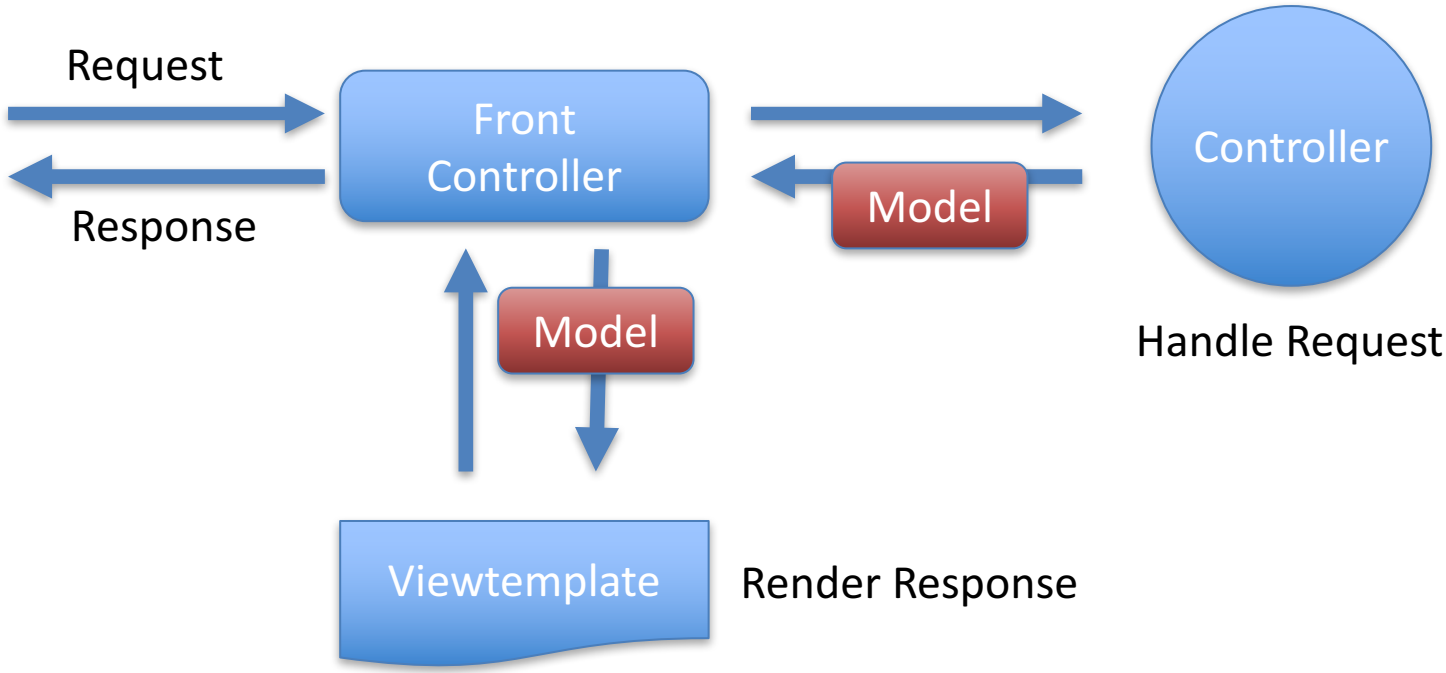
Spring MVC

What is Spring MVC?

What is Spring MVC?

- Request based web framework
- Implements MVC pattern
- FrontController (DispatcherServlet) delegates requests



Spring MVC Setup

Add Dependencies

- Adding Spring MVC Starter to your project and getting dependencies for Spring MVC, Jackson, Hibernate Validator and Tomcat.

- Thymeleaf Stater provides a templateing engine to render HTML pages.

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-thymeleaf</artifactId>
</dependency>
```

Setup

- Starters configure infrastructre Spring beans automatically
  - Thymeleaf
    - Templateing engine
    - Caching enabled -> Spring Boot Devtools
  - Spring MVC
    - ContextLoaderListener
    - DispatcherServlet
    - @EnableWebMvc with formatters, converters and validators
    - Static resources served from /static, /public, /resources or /META-INF/resources
    - Templates served from /templates
    - Provides default /error mapping
    - Default MessageSource for I18N

Controllers

Simple Controller with View

```
@Controller
public class SimpleController {

    @Value("${spring.application.name}")
    private String appName;

    @GetMapping("/")
    public String homePage(Model model) {
        model.addAttribute("appName", appName);
        return"home";
    }
}
```

Thymeleaf Template

- HTML added to /templates
- Static resources can be served from /static, /public, /resources or /META-INF/resources

```
<!DOCTYPEhtml>
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:th="http://www.thymeleaf.org">
<head>
    <title>Home Page</title>
</head>
<body>
    <h1>Hello !</h1>
    <p>
        Welcome to <span th:text="${appName}">Our App</span>
    </p>
</body>
</html>
```

Request Mappings

@RequestMapping

- Generic
  - @RequestMapping
- HTTP default methods
  - @GetMapping
  - @PostMapping
  - @PutMapping
  - @DeleteMapping
  - @PatchMapping

Controller Methods

```
@RequestMapping(path="/employees", method=RequestMethod.GET)
public String show(HttpServletRequest request, Model model) { … }
```

```
@GetMapping("/employees/{id}/address/{addressId}")
public String show(@PathVariable("id") Long id,
    @PathVariable int addressId, Locale locale, Model model,
    @RequestHeader("user-agent") String agent) { … }
```

```
@GetMapping("/employees")
public String show(@RequestParam Long id,
    @RequestParam("addressId") int addressId, Principal user,
    Map<String, Object> model, HttpSession session,
    @CookieValue("acceptedDate") String acceptedDate) { … }
```

Status Codes & Media Types

HTTP Status Codes

- Status codes are an indicator the result of the server's attempt to satisfy the request

- Divided in categories
  - 1XX: Informational
  - 2XX: Success
  - 3XX: Redirection
  - 4XX: Client Error
  - 5XX: Server Error

Media Types

- Accept & Content-Type HTTP Headers
- Client and server describes the content

REST Controller

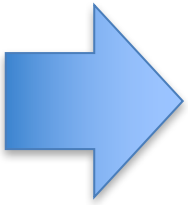
@ResponseBody

- Converter for response used because of @ResponseBody
- Converter handles rendering, no view involved
- Uses Accept-Header for Content Negotiation

```
@Controller
public class EmployeeController {

    @GetMapping(path="/employees/{id}")
    public @ResponseBody Employee showEmployee(
        @PathVariable("id") long id) { … }
}
```

```
GET /employees/42
Host: www.jobs.com
Accept: application/json
```



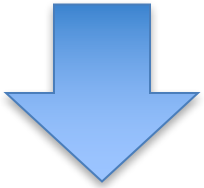
```
HTTP/1.1 200 OK
Date: ...
Content-Length: 723
Content-Type:
application/json

{
 "employee": {
    "id": 123,
    "address": [ … ], … }
}
```

@RestController Simplification

```
@Controller
public class EmployeeController {

    @GetMapping(path="/employees/{id}")
    public @ResponseBody Employee showEmployee(
        @PathVariable("id") long id) { … }
}
```



```
@RestController
public class EmployeeController {

    @GetMapping(path="/employees/{id}")
    public Employee showEmployee(
        @PathVariable("id") long id) { … }
}
```

@ResponseStatus

- Controller returns by default status 200 OK
- @ResponseStatus to override status code
- @ResponseStatus also on void methods

```
@RestController
public class EmployeeController {

    @PutMapping(path="/employees/{id}")
    @ResponseStatus(HttpStatus.NO_CONTENT)
    public void updateEmployee(
        @PathVariable("id") long id, Employee employee) {
        // Update employee data
    }
}
```

@ErrorHandler & @ControllerAdvice

- @ErrorHandler method on controller handles Exception
- Supports @ResponseStatus and custom error message
- @ControllerAdvice makes them available for all controllers

```
@Slf4j
@ControllerAdvice
public class RestEmployeeControllerAdvice {

    @ExceptionHandler(EmployeeNotFoundException.class)
    @ResponseStatus(HttpStatus.NOT_FOUND)
    public void notFound(EmployeeNotFoundException e){
        log.error("Error occured: {}", e);
    }
}
```

RestTemplate

RestTemplate

```
RestTemplate restTemplate = new RestTemplate();
```

HTTP	RESTTEMPLATE
DELETE	`delete(String, String...)`
GET	`getForObject(String, Class, String...)`
HEAD	`headForHeaders(String, String...)`
OPTIONS	`optionsForAllow(String, String...)`
POST	`postForLocation(String, Object, String...)`
PUT	`put(String, Object, String...)`

Important notes

Important notes

- Spring MVC is very powerful and the base technology for frameworks like Spring Webflow and Spring Data REST
- API Design Matters
  - URIs represent resources, not actions
  - HTTP verbs are general, but can be used in ways that make anything possible
- Implementation isn't rocket science
  - Pure REST with Spring MVC
  - Spring HATEOAS
- Easy testing
  - Out-of-container, but full Spring stack -> Spring MockMVC