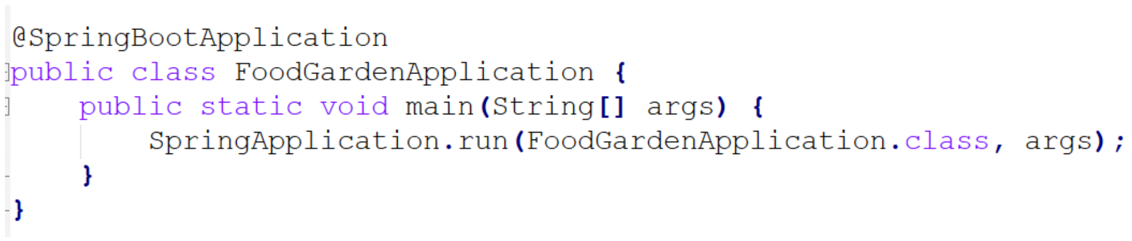
Explainer Document: Java and HTML Files in the foodGarden Spring Boot Project

This guide helps you understand how the Java and HTML files in the foodGarden project work together to create a basic Spring Boot web app. Each file plays a role in displaying content or processing user input.

Java Files Explained

1. FoodGardenApplication.java

This is the **main class** that starts your entire Spring Boot app.



```java
@SpringBootApplication
public class FoodGardenApplication {
    public static void main(String[] args) {
        SpringApplication.run(FoodGardenApplication.class, args);
    }
}
```

- This tells Spring Boot to auto-configure and start running the project.
- When you run mvn spring-boot:run, this file is executed.

2. HomeController.java

This class handles navigation to the basic pages (/, /about, /garden).

```java
@Controller
public class HomeController {

    @GetMapping("/")
    public String home() {
        return "homepage";  // Maps to src/main/resources/templates/homepage.html
    }

    @GetMapping("/about")
    public String about() {
        return "about";  // Maps to src/main/resources/templates/about.html
    }

    @GetMapping("/garden")
    public String gardenForm() {
        return "garden_form";  // Maps to src/main/resources/templates/garden_form.html
    }
}
```

- @Controller makes this class a web controller.
- Each method returns the name of an HTML file in the templates folder.
- These pages are static: they just show information or provide a form.

3. RecommendationController.java

This file processes the garden area input and decides which plant to recommend.

```
@Controller
public class RecommendationController
{

    @PostMapping("/recommendation")
    public String getRecommendation(@RequestParam("area") int area, Model model)
    {
        // Determine the recommended plants and image
        String[] recommendation = getPlantRecommendation(area);

        // Add data to the model
        model.addAttribute("area", area);
        model.addAttribute("plant", recommendation[0]);
        model.addAttribute("image", recommendation[1]);

        // Return the view name (Thymeleaf template)
        return "recommendation";
    }

    private String[] getPlantRecommendation(int area)
    {
        if (area < 10)
        {
            return new String[]{"Herbs like basil, mint, or parsley.", "herbs.jpg"};
        }
        else if (area >= 10 && area < 50)
        {
            return new String[]{"Small vegetables like tomatoes, lettuce, or carrots.", "vegetables.jpg"};
        }
        else
        {
            return new String[]{"Larger vegetables or even small fruit trees like apples or peaches.", "large_garden.jpg"};
        }
    }
}
```

- Handles the POST request from the garden form.
- Based on the area input, it chooses a suitable plant (or set of plants).
- Passes this data to the recommendation.html page.

HTML Template Files Explained

These are found in src/main/resources/templates/.

1. homepage.html

- The welcome page.
- Explains the project idea: food gardens for food security.
- Shows two case studies with images.
- Links to About and Garden Form.

2. about.html

- Page to describe the group.
- Includes an image of the group.
- Customizable by students to list members or fun facts.

3. garden_form.html

- Interactive form where the user enters garden area.
- Sends the number to /recommendation using POST.
- This is where the backend logic kicks in!

4. recommendation.html

- Displays the result from the RecommendationController.
- Shows:
  - Area size entered by the user
  - Suggested plant(s)
  - Image of the recommended plant

It uses Thymeleaf tags like th:text and th:src to show dynamic content from Java.

How It All Connects

1. User lands on / → sees homepage.
2. They click on "Garden Form" → go to /garden.
3. User types in an area and clicks submit → form goes to /recommendation.
4. Java logic calculates a plant based on area.
5. HTML template shows recommendation using data from Java.

Student Tips

- You can change or improve the logic in RecommendationController.java.
- You can replace the images or case studies with your own.
- Customize the About page to reflect your team.

Let this be your playground to learn how back-end Java and front-end HTML work together in web development!