTodos.jsp**

<%@ taglib prefix="c" uri="jakarta.tags.core" %>

<html>

  <body>

    <div>

      <h1>Your Todoss</h1>

      <table class="table">

        <thead>

          <tr>

            <th>id</th>

            <th>Description</th>

            <th>Target Date</th>

            <th>Is Done?</th>

          </tr>

        </thead>

        <tbody>

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

            <tr>

              <td>${todo.id}</td>

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

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

              <td>${todo.done}</td>

            </tr>

          </c:forEach>

        </tbody>

      </table>

    </div>

  </body>

</html>

**Adding dependencies for bootstrap**

<link href="webjars/bootstrap/5.1.3/css/bootstrap.min.css" rel="stylesheet" >

<script src="webjars/bootstrap/5.1.3/js/bootstrap.min.js"></script>

<script src="webjars/jquery/3.6.0/jquery.min.js"></script>

<dependency>

<groupId>org.webjars</groupId>

<artifactId>bootstrap</artifactId>

<version>5.1.3</version>

</dependency>

<dependency>

<groupId>org.webjars</groupId>

<artifactId>jquery</artifactId>

<version>3.6.0</version>

</dependency>

**listTodos.jsp**

<%@ taglib prefix="c" uri="jakarta.tags.core" %>

<html>

    <head>

        <link href="webjars/bootstrap/5.1.3/css/bootstrap.min.css" rel="stylesheet" >

        <title>List Todos Page</title>

    </head>

    <body>

        <div class="container">

            <h1>Your Todos</h1>

            <table class="table">

                <thead>

                    <tr>

                        <th>id</th>

                        <th>Description</th>

                        <th>Target Date</th>

                        <th>Is Done?</th>

                    </tr>

                </thead>

                <tbody>

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

                        <tr>

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

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

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

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

                        </tr>

                    </c:forEach>

                </tbody>

            </table>

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

        </div>

        <script src="webjars/bootstrap/5.1.3/js/bootstrap.min.js"></script>

        <script src="webjars/jquery/3.6.0/jquery.min.js"></script>

    </body>

</html>

**Login.jsp**

<html>

    <head>

        <title>Login Page</title>

        </head>

        <body>

            <div class="container">

           <h1>Login</h1>

           <pre>

            ${errorMessage}

           </pre>

           <form method="post" >

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

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

            <input type="submit">

           </form>

        </div>

        </body>

</html>

**Welcome.jsp**

<html>

    <head>

        <title>Welcome Page</title>

        </head>

        <body>

          <div class="container">

            <h1>Welcome ${name}</h1>

            <hr>

            <a href="list-todos">Manage</a> your todoss

        </div>

        </body>

</html>

**TODO.JSP**

<%@ taglib prefix="c" uri="jakarta.tags.core" %>

<html>

    <head>

        <link href="webjars/bootstrap/5.1.3/css/bootstrap.min.css" rel="stylesheet" >

        <title>Add Todos Page</title>

    </head>

    <body>

        <div class="container">

            <h1>Enter Todo Details</h1>

            <form method="post">

                Description: <input type="text" name="description"/>

                <input type="submit" class="btn btn-success"/>

            </form>

        </div>

        <script src="webjars/bootstrap/5.1.3/js/bootstrap.min.js"></script>

        <script src="webjars/jquery/3.6.0/jquery.min.js"></script>

    </body>

</html>

**TodoService.java**

**public** **void** addTodo(String username, String description, LocalDate targetDate, **boolean** done) {

Todo todo = **new** Todo(++*todosCount*,username,description,targetDate,done);

*todos*.add(todo);

}

**TodoController.java**

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

**public** String showNewTodoPage() {

**return** "todo";

}

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

**public** String addNewTodo(@RequestParam String description, ModelMap model) {

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

todoService.addTodo(username,description,LocalDate.*now*().plusYears(1), **false**);

**return** "redirect:list-todos";

}

**Validations with Spring Boot**

1**: Spring Boot Starter Validation** :pom.xml

2: **Command Bean (Form Backing Object)** :2-way binding (todo.jsp & TodoController.java)

3: **Add Validations to Bean:** Todo.java

4: **Display Validation Errors in the View:** todo.jsp

1…..<dependency>

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

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

</dependency>

2. we have to add validation to the field. We can do this directly to Todo.java without using the @RequestParam.so remove the @RequestParam from addNewTodo.

