1.Event Register

```
<html>
<head>
<script>
    function registration()
        {

        var name= document.getElementById("t1").value;
        var email= document.getElementById("t2").value;
        var uname= document.getElementById("t3").value;
        var pwd= document.getElementById("t4").value;
        var cpwd= document.getElementById("t5").value;
        //email id expression code
        var pwd_expression = /^(?=.*?[A-Z])(?=.*?[a-z])(?=.*?[0-9])(?=.*?[#?!@$
%^&*-])/;
                var letters = /^[A-Za-z]+$/;
                var filter = /^([a-zA-Z0-9_\.\-])+\@(([a-zA-Z0-9\-])+\.)+([a-zA-
Z0-9]{2,4})+$/;

                if(name=='')
                {
                        alert('Please enter your name');
                }
                else if(!letters.test(name))
                {
                        alert('Name field required only alphabet characters');
                }
                else if(email=='')
                {
                        alert('Please enter your user email id');
                }
                else if (!filter.test(email))
                {
                        alert('Invalid email');
                }
                else if(uname=='')
                {
                        alert('Please enter the user name.');
                }
                else if(!letters.test(uname))
                {
                        alert('User name field required only alphabet
characters');
                }
                else if(pwd=='')
                {
                        alert('Please enter Password');
                }
                else if(cpwd=='')
                {
                        alert('Enter Confirm Password');
                }
                else if(!pwd_expression.test(pwd))
                {
                        alert ('Upper case, Lower case, Special character and
Numeric letter are required in Password filed');
                }
                else if(pwd != cpwd)
                {
                        alert ('Password not Matched');
                }
                else if(document.getElementById("t5").value.length < 6)
                {
                        alert ('Password minimum length is 6');
                }
                else if(document.getElementById("t5").value.length > 12)
                {
                        alert ('Password max length is 12');
                }
                else
                {

               alert('Thank You for Login & You are Redirecting to Campuslife
Website');

                        window.location.pathname="G://welcome.html";
                }
        }
        function clearFunc()
        {
                document.getElementById("t1").value="";
                document.getElementById("t2").value="";
                document.getElementById("t3").value="";
                document.getElementById("t4").value="";
                document.getElementById("t5").value="";
        }


</script>


  <title>Welcome To Registration Form</title>
</head>

        <body>

        <h2>Create Your Account</h2>

        <table>
        <tr>
        <td >Enter Name :</td>
        <td><input type="text" id="t1" class="tb" /></td>
        </tr>
        <tr>
        <td >Enter Email ID :</td>
        <td><input type="text" id="t2" class="tb" /></td>
        </tr>
        <tr>
        <td>Enter Username :</td>
        <td><input type="text" id="t3" class="tb" /></td>
        </tr>
        <tr>
        <td >Enter Password :</td>
        <td><input type="password" id="t4" class="tb" /></td>
        </tr>
        <tr>
        <td>Enter Confirm Password :</td>
        <td><input type="password" id="t5" class="tb" /></td>
        </tr>
        <tr>
        <td></td>
        <td>
        <input type="reset" value="Clear Form" onclick="clearFunc()" id="res"
class="btn" />
        <input type="submit" value="Create Account" class="btn"
onclick="registration()" /></td>
```

```
        </tr>
        </table>
        </body>
        </html>
```

welcome.html

```
<html>
<head>
<title>Welcome</title>
</head>

<body>
You have Successfully registered!
</body>
</html>
```

2.Task

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<link rel="stylesheet" href="styles.css">
<title>Task Manager</title>
</head>
<body>
<div class="container">
<h1>Task Manager</h1>
<div class="task-list">
<ul id="task-list"></ul>
</div>
<div class="add-task">
<input type="text" id="task-input" placeholder="Add a new task">
<button id="add-button">Add</button>
</div>
</div>
<script type="module" src="app.js"></script>
</body>
</html>
```

```
// task.js
export class Task {
constructor(id, text) {
this.id = id;
this.text = text;
}
}
```

```
// app.js
import { Task } from './task.js';
class TaskManager {
constructor() {
```

