Answer the following

1. **Explain the @Controller and @RequestMapping annotations with examples.**

Ans –

@Controller and @RequestMapping Annotations

@Controller Annotation:

- **Purpose**: Marks a class as a Spring MVC controller. It is a specialization of the @Component annotation, making it eligible for component scanning and dependency injection.
- **Functionality**: Handles incoming web requests and returns a view (e.g., a JSP or Thymeleaf template) or data.
- Works in conjunction with @RequestMapping to map specific URLs to controller methods.

Example:

```
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;


@Controller
public class HomeController {


   @RequestMapping("/")
   public String homePage() {
      // Returns the name of the view (e.g., home.html in templates directory)
      return "home";
   }
}
```

@RequestMapping Annotation:

- **Purpose**: Maps HTTP requests to handler methods of @Controller or @RestController.
- Can be applied at the class level and/or method level to create mappings for a particular URL.
- Supports HTTP methods (GET, POST, PUT, DELETE, etc.) and media types (produces, consumes).

Attributes:

1. **value**: Specifies the URL pattern for the request.
2. **method**: Restricts the handler to specific HTTP methods.
3. **produces**: Specifies the type of response content (e.g., application/json).
4. **consumes**: Specifies the type of accepted request content (e.g., application/json).

Example:

```
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestMapping;


@Controller
@RequestMapping("/api")
public class ApiController {


   // Handles GET requests to /api/hello
   @GetMapping("/hello")
   public String sayHello() {
      return "hello"; // Returns the hello.html view
   }
```

```
    // Handles POST requests to /api/submit
    @PostMapping("/submit")
    public String submitData() {
        return "success"; // Returns the success.html view
    }
}
```

2. Explain the use of the Spring DevTools dependency.

Ans –

Purpose:

The **Spring DevTools** dependency is used to enhance developer productivity by providing features like automatic restart, live reload, and property defaults during application development.

Features:

1. **Automatic Restart**:
    - Automatically restarts the application whenever a change is made to the classpath files (e.g., .java or .class files).
    - This eliminates the need for manually restarting the server during development.
2. **Live Reload**:
    - Integrates with **LiveReload** (via browser extensions) to automatically refresh the browser when resources like HTML or CSS files are modified.
3. **Property Defaults**:
    - Disables caching in development mode, so changes to Thymeleaf templates, for example, are immediately visible.
    - Activates development-friendly properties, like enabling detailed error pages.
4. **Remote Debugging**:

- Provides tools for remotely monitoring and debugging applications.

Adding the Dependency:

In your pom.xml (for Maven):

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-devtools</artifactId>
    <scope>runtime</scope>
    <optional>true</optional>
</dependency>
```

Works:

- Spring Boot monitors the application's classpath for changes.
- When changes are detected, it restarts the application context while preserving cached objects like in-memory databases.