To configure the form backing object in todo.jsp

(search in google: spring form tag library documentation

<https://docs.spring.io/spring-framework/docs/3.2.x/spring-framework-reference/html/view.html>

paste(<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form" %>) in todo.jsp

add <form:form method="post" modelAttribute="todo"> in todo.jsp

value inside the modelAttribute is same as [**public** String addNewTodo( ModelMap model,Todo todo) ]in TodoController.java

Now we have to configure description element of todo.jsp :

We have to remove name and add path:

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

                Description: <input type="text" **path="description"/>**

                <input type="submit" class="btn btn-success"/>

            </form:form>

Now,changes are

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

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

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

Todo todo =**new** Todo(0,username, "", LocalDate.*now*().plusYears(1), **false**);

model.put("todo", todo);

**return** "todo";

}

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

**public** String addNewTodo( ModelMap model,Todo todo) {

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

todoService.addTodo(username, todo.getDescription(), LocalDate.*now*().plusYears(1), **false**);

**return** "redirect:list-todos";

}

**Todo.jsp**

<%@ taglib prefix="c" uri="jakarta.tags.core" %>

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

<html>

    <head>

        <link href="webjars/bootstrap/5.1.3/css/bootstrap.min.css" rel="stylesheet" >

        <title>Add Todos Page</title>

    </head>

    <body>

        <div class="container">

            <h1>Enter Todo Details</h1>

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

                Description: <form:input type="text" path="description" required="required"/>

                <form:input type="hidden" path="id"/>

                <form:input type="hidden" path="done"/>

                <input type="submit" class="btn btn-success"/>

            </form:form>

        </div>

        <script src="webjars/bootstrap/5.1.3/js/bootstrap.min.js"></script>

        <script src="webjars/jquery/3.6.0/jquery.min.js"></script>

    </body>

</html>

Note: if we don’t type the following it will through an error:

<form:input type="hidden" path="id"/>

<form:input type="hidden" path="done"/>

**Now adding the validations**

Write the following in todo.java:

@Size(min=10, message="Enter atleaast 10 charcters")

**private** String description;

add **@Valid** and **BindingResult result** in TodoController.java

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

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

**return** "todo";

}

//rest code

}

Also add the following in todo.jsp:

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

                Description: <form:input type="text" path="description" required="required"/>

**<form:errors path="description"/>**

                <form:input type="hidden" path="id"/>

                <form:input type="hidden" path="done"/>

                <input type="submit" class="btn btn-success"/>

            </form:form>

**Note: for adding css add cssClass in todo.jsp**

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

**Now delete operation**

**listTodos.jsp**

            <table class="table">

                <thead>

                    <tr>

                        <th>id</th>

                        <th>Description</th>

                        <th>Target Date</th>

                        <th>Is Done?</th>

**<th></th>**

                    </tr>

                </thead>

                <tbody>

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

                        <tr>

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

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

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

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

**<td><a href="delete-todo?id=${todo.id}" class="btn btn-warning">delete${todo.id}</a></td>**

                        </tr>

                    </c:forEach>

                </tbody>

            </table>

**TodoService.java**

**public** **void** deleteById(**int** id) {

Predicate<? **super** Todo> predicate =todo -> todo.getId()==id;

*todos*.removeIf(predicate);

}

**TodoController.java**

@RequestMapping("delete-todo")

**public** String deleteTodo(@RequestParam **int** id) {//RequestParam to capture the id

todoService.deleteById(id);

**return** "redirect:list-todos";

}

UPDATE

Listtodos.jsp

<th></th>

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

**TodoService.java**

public Todo findById(int id) {

Predicate<? super Todo> predicate = todo -> todo.getId() == id;

Todo todo = *todos*.stream().filter(predicate).findFirst().get();

return todo;

}

public void updateTodo(@Valid Todo todo) {

deleteById(todo.getId());

*todos*.add(todo);

}

**TodoController.java**

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

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

Todo todo = todoService.findById(id);

model.addAttribute("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";

}

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

todo.setUsername(username);

todoService.updateTodo(todo);

**return** "redirect:list-todos";

}

Now, adding the target date

<dependency>

<groupId>org.webjars</groupId>

<artifactId>bootstrap-datepicker</artifactId>

<version>1.9.0</version>

</dependency>

**Todo.jsp**

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

                <fieldset class="mb-3">

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

                    <form:input type="text" path="description" required="required"/>

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

                </fieldset>

                <fieldset class="mb-3">

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

                    <form:input type="text" path="targetDate" required="required"/>

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

                </fieldset>

<form:input type="hidden" path="id"/>

                <form:input type="hidden" path="done"/>

                <input type="submit" class="btn btn-success"/>

            </form:form>

**Application.properties**

**spring.mvc.format.date=yyyy-MM-dd**

Add the following in Todo.jsp to add css and js for date field:

<link href="webjars/bootstrap-datepicker/1.9.0/css/bootstrap-datepicker.standalone.min.css" rel="stylesheet" >

<script src="webjars/jquery/3.6.0/jquery.min.js"></script>

<script src="webjars/bootstrap-datepicker/1.9.0/js/bootstrap-datepicker.min.js"></script>

<script type="text/javascript">

$('#targetDate').datepicker({

format: 'yyyy-mm-dd'

});

</script>

**Final todo.jsp**

<%@ taglib prefix="c" uri="jakarta.tags.core" %>

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

<html>

    <head>

        <link href="webjars/bootstrap/5.1.3/css/bootstrap.min.css" rel="stylesheet" >

        <link href="webjars/bootstrap-datepicker/1.9.0/css/bootstrap-datepicker.standalone.min.css" rel="stylesheet" >

        <title>Add Todos Page</title>

    </head>

    <body>

        <div class="container">

            <h1>Enter Todo Details</h1>

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

                <fieldset class="mb-3">

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

                    <form:input type="text" path="description" required="required"/>

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

                </fieldset>

                <fieldset class="mb-3">

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

                    <form:input type="text" path="targetDate" required="required"/>

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

                </fieldset>

                <form:input type="hidden" path="id"/>

                <form:input type="hidden" path="done"/>

                <input type="submit" class="btn btn-success"/>

            </form:form>

        </div>

        <script src="webjars/bootstrap/5.1.3/js/bootstrap.min.js"></script>

        <script src="webjars/jquery/3.6.0/jquery.min.js"></script>

        <script src="webjars/bootstrap-datepicker/1.9.0/js/bootstrap-datepicker.min.js"></script>

        <script type="text/javascript">

            $('#targetDate').datepicker({

                format: 'yyyy-mm-dd'

            });

        </script>

    </body>

</html>

**Now: build the navigation bar**

Paste the below code inside body of **listTodos.java:**

<nav class="navbar navbar-expand-md navbar-light bg-light mb-3 p-1">

<a class="navbar-brand m-1" href="https://courses.in28minutes.com">in28minutes</a>

<div class="collapse navbar-collapse">

<ul class="navbar-nav">

<li class="nav-item"><a class="nav-link" href="/">Home</a></li>

<li class="nav-item"><a class="nav-link" href="/list-todos">Todos</a></li>

</ul>

</div>

<ul class="navbar-nav">

<li class="nav-item"><a class="nav-link" href="/logout">Logout</a></li>

</ul>

</nav>

Now: create a common folder(header.jspf,footer.jspf,navigation.jspf files inside it)

Some configurations are changed

**SPRING SECURITY**

Some cleanup as we have done some login previously.Now spring security will handle that.

Go to the login controller:

Delete **AuthenticationService.java**

**Delete login.jsp**

Change name of LoginController.java to WelcomeController.java

**WelcomeController.java** is like as below:

package com.supre.springboot.myfirstwebapp.login;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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;

@Controller

@SessionAttributes("name")

public class Welcome Controller {

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

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

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

**return** "welcome";

}

}

**SPRING SECURITY**

Add the dependency:\

<dependency>

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

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

<scope>test</scope>

</dependency>

->create a new class (SpringSecurityConfiguration) inside the package (com.supre.springboot.myfirstwebapp.security)

**SpringSecurityConfiguration.java**

**package** com.supre.springboot.myfirstwebapp.security;

**import** java.util.function.Function;

**import** org.springframework.context.annotation.Bean;

**import** org.springframework.context.annotation.Configuration;

**import** org.springframework.security.provisioning.InMemoryUserDetailsManager;

**import** org.springframework.security.core.userdetails.User;

**import** org.springframework.security.core.userdetails.UserDetails;

**import** org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

**import** org.springframework.security.crypto.password.PasswordEncoder;

@Configuration

**public** **class** SpringSecurityConfiguration {

@Bean

**public** InMemoryUserDetailsManager createUserDetailsManager() {

Function<String, String> passwordEncoder

= input -> passwordEncoder().encode(input);

UserDetails userDetails = User.*builder*()

.passwordEncoder(passwordEncoder)

.username("in28minutes")

.password("dummy")

.roles("USER","ADMIN")

.build();