```
this.tasks = JSON.parse(localStorage.getItem('tasks')) || [];
this.taskList = document.getElementById('task-list');
this.taskInput = document.getElementById('task-input');
this.addButton = document.getElementById('add-button');
this.addButton.addEventListener('click', this.addTask.bind(this));
this.renderTasks();
}
addTask() {
const taskText = this.taskInput.value.trim();
if (taskText === '') return;
const taskId = new Date().getTime();
const task = new Task(taskId, taskText);
this.tasks.push(task);
this.saveTasks();
this.renderTasks();
this.taskInput.value = '';
}
saveTasks() {
localStorage.setItem('tasks', JSON.stringify(this.tasks));
}
renderTasks() {
this.taskList.innerHTML = '';
this.tasks.forEach(task => {
const li = document.createElement('li');
li.innerHTML = `<span>${task.text}</span><button
data-id="${task.id}">Delete</button>`;
this.taskList.appendChild(li);
li.querySelector('button').addEventListener('click', this.deleteTask.bind(this));
});
}

deleteTask(event) {
const taskId = parseInt(event.target.getAttribute('data-id'));
this.tasks = this.tasks.filter(task => task.id !== taskId);
this.saveTasks();
this.renderTasks();
}
}
const taskManager = new TaskManager();
```

3.webcontroller

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>AJAX</title>
<style>
*{
font-family: arial;
text-align:center;
}
```

```
</style>
</head>
<body>
<div id="main">
<h1>Current time: <span id="time"></span></h1>
<button onclick="getTime()">Update time</button>
</div>
<script>
var t = document.querySelector("#time");
const getTime = ()=>{
fetch("/time").then(async(res)=>{
console.log()
t.innerHTML = await res.text();
})
}
</script>
</body>
</html>
```

Webcontroller

```
package com.example.Ex3.controllers;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
@Controller
public class WebController {
@GetMapping
public String index(){
return "index";
}

}
```

API controller

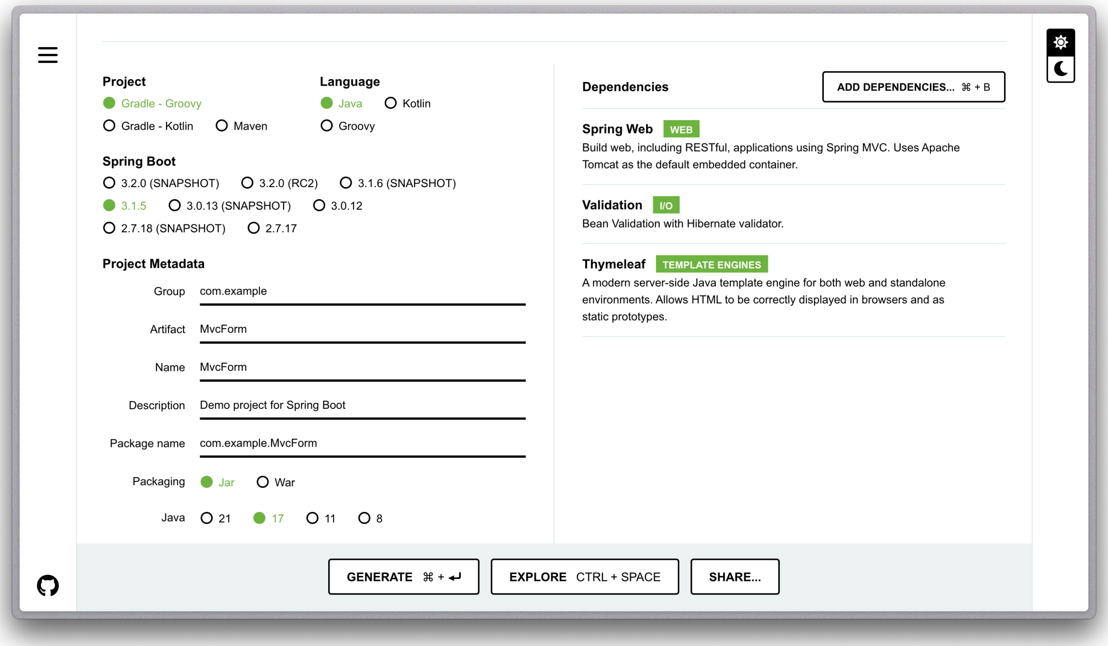
```
package com.example.Ex3.controllers;
```

