FPT Education

FPT POLYTECHNIC

cdio™

Conceive Design Implement Operate

# SPRING BOOT CONTROLLER

## SLIDE 2.1

www.poly.edu.vn

# AGENDA

- PHẦN I:
  - ÁNH XẠ REQUEST VỚI METHOD
    - @REQUESTMAPPING
    - @GETMAPPING
    - @POSTMAPPING
  - ĐỌC DỮ LIỆU NGƯỜI DÙNG
    - @REQUESTPARAM
    - @REQUESTPART
    - @PATHVARIABLE
    - @COOKIEVALUE
    - OPTIONAL<T>
- PHẦN II:
  - CHIA SẺ DỮ LIỆU VỚI VIEW
    - MODEL
    - @MODELATTRIBUTE
  - ĐIỀU HƯỚNG
    - RETURN “VIEW-PATH”
    - RETURN “FORWARD:<URL>”
    - RETURN “REDIRECT:<URL>”

# Request Mapping

# @RequestMapping(value)

## Đầy đủ

```
@RequestMapping(value = {"url"})
public String method() {
        return "view-path";
}
@RequestMapping(value = {"url1", "url2"})
public String method() {
        return "view-path";
}
```

## Thu gọn

```
@RequestMapping("url")
public String method() {
        return "view-path";
}
@RequestMapping({"url1", "url2"})
public String method() {
        return "view-path";
}
```

# Mapping Summary

- @RequestMapping(url)
  - Không phân biệt method tức thực hiện request với method là POST hoặc GET