**return** **new** InMemoryUserDetailsManager(userDetails);

}

@Bean

**public** PasswordEncoder passwordEncoder() {

**return** **new** BCryptPasswordEncoder();

}

}

**WelcomeController.java (Refactoring and Removing Hardcoding of User Id(Supreet))**

@Controller

@SessionAttributes("name")

**public** **class** WelcomeController {

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

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

model.put("name", **getLoggedinUsername());**

**return** "welcome";

}

**private** String getLoggedinUsername() {

Authentication authentication =

SecurityContextHolder.*getContext*().getAuthentication();

**return** authentication.getName();

}

}

Welcome controller will take the value from the welcome page and then todoController will match the values for authentication

**TodoController.java**

@RequestMapping("list-todos")

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

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

List<Todo> todos = todoService.findByUsername**(username**);

model.addAttribute("todos", todos);

**return** "listTodos";

}

Above, we can see the findByUserName method. We can see the methos in TodoService.java

In the method, we are returning the todos page.we are not filtering it by username.

**TodoService.java**

public List<Todo> findByUsername(String username){

Predicate<? super Todo> predicate =

todo -> todo.getUsername().equalsIgnoreCase(username);

return todos.stream().filter(predicate).toList();

}

If username matches, then we will show the todo list.

**TodoController.java (write the code at last)**

**private** String getLoggedInUsername(ModelMap model) {

Authentication authentication =

SecurityContextHolder.*getContext*().getAuthentication();

**return** authentication.getName();

}

For adding multiple users, we have to change **SpringSecurityConfiguration.java**

**package** com.supre.springboot.myfirstwebapp.security;

**import** java.util.function.Function;

**import** org.springframework.context.annotation.Bean;

**import** org.springframework.context.annotation.Configuration;

**import** org.springframework.security.provisioning.InMemoryUserDetailsManager;

**import** org.springframework.security.core.userdetails.User;

**import** org.springframework.security.core.userdetails.UserDetails;

**import** org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

**import** org.springframework.security.crypto.password.PasswordEncoder;

@Configuration

**public** **class** SpringSecurityConfiguration {

@Bean

**public** InMemoryUserDetailsManager createUserDetailsManager() {

UserDetails userDetails1 = createNewUser("in28minutes", "dummy");

UserDetails userDetails2 = createNewUser("ranga", "dummydummy");

**return** **new** InMemoryUserDetailsManager(userDetails1, userDetails2);

}

**private** UserDetails createNewUser(String username, String password) {

Function<String, String> passwordEncoder

= input -> passwordEncoder().encode(input);

UserDetails userDetails = User.*builder*()

.passwordEncoder(passwordEncoder)

.username(username)

.password(password)

.roles("USER","ADMIN")

.build();

**return** userDetails;

}

@Bean

**public** PasswordEncoder passwordEncoder() {

**return** **new** BCryptPasswordEncoder();

}

}

**Springboot Starter JPA and H2**

Add the dependencies

<dependency>

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

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<dependency>

<groupId>com.h2database</groupId>

<artifactId>h2</artifactId>

<scope>runtime</scope>

</dependency> //logging.level.org.springframework=debug

->paste (spring.datasource.url=jdbc:h2:mem:testdb) in application.properties

-> <http://localhost:8080/h2-console>

->Add the following to **SpringSecurityConfiguration.java**

//All URLs are protected

//A login form is shown for unauthorized requests

//CSRF disable

//Frames

@Bean

**public** SecurityFilterChain filterChain(HttpSecurity http) **throws** Exception {

http.authorizeHttpRequests(

auth -> auth.anyRequest().authenticated());

http.formLogin(*withDefaults*());

http.csrf().disable();

http.headers().frameOptions().disable();

**return** http.build();

}

* -> Add the Annotations in todo.jpa
* @Entity
* **public** **class** Todo {
* @Id
* @GeneratedValue
* **private** **int** id;

->create a file(data.sql) in src/main/resources

->write (spring.jpa.defer-datasource-initialization=true) in application.properties

->Write the following in data.sql

insert into todo (ID, USERNAME, DESCRIPTION, TARGET\_DATE, DONE)

values(10001,'supreet', 'Get AWS Certified', CURRENT\_DATE(), false);

insert into todo (ID, USERNAME, DESCRIPTION, TARGET\_DATE, DONE)

values(10002,'supreet', 'Get Azure Certified', CURRENT\_DATE(), false);

insert into todo (ID, USERNAME, DESCRIPTION, TARGET\_DATE, DONE)

values(10003,'supreet', 'Get GCP Certified', CURRENT\_DATE(), false);

insert into todo (ID, USERNAME, DESCRIPTION, TARGET\_DATE, DONE)

values(10004,'kumar', 'Learn DevOps', CURRENT\_DATE(), false);

->Now create the interface (TodoRepository) in (**package** com.supre.springboot.myfirstwebapp.todo)

**package** com.supre.springboot.myfirstwebapp.todo;

**import** org.springframework.data.jpa.repository.JpaRepository;

**public** **interface** TodoRepository **extends** JpaRepository<Todo, Integer> {

}

->create the TodoControllerJpa .java (copy of TodoController.java(as a backup) and comment the @Controller in TodoController.java)

->Add the following in TodoControllerJpa

**public** **class** TodoControllerJpa {

**private** TodoService todoService;

private TodoRepository todoRepository;

**public** TodoControllerJpa(TodoService todoService, **TodoRepository todoRepository**) {

**super**();

**this**.todoService = todoService;

this.todoRepository=todoRepository;

}

Note: todoRepository has different functions.those functions where aar enot there we can implement them in TodoRepository

**TodoRepository.java**

**public** **interface** TodoRepository **extends** JpaRepository<Todo, Integer> {

**public** List<Todo> findByUsername(String username);

}

**TodoController.jpa**

@RequestMapping("list-todos")

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

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

// List<Todo> todos = todoService.findByUsername(username);

List<Todo> todos = todoRepository.findByUsername(username);

model.addAttribute("todos", todos);

**return** "listTodos";

}

**Now to add todo with interface(todo button)**

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

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

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

**return** "todo";

}

String username = getLoggedInUsername(model);

todo.setUsername(username);

todoRepository.save(todo);

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

//todoService.addTodo(username, todo.getDescription(), todo.getTargetDate(), todo.isDone());

**return** "redirect:list-todos";

}

**Now to delete todo**

@RequestMapping("delete-todo")

**public** String deleteTodo(@RequestParam **int** id) {//RequestParam to capture the id

todoRepository.deleteById(id);

//todoService.deleteById(id);

**return** "redirect:list-todos";

}

**Update todo**

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

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

**Todo todo = todoRepository.findById(id).get();**

//Todo todo = todoService.findById(id);

model.addAttribute("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";

}

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

todo.setUsername(username);

**todoRepository.save(todo);**

//todoService.updateTodo(todo);

**return** "redirect:list-todos";

}

**TodoControllerJpa.java**

package com.supre.springboot.myfirstwebapp.todo;

import java.time.LocalDate;

import java.util.List;

import org.springframework.security.core.Authentication;

import org.springframework.security.core.context.SecurityContextHolder;

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 jakarta.validation.Valid;

@Controller

@SessionAttributes("name")

public class TodoControllerJpa {

//private TodoService todoService;

private TodoRepository todoRepository;

// public TodoControllerJpa(TodoService todoService, TodoRepository todoRepository) {

// super();

// this.todoService = todoService;

// this.todoRepository=todoRepository;

// }

public TodoControllerJpa( TodoRepository todoRepository) {

super();

this.todoRepository=todoRepository;

}

@RequestMapping("list-todos")

public String listAllTodos(ModelMap model) {

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

// List<Todo> todos = todoService.findByUsername(username);

List<Todo> todos = todoRepository.findByUsername(username);

model.addAttribute("todos", todos);

return "listTodos";

}

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

public String showNewTodoPage(ModelMap model) {

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

Todo todo =new Todo(0,username, "", LocalDate.now().plusYears(1), false);

model.put("todo", todo);

return "todo";

}

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

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

if(result.hasErrors()) {

return "todo";

}

String username = getLoggedInUsername(model);

todo.setUsername(username);

todoRepository.save(todo);

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

//todoService.addTodo(username, todo.getDescription(), todo.getTargetDate(), todo.isDone());

return "redirect:list-todos";

}

@RequestMapping("delete-todo")

public String deleteTodo(@RequestParam int id) {//RequestParam to capture the id

todoRepository.deleteById(id);

//todoService.deleteById(id);

return "redirect:list-todos";

}

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

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

Todo todo = todoRepository.findById(id).get();

//Todo todo = todoService.findById(id);

model.addAttribute("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";

}

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

todo.setUsername(username);

todoRepository.save(todo);

//todoService.updateTodo(todo);

return "redirect:list-todos";

}

private String getLoggedInUsername(ModelMap model) {

Authentication authentication =

SecurityContextHolder.getContext().getAuthentication();

return authentication.getName();

}

}