Spring Professional Exam Tutorial v5.0

Question 05

Question 05 - How is an incoming request mapped to a controller and mapped to a method?

Incoming request is mapped to a controller and a method by `DispatcherServlet`, which uses `HandlerMapping` and `HandlerAdapter` components for this purpose.

`HandlerMapping` components are used during Spring initialization to scan classpath for `@Controller` or `@RestController` classes with one of request mapping annotations that are part of annotation based programming model:

- `@RequestMapping`
- `@GetMapping`
- `@PostMapping`
- `@PutMapping`
- `@PatchMapping`
- `@DeleteMapping`

`HandlerAdapter` components are responsible for execution of method identified as handler candidate for the request.

Question 05 - How is an incoming request mapped to a controller and mapped to a method?

When request is performed against the server following steps are executed:

1. Application Server (Standalone or Embedded) searches for Servlet that can handle request, `DispatcherServlet` is selected based on Servlet Registration and `url-pattern`.
2. `DispatcherServlet` uses `HandlerMapping` classes to get request mapping information and `HandlerAdapter`.
3. `DispatcherServlet` uses `HandlerAdapter` to execute controller method that will handle request.
4. `DispatcherServlet` interprets results of method execution and renders `View` with help of `ViewResolver` classes.

Question 05 - How is an incoming request mapped to a controller and mapped to a method?

`@RequestMapping` allows you to specify conditions that request has to match for a method to be used as request handler. `@RequestMapping` can be used at class or method level, when used at the class level, all method level mappings inherit this primary mapping, narrowing it to a specific handler method.

For example, below controllers are supposed to map `GET /say/hello requests`, even though request mapping is defined differently, all are equal.

```
@Controller
@RequestMapping("/say/hello")
public class HelloController {

    @RequestMapping(method = GET)
    public ResponseEntity<String> sayHello() {
        return ResponseEntity.ok()
                .body("Hello");
    }
}
```

```
@Controller
@RequestMapping(path = "/say", method = GET)
public class HelloController {

    @RequestMapping("/hello")
    public ResponseEntity<String> sayHello() {
        return ResponseEntity.ok()
                .body("Hello");
    }
}
```

```
@Controller
public class HelloController {

    @RequestMapping(path = "/say/hello", method = GET)
    public ResponseEntity<String> sayHello() {
        return ResponseEntity.ok()
                .body("Hello");
    }
}
```

Question 05 - How is an incoming request mapped to a controller and mapped to a method?

`@RequestMapping` annotation allows you to specify following criteria for request:

- `path` - uri path/paths for request, for example `/api/books`
- `method` - supported HTTP method/methods: `GET, POST, HEAD, OPTIONS, PUT, PATCH, DELETE, TRACE`
- `params` - required parameters of request, for example `key1=value1, key2!=value2, key1, !key1`
- `headers` - header needs to match specified condition, for example `header1=value1, header2!=value2, header1, !header1, content-type=text/*`
- `consumes` - media types that can be consumed by request, for example `application/json`
- `produces` - media types that are produced by method handling the request, for example `application/pdf`

Question 05 - How is an incoming request mapped to a controller and mapped to a method?

Spring MVC also supports composed annotations for request mapping:

- `@GetMapping`
- `@PostMapping`
- `@PutMapping`
- `@PatchMapping`
- `@DeleteMapping`

Each of those annotations allows you to specify same conditions as `@RequestMapping` except for HTTP `method` field, following fields in `@*Mapping` are aliases to `@RequestMapping`: `path, params, headers, consumes, produces.`

In most of the cases it is possible to translate mappings between those annotations, one example when this is not possible is when creating `HTTP HEAD` request mapping.

```java
@Controller
public class HelloController {

    @RequestMapping(path = "/say/hello", method = GET)
    public ResponseEntity<String> sayHello() {
        return ResponseEntity.ok()
                .body("Hello");
    }
}
```

```java
@Controller
public class HelloController {

    @GetMapping(path = "/say/hello")
    public ResponseEntity<String> sayHello() {
        return ResponseEntity.ok()
                .body("Hello");
    }
}
```