

# Bài 2

# Spring MVC Controller

Module: BOOTCAMP WEB-BACKEND DEVELOPMENT

# Kiểm tra bài trước

Hỏi và trao đổi về các khó khăn gặp phải trong bài "Introduction to Spring MVC"

Tóm tắt lại các phần đã học từ bài "Introduction to Spring MVC"

- Trình bày được ý nghĩa của Controller
- Trình bày được ý nghĩa của ModelAndView Interface
- Trình bày được ý nghĩa của ModelMap Interface
- Trình bày được ý nghĩa của ViewResolver Interface
- Định nghĩa được URI với các phương thức khác nhau như GET, POST, PUT, PATH, DELETE

# Thảo luận

Annotated Controller  
RequestMapping

# Annotated Controller

- Spring MVC cung cấp các annotation như @Controller và @RestController để khai báo các controller
- Các annotation này thực hiện các nhiệm vụ như ánh xạ tới URL, khai báo tham số của request, xử lý ngoại lệ...
- Ví dụ khai báo controller:

@Controller

```
public class HelloController {  
    @GetMapping("/hello")  
    public String handle(Model model) {  
        model.addAttribute("message", "Hello  
        World!"); return "index";  
    }  
}
```

# Tự động phát hiện controller (1/2)

- Để có thể tự động phát hiện được controller, cần khai báo thuộc tính component-scan cho ứng dụng
- Ví dụ sử dụng lớp cấu hình Java:

```
@Configuration  
@ComponentScan("chinh.nguyen.controller")
```

```
public class WebConfig {  
    // ...  
}
```

# Tự động phát hiện controller (2/2)

- Ví dụ sử dụng file cấu hình XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:p="http://www.springframework.org/schema/p"
       xmlns:context="http://www.springframework.org/schema/context"
       xsi:schemaLocation="
           http://www.springframework.org/schema/beans
           http://www.springframework.org/schema/beans/spring-beans.xsd
           http://www.springframework.org/schema/context
           http://www.springframework.org/schema/context/spring-context.xsd">
  <context:component-scan base-package="com.codegym.web"/>
  <!-- ... -->
</beans>
```

- Annotation @RequestMapping được sử dụng để ánh xạ các request tới các action tương ứng của controller
- @RequestMapping bao gồm một số thuộc tính để ánh xạ đến:
  - URL
  - HTTP Method
  - Các tham số
  - Các headers
  - Media types
- @RequestMapping có thể áp dụng cho lớp hoặc phương thức



# Các biến thể của @RequestMapping

---

- @RequestMapping còn có một số biến thể dành cho các HTTP Method cụ thể:
  - @GetMapping
  - @PostMapping
  - @PutMapping
  - @DeleteMapping
  - @PatchMapping

# RequestMapping: Ví dụ

```
@RestController
@RequestMapping("/persons")
class PersonController {

    @GetMapping("/{id}")
    public Person getPerson(@PathVariable Long id) {
        // ...
    }

    @PostMapping
    @ResponseStatus(HttpStatus.CREATED)
    public void add(@RequestBody Person person) {
        // ...
    }
}
```

# Demo

Request Mapping

# URI patterns

- Có thể ánh xạ tới các request bằng cách sử dụng các mẫu đại diện sau:

?	Trùng khớp một ký tự
*	Trùng khớp với 0 hoặc nhiều ký tự
**	Trùng khớp với 0 hoặc nhiều phần của đường dẫn (path segment)

- Sử dụng annotation `@PathVariable` để khai báo các biến của đường dẫn
- Ví dụ, khai báo ở phương thức:

```
@GetMapping("/owners/{ownerId}/pets/{petId}")
public Pet findPet(@PathVariable Long ownerId, @PathVariable Long petId) {
    // ...
}
```

- Hoặc, khai báo ở lớp:

```
@Controller
@RequestMapping("/owners/{ownerId}")
public class OwnerController {
    @GetMapping("/pets/{petId}")
    public Pet findPet(@PathVariable Long ownerId, @PathVariable Long petId) {
        // ...
    }
}
```

# @PathVariable sử dụng Regex

- Có thể sử dụng biểu thức {varName:regex} để khai báo các biến đường dẫn sử dụng Regex
- Ví dụ, với đường dẫn "/spring-web-3.0.5.jar", có thể tách thành các biến name, version và file extension như sau:

```
@GetMapping("/{name:[a-z-]+}-{  
version:\\d\\.\\d\\.\\d}{ext:\\.[a-z]+}")  
public void handle(@PathVariable String version, @PathVariable String ext)  
{  
    // ...  
}
```

# Ánh xạ với Content-Type của Request

- Sử dụng thuộc tính consumes để ánh xạ đến Content-Type của request
- Ví dụ:

```
@PostMapping(path = "/pets", consumes = "application/json")
```

```
public void addPet(@RequestBody Pet pet) {  
    // ...  
}
```

- Hoặc phủ định:

```
@PostMapping(path = "/pets", consumes = "!text/plain")
```

```
public void addPet(@RequestBody Pet pet) {  
    // ...  
}
```

# Ánh xạ với Accept của Request

- Sử dụng thuộc tính `produces` để ánh xạ đến `Accept` của Request

- Ví dụ:

```
@GetMapping(path = "/pets/{petId}", produces = "application/json;charset=UTF-8")
@ResponseBody
public Pet getPet(@PathVariable String petId) {
    // ...
}
```



# Ánh xạ tới tham số của request

- Sử dụng thuộc tính `params` để ánh xạ tới tham số của đường dẫn
- Ví dụ:

```
@GetMapping(path = "/pets/{petId}", params = "myParam=myValue")  
public void findPet(@PathVariableString petId) {  
    // ...  
}
```

# Ánh xạ tới header của Request

- Sử dụng thuộc tính headers để ánh xạ đến header của request
- Ví dụ:

```
@GetMapping(path = "/pets", headers = "myHeader=myValue")  
public void findPet(@PathVariable String petId)  
{  
    // ...  
}
```

# Thảo luận

Handler Method

# Tham số của handler method

---

- Handler method có thể có nhiều tham số, chẳng hạn như:
  - HttpServletRequest
  - HttpServletResponse
  - HttpSession
  - @PathVariable
  - @RequestParam
  - @RequestHeader
  - @RequestBody
  - @ModelAttribute
  - @ModelAttribute
  - @ModelAttribute
  - ... và các tham số khác

# Giá trị trả về của handler method

- Handler method có thể trả về một trong các giá trị thuộc các loại sau:
  - @ResponseBody
  - String
  - View
  - @ModelAttribute
  - ModelAndView
  - void
  - StreamingResponseBody
  - org.springframework.ui.Model
  - ... và các loại giá trị trả về sau

# @RequestParam

- Sử dụng @RequestParam để truy cập một tham số của URI với một tham số của handler method
- Ví dụ:

```
@Controller
@RequestMapping("/pets")
public class EditPetForm {
    // ...
    @GetMapping
    public String setupForm(@RequestParam("petId") int petId, Model model) {
        Pet pet = this.clinic.loadPet(petId); model.addAttribute("pet", pet);
        return "petForm";
    }
    // ...
}
```

# @ModelAttribute

- Sử dụng @ModelAttribute để truy cập đến thuộc tính của một model (hoặc khởi tạo model nếu chưa có)
- Các thuộc tính của model được liên kết với các trường dữ liệu có cùng tên
- Ví dụ:

```
@PostMapping("/owners/{ownerId}/pets/{petId}/edit")  
public String processSubmit(@ModelAttribute Pet pet) { }
```

- Hoặc:

```
@PutMapping("/accounts/{account}")  
public String save(@ModelAttribute("account") Account account) {  
    // ...  
}
```

- Sử dụng MultipartFile để truy cập đến file được upload lên
- Ví dụ:

@Controller

```
public class FileUploadController {  
    @PostMapping("/form")  
    public String handleFormUpload(@RequestParam("name") String name,  
                                    @RequestParam("file") MultipartFile file) {  
        if (!file.isEmpty()) {  
            byte[] bytes = file.getBytes();  
            // store the bytes somewhere  
            return "redirect:uploadSuccess";  
        }  
        return "redirect:uploadFailure";  
    }  
}
```



# Demo

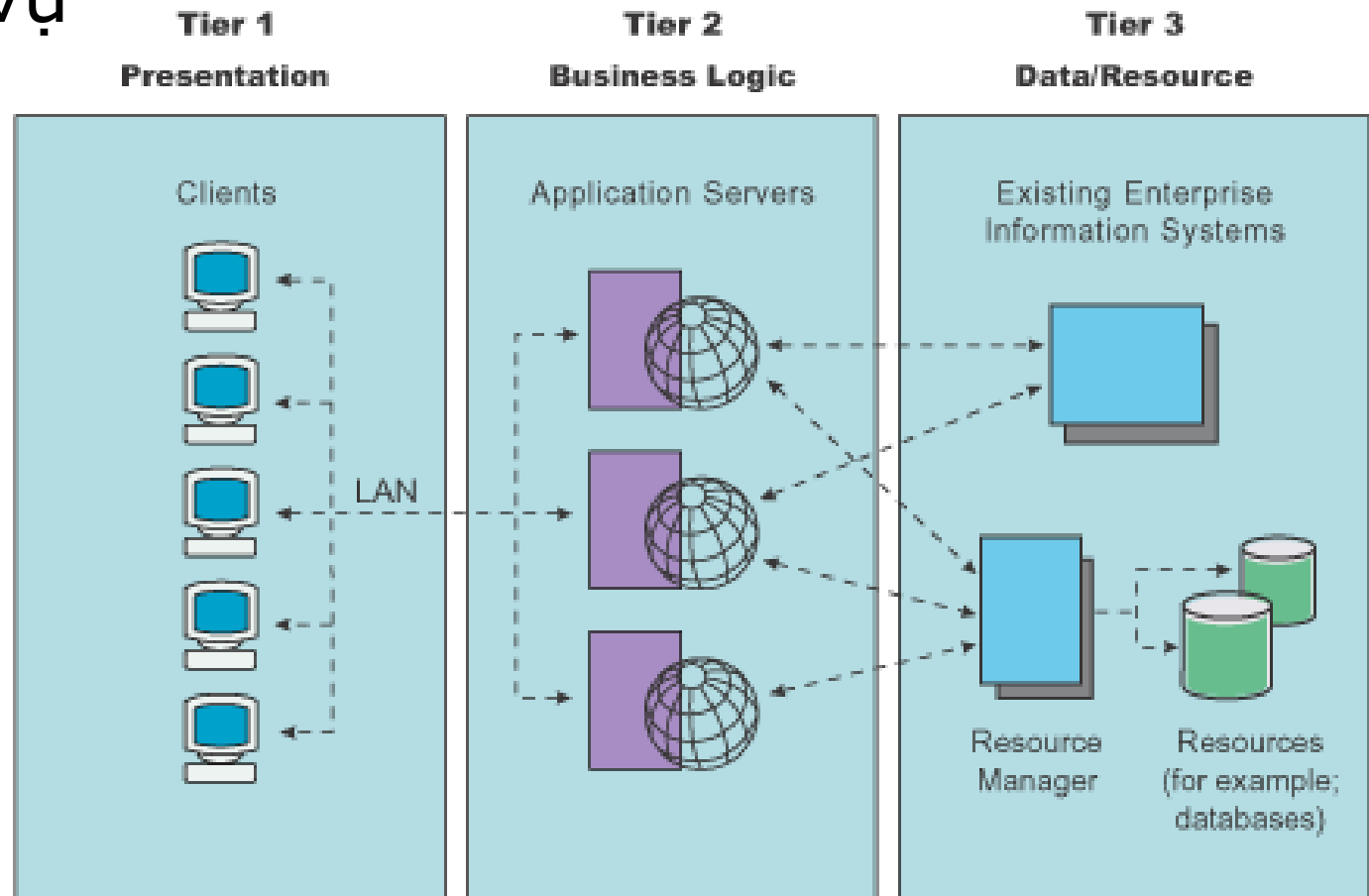
Handler Method

# Thảo luận

Kiến trúc của ứng dụng  
Spring MVC

# Kiến trúc 3-Tier và N-

- Chia tách các nhiệm vụ
- Tái sử dụng các tầng
- Dễ bảo trì



# Các tầng của Spring MVC

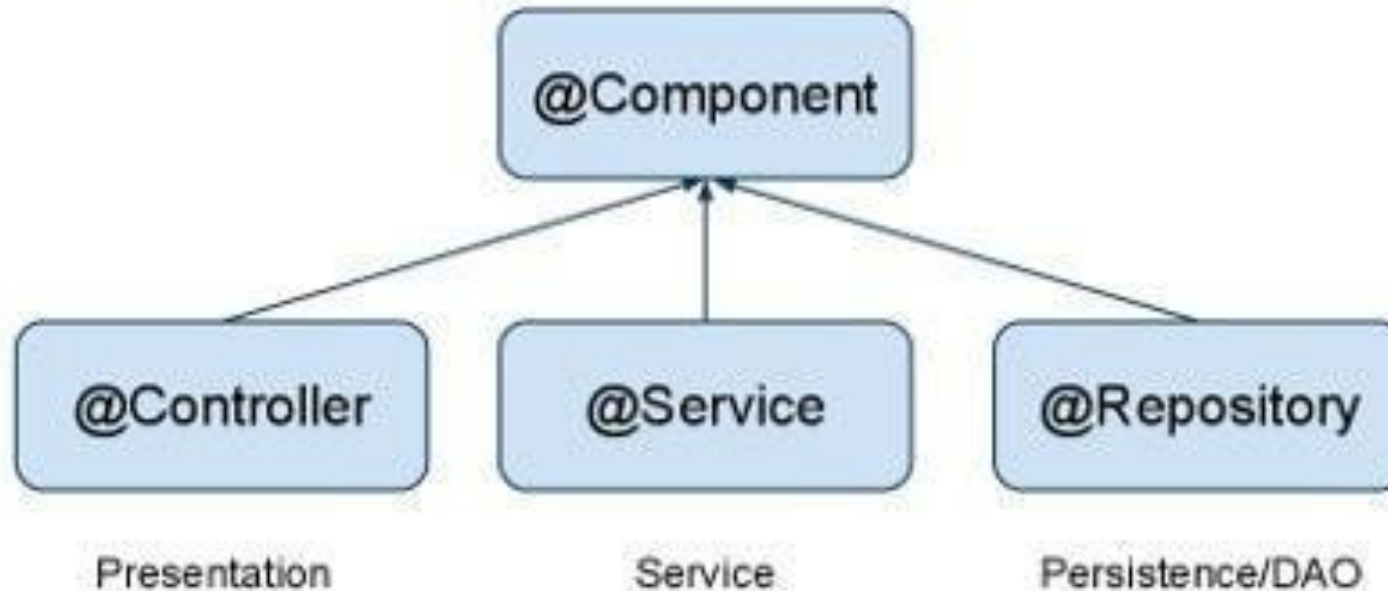
---

View

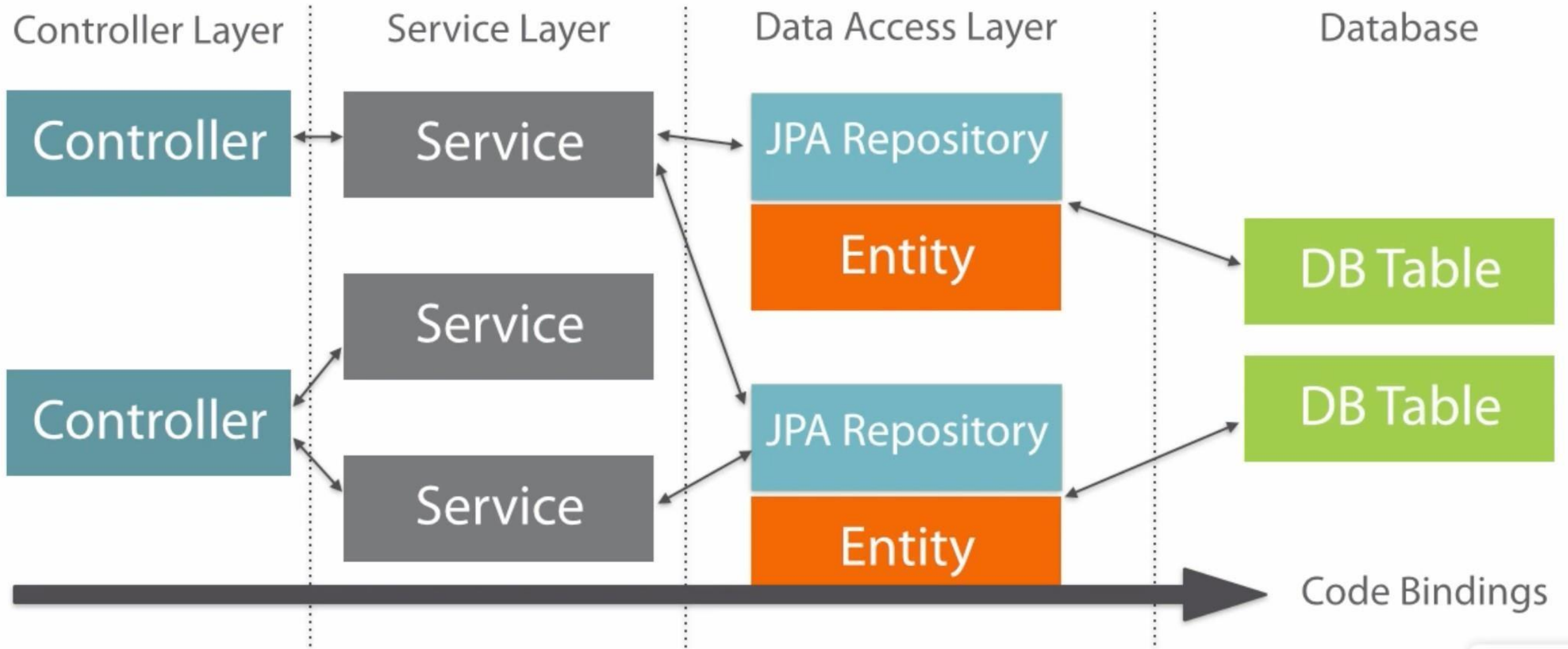
Controller

Data Model/Database

# Các component của ứng dụng Spring MVC

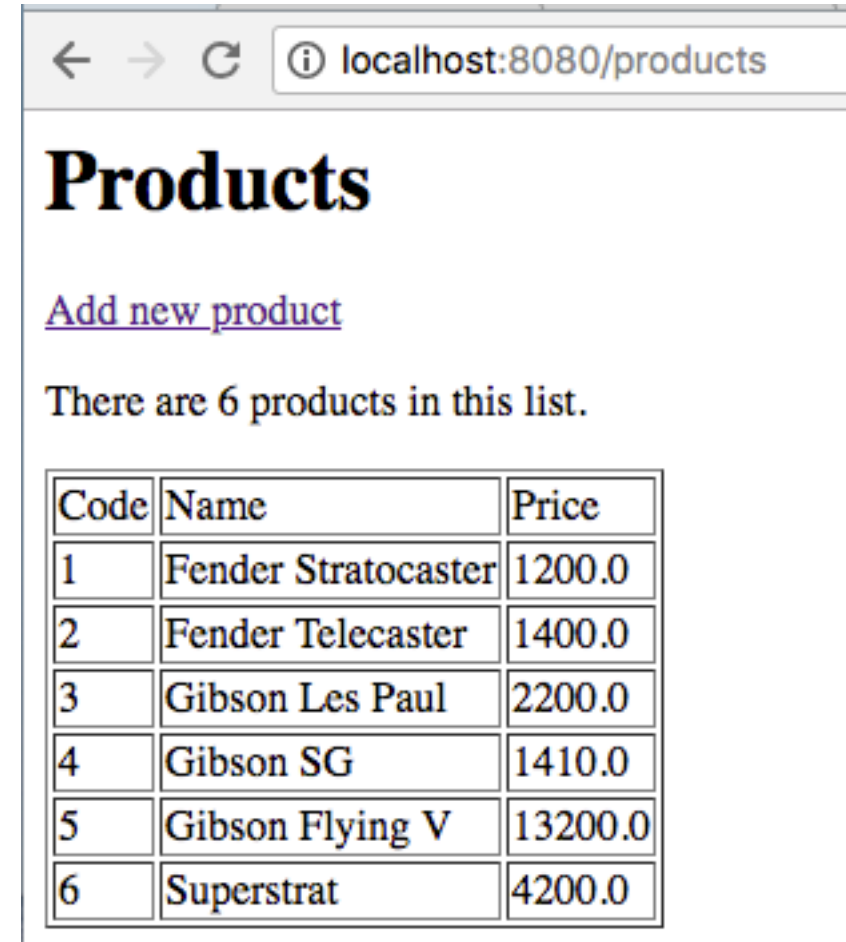


# Repository Architecture



# DEMO: Repository Architecture

- Model:
  - Class Product
- Repository:
  - ProductRepository
  - ProductRepositoryImpl
- Service:
  - ProductService
  - ProductServiceImpl
- Controller:
  - ProductController



← → ↻ ⓘ localhost:8080/products

## Products

[Add new product](#)

There are 6 products in this list.

Code	Name	Price
1	Fender Stratocaster	1200.0
2	Fender Telecaster	1400.0
3	Gibson Les Paul	2200.0
4	Gibson SG	1410.0
5	Gibson Flying V	13200.0
6	Superstrat	4200.0

- Controller là đối tượng nhận và xử lý các request
- Sử dụng annotation @Controller để khai báo các controller
- Sử dụng @RequestMapping để ánh xạ các URL tới các handler method của controller
- @RequestMapping có các biến thể khác như @GetMapping, @PostMapping, @PutMapping...
- 3 component quan trọng của Spring MVC là @Controller, @Service và @Repository
- Kiến trúc Repository là một kiến trúc tốt để xây dựng các ứng dụng web Spring MVC



# Hướng dẫn

Hướng dẫn làm bài thực hành và bài tập

Chuẩn bị bài tiếp theo: Views and Thymeleaf