EX-3 3

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;
import java.time.LocalDateTime;
@RestController()
public class ApiController {
Logger logger = LoggerFactory.getLogger(ApiController.class);
@GetMapping("/time")
public String time(){
logger.info("API is accessed : "+ LocalDateTime.now().toString());
return LocalDateTime.now().toString();
}
}
```

EX-5 : Validation, File upload and session tracking. (1)

1. Generate spring project with required dependencies



Project
[✓ Gradle - Groovy] Gradle - Kotlin Maven

Language
[✓ Java] Kotlin Groovy

Spring Boot
3.2.0 (SNAPSHOT) 3.2.0 (RC2) 3.1.6 (SNAPSHOT) [✓ 3.1.5] 3.0.13 (SNAPSHOT) 3.0.12 2.7.18 (SNAPSHOT) 2.7.17

Project Metadata

Group	com.example
Artifact	MvcForm
Name	MvcForm
Description	Demo project for Spring Boot
Package name	com.example.MvcForm
Packaging	[✓ Jar] War
Java	21 [✓ 17] 11 8

Dependencies

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

Validation I/O
Bean Validation with Hibernate validator.

Thymeleaf TEMPLATE ENGINES
A modern server-side Java template engine for both web and standalone environments. Allows HTML to be correctly displayed in browsers and as static prototypes.

2. Open the generated project in IDE.
3. Create a validation.html file in **"src > main > resources > templates"**

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">

  <head>
    <title>Validation</title>
  </head>
  <body>
    <form action="#" th:action="@{/validation/process}" th:object="${formPayload}" method="post">
        <table>
          <tr>
            <td>Name:</td>
            <td><input type="text" th:field="*{name}" /></td>
            <td th:if="${#fields.hasErrors('name')}" th:errors="*{name}">Name Error</td>
          </tr>
          <tr>
            <td>Email:</td>
            <td><input type="text" th:field="*{email}" /></td>
            <td th:if="${#fields.hasErrors('email')}" th:errors="*{email}">Email Error</td>
          </tr>
          <tr>
            <td><button type="submit">Submit</button></td>
          </tr>
        </table>
      </form>
      <p th:text="${message}"></p>
  </body>

</html>
```

4. Create a session.html file in **“src > main > resources > templates”**

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">

  <head>
    <title>Session</title>
  </head>
  <body>
      <p th:text="${page_count}"></p>
  </body>

</html>
```

5. Create a file.html file in **“src > main > resources > templates”**

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
  <body>
    <form action="#" th:action="@{/file/process}" th:object="${fileForm}" method="post" enctype="multipart/form-data">
      <table>
          <td>Image:</td>
          <td>
            <input type="file" th:field="*{file}" />
          </td>
        </tr>
        <tr>
          <td><button type="submit">Submit</button></td>
        </tr>
      </table>
      <p th:text="${message}"></p>

    </form>
  </body>
</html>
```

6. Create a new package “models”
7. Create a model class FormPayload inside models package

```
import jakarta.validation.constraints.Email;
import jakarta.validation.constraints.NotEmpty;
import jakarta.validation.constraints.NotNull;
import jakarta.validation.constraints.Size;

public class FormPayload {


  @NotEmpty
  @Size(min = 1)
  private String name;

  @NotEmpty
  @Email
  private String email;



  public String getName() {
    return name;
  }
  public void setName(String name) {
    this.name = name;
  }
  public String getEmail() {
    return email;
  }
  public void setEmail(String email) {
    this.email = email;
  }
}
```

8. Create a model class FileForm inside models package

```
import org.springframework.web.multipart.MultipartFile;

public class FileForm {

  private MultipartFile file;

  public MultipartFile getFile() {
    return file;
  }

  public void setFile(MultipartFile file) {
    this.file = file;
  }
```

9. Create a new package “controllers”
10. Create a new class FileController inside controllers package

```
import java.io.File;
import java.io.IOException;

import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;

import com.example.Mvc.models.FileForm;

@Controller
public class FileController {

  @GetMapping("/file")
  public String index(FileForm fileForm) {

    return "file";
  }


  @PostMapping("/file/process")
  public String uploadFile(FileForm fileForm,Model model) throws IllegalStateException, IOException {


    fileForm.getFile().transferTo(new File("/Users/oswinjerome/Projects/MCA/test.jpg"));


    model.addAttribute("message","File uploaded successfully");


    return "file";

  }

}
```

