**Case Study: Simple Food Application (No Database)**

Creating a simple food application using Spring Boot with Thymeleaf and without a database involves setting up a web application where you can manage food items (e.g., view a list, add items, etc.) in memory. Here’s a basic case study outline:

**Project Structure**

css

Copy code

food-application

├── src

│ ├── main

│ │ ├── java

│ │ │ └── com

│ │ │ └── example

│ │ │ └── foodapplication

│ │ │ ├── FoodApplication.java

│ │ │ ├── controller

│ │ │ │ └── FoodController.java

│ │ │ ├── model

│ │ │ │ └── FoodItem.java

│ │ │ └── service

│ │ │ └── FoodService.java

│ │ └── resources

│ │ ├── templates

│ │ │ ├── food-list.html

│ │ │ └── add-food.html

│ │ └── application.properties

└── pom.xml

**Step 1: Create the Spring Boot Application**

**FoodApplication.java**

java

Copy code

package com.example.foodapplication;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class FoodApplication {

public static void main(String[] args) {

SpringApplication.run(FoodApplication.class, args);

}

}

**Step 2: Create the Model**

**FoodItem.java**

java

Copy code

package com.example.foodapplication.model;

public class FoodItem {

private String name;

private String description;

// Constructors, getters, and setters

public FoodItem(String name, String description) {

this.name = name;

this.description = description;

}

public String getName() {

return name;

}

public String getDescription() {

return description;

}

}

**Step 3: Create the Service**

**FoodService.java**

java

Copy code

package com.example.foodapplication.service;

import com.example.foodapplication.model.FoodItem;

import org.springframework.stereotype.Service;

import java.util.ArrayList;

import java.util.List;

@Service

public class FoodService {

private final List<FoodItem> foodItems = new ArrayList<>();

public List<FoodItem> getAllFoodItems() {

return foodItems;

}

public void addFoodItem(FoodItem foodItem) {

foodItems.add(foodItem);

}

}

**Step 4: Create the Controller**

**FoodController.java**

java

Copy code

package com.example.foodapplication.controller;

import com.example.foodapplication.model.FoodItem;

import com.example.foodapplication.service.FoodService;

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

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

@Controller

public class FoodController {

@Autowired

private FoodService foodService;

@GetMapping("/")

public String listFood(Model model) {

model.addAttribute("foodItems", foodService.getAllFoodItems());

return "food-list";

}

@GetMapping("/add-food")

public String showAddFoodForm() {

return "add-food";

}

@PostMapping("/add-food")

public String addFood(@RequestParam String name, @RequestParam String description) {

FoodItem foodItem = new FoodItem(name, description);

foodService.addFoodItem(foodItem);

return "redirect:/";

}

}

**Step 5: Create Thymeleaf Templates**

**food-list.html**

html

Copy code

<!DOCTYPE html>

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

<head>

<title>Food List</title>

</head>

<body>

<h1>Food Items</h1>

<table>

<tr>

<th>Name</th>

<th>Description</th>

</tr>

<tr th:each="food : ${foodItems}">

<td th:text="${food.name}"></td>

<td th:text="${food.description}"></td>

</tr>

</table>

<a href="/add-food">Add Food Item</a>

</body>

</html>

**add-food.html**

html

Copy code

<!DOCTYPE html>

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

<head>

<title>Add Food</title>

</head>

<body>

<h1>Add Food Item</h1>

<form action="/add-food" method="post">

<label for="name">Name:</label>

<input type="text" id="name" name="name" required>

<label for="description">Description:</label>

<input type="text" id="description" name="description" required>

<button type="submit">Add Food Item</button>

</form>

<a href="/">Back to Food List</a>

</body>

</html>

**Step 6: Configure application.properties**

You may not need specific properties for this simple case, but you can customize server settings if necessary.

**Step 7: Run the Application**

1. Build and run your Spring Boot application.
2. Visit http://localhost:8080/ to view the food list and add new food items.

**Summary**

This simple Spring Boot web application allows users to view a list of food items and add new ones without using a database, leveraging in-memory storage through a service layer. You can extend this application further by adding more features like editing and deleting food items or improving the UI.