11. Create a new class SessionController inside controllers package

```
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;

import jakarta.servlet.http.HttpSession;

@Controller
public class SessionController {
  @GetMapping("/session")
  public String index(HttpSession session,Model model) {

    int pageCount = Integer.valueOf(session.getAttribute("page_count")==null ? "0" :session.getAttribute("page_count").toString())

    session.setAttribute("page_count", pageCount + 1);

    model.addAttribute("page_count", "You have visited "+ (pageCount+1+" times"));

    return "session";
  }

}
```

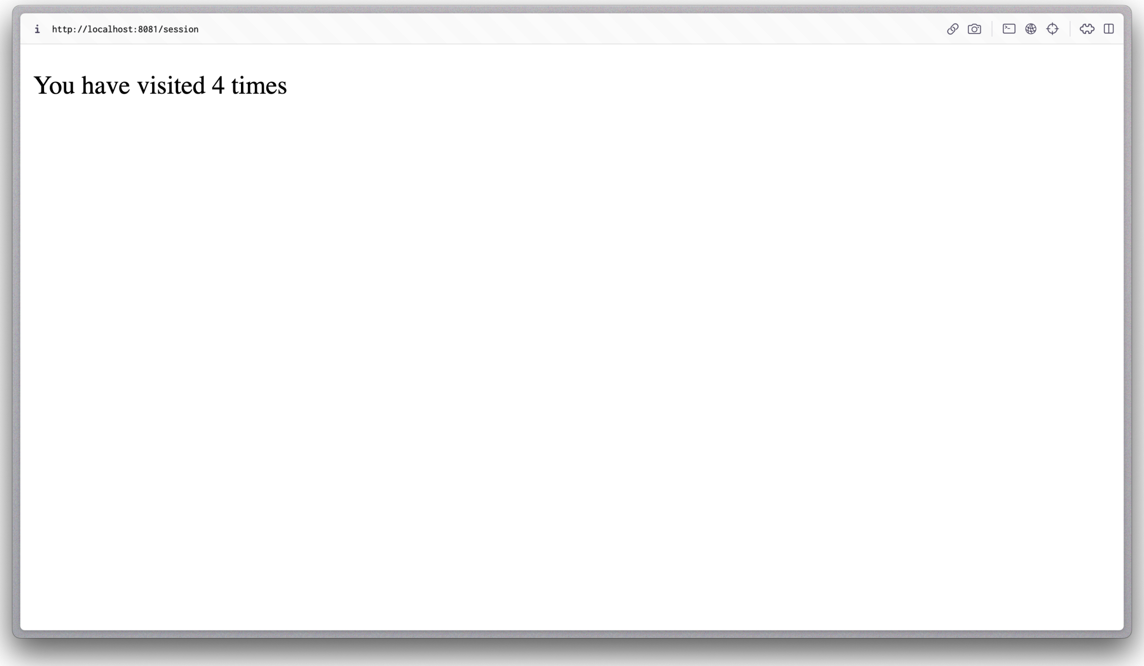
12. Create a new class ValidationController inside controllers package

```
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;

import com.example.Mvc.models.FormPayload;

import jakarta.validation.Valid;

@Controller
public class ValidationController {


  @GetMapping("/validation")
  public String index(FormPayload formPayload) {

    return "validation";
  }


  @PostMapping("/validation/process")
  public String processForm( @Valid FormPayload formPayload, BindingResult bindingResult, Model model) {


    if(bindingResult.hasErrors()) {

      return "validation";
    }

    model.addAttribute("message", "Form is valid");

    return "validation";

  }

}
```

13. Run the application and access
    a. http://localhost:8080/validation
    b. http://localhost:8080/file
    c. http://localhost:8080/session

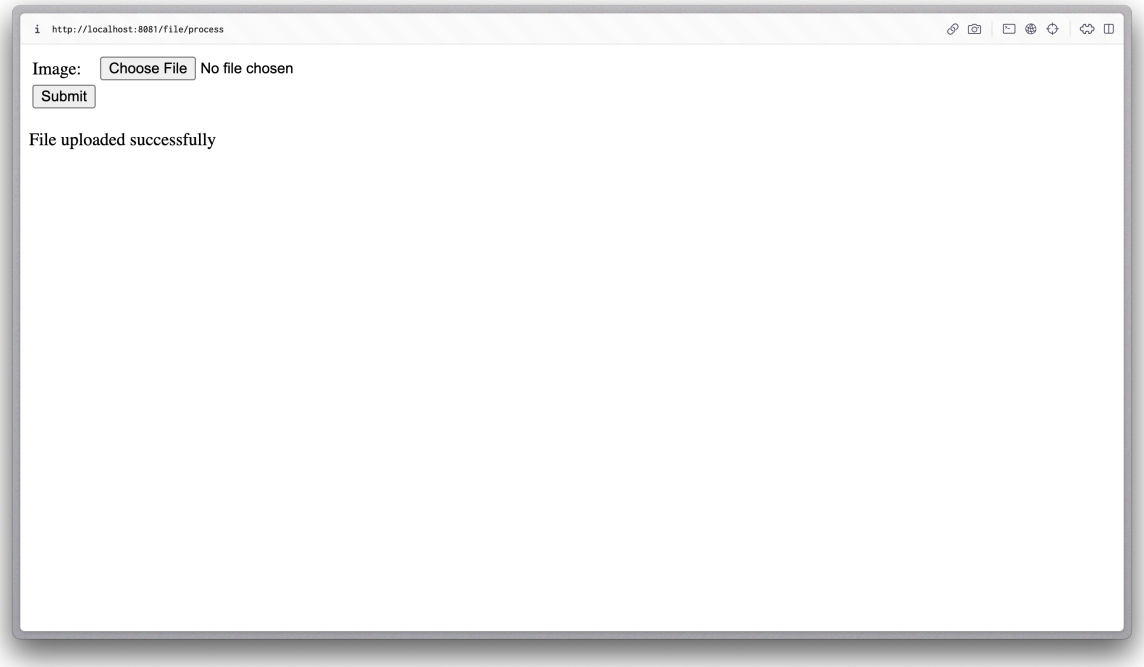
Output

1. http://localhost:8080/session


http://localhost:8081/session

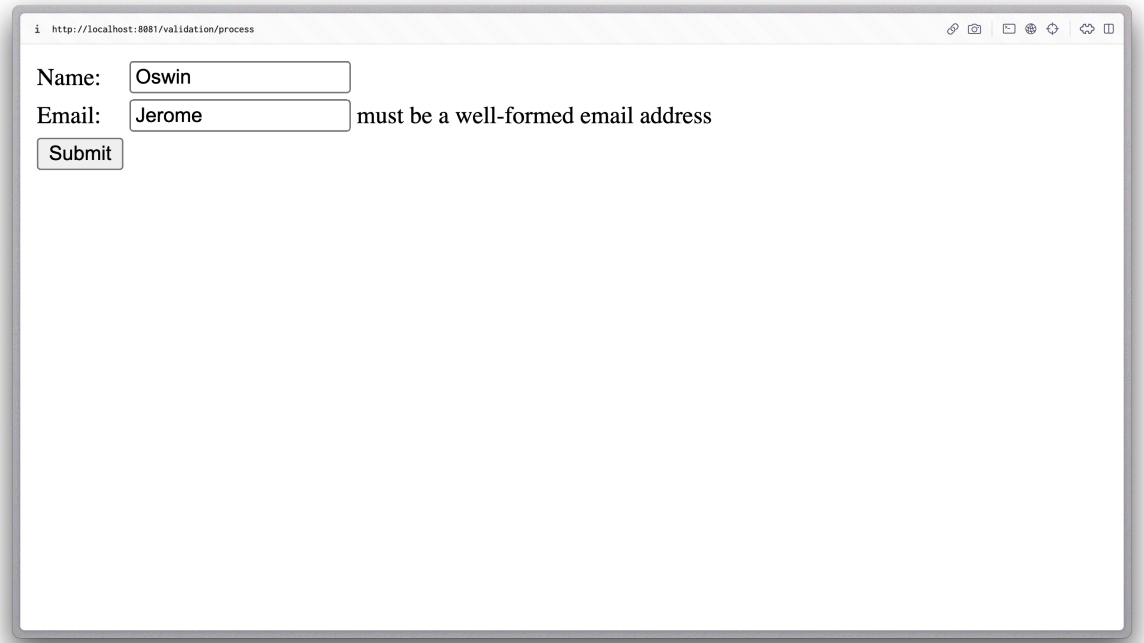
You have visited 4 times

2. http://localhost:8080/file


http://localhost:8081/file/process

File uploaded successfully

3. http://localhost:8080/validation



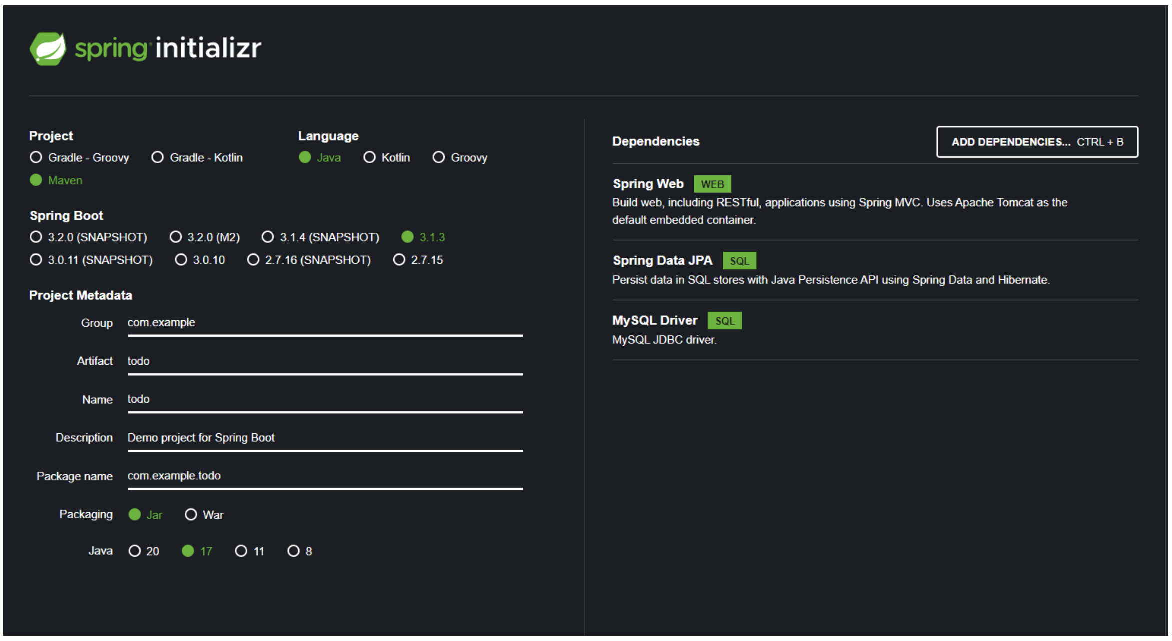
http://localhost:8081/validation/process

Name: Oswin

Email: Jerome must be a well-formed email address

EX-8 : Data JPA (Paging and Searching)

1. Generate spring project with required dependencies



spring initializr

Project
Gradle - Groovy Gradle - Kotlin [✓ Maven]

Language
[✓ Java] Kotlin Groovy

Spring Boot
3.2.0 (SNAPSHOT) 3.2.0 (M2) 3.1.4 (SNAPSHOT) [✓ 3.1.3]
3.0.11 (SNAPSHOT) 3.0.10 2.7.16 (SNAPSHOT) 2.7.15

Project Metadata

Group	com.example
Artifact	todo
Name	todo
Description	Demo project for Spring Boot
Package name	com.example.todo
Packaging	[✓ Jar] War
Java	20 [✓ 17] 11 8

Dependencies

ADD DEPENDENCIES... CTRL + B

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

Spring Data JPA SQL
Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.

MySQL Driver SQL
MySQL JDBC driver.

2. Open the genetated project in IDE.
3. Configure database details in "application.properties" file

```
spring.jpa.hibernate.ddl-auto=update
spring.datasource.url=jdbc:mysql://${MYSQL_HOST:localhost}:3306/todo_db
spring.datasource.username=root
spring.datasource.password=
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.jpa.show-sql = true
```

4. Create a new package "models" inside the main package
5. Create a java class "`TodoItem`" inside models package.

```
package com.college.todo.models;


import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;

@Entity
public class TodoItem {


    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;
    private String title;
    private String description;

    public TodoItem(){}

   public TodoItem(String title, String description){

        this.title = title;
        this.description = description;
    }

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }



}
```

6. Create a new package “repos” inside the main package
7. Create a java interface “`TodoRepository`” inside controllers package.

```
package com.college.todo.repos;

import com.college.todo.models.TodoItem;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.CrudRepository;
import org.springframework.data.repository.PagingAndSortingRepository;

import java.util.List;

public interface TodoRepository extends CrudRepository<TodoItem,Long>, PagingAndSortingRepository<TodoItem,Long> {

    Page<TodoItem> findByTitleContaining(String title, Pageable pageable);

}
```

8. Create a new package “controllers” inside the main package
9. Create a java class “WebController” inside controllers package.

```
package com.college.todo.controllers;


import com.college.todo.models.TodoItem;
import com.college.todo.repos.TodoRepository;
import org.springframework.beans.BeanUtils;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Bean;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.PageRequest;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.*;

@Controller
public class WebController {

    @Autowired
    TodoRepository todoRepository;

    @GetMapping
    public String getTodos(@RequestParam(defaultValue = "1") int page,@RequestParam(defaultValue = "") String query , Model model)
```

```
        Page<TodoItem>  todos = todoRepository.findByTitleContaining(query, PageRequest.of(page-1,4));

        model.addAttribute("todos",todos);

        return "todos";
    }

    @GetMapping("/create")
    public String createPage(){
        return "create";
    }

    @GetMapping("/todo/{id}")
    public String deleteTodo(@PathVariable Long id){
        todoRepository.deleteById(id);
        return "redirect:/";
    }



    @PostMapping("/")
    public String saveTodo(TodoRec todoRec){

        TodoItem todo = new TodoItem();

        BeanUtils.copyProperties(todoRec,todo);

        todo = todoRepository.save(todo);

        return "redirect:/";
    }


    record TodoRec(String title,String description){}

}
```

10. Create a todos.html file in **“src > main > resources > templates”**

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <link rel="stylesheet" href="/style.css" />
</head>
<body>
    <div>

        <a href="/create">Create Todo</a>

    </div>


    <form method="get" action="/">
        <div>
            <label for="query">Title</label>
            <input name="query" type="text" id="query" />
        </div>
        <div>
            <input  type="submit"  value="SEARCH"/>
        </div>
    </form>

    <section id="todos">
        <div class="todo" th:each="todo :${todos}">
            <div>
                <h3 th:text="${todo.title}">Title</h3>
                <p th:text="${todo.description}">Description</p>
            </div>
            <div>
                <a th:href="@{/todo/{id}(id=${todo.id})}">DELETE</a>
            </div>
        </div>
    </section>
    <ul id="pages">
        <li  th:each="i: ${#numbers.sequence(1, todos.getTotalPages())}" th:if="${todos.getTotalPages()!=0}">
            <a th:href="@{/?page={page}(page=${i})}" th:text="${i}"></a>
        </li>
    </ul>
```

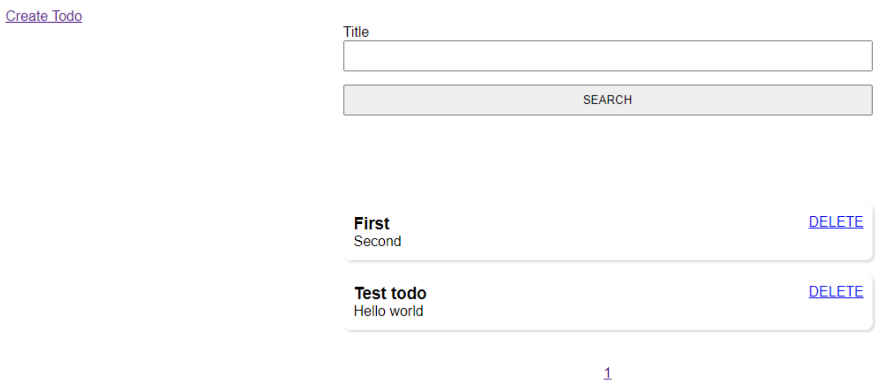
```
</body>
</html>
```

11. Create a create.html file in **“src > main > resources > templates”**

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <link rel="stylesheet" href="/style.css" />
</head>
<body>
<div>

    <a href="/">List Todo</a>

</div>
<form method="post" action="/">
    <div>
        <label for="title">Title</label>
        <input name="title" type="text" id="title" />
    </div>
    <div>
        <label for="description">Description</label>
        <input name="description" type="text" id="description" />
    </div>
    <div>
        <input  type="submit" />
    </div>
</form>

</body>
</html>
```

12. Run the application and access http://localhost:8080

Output



First
Second
DELETE

Test todo
Hello world
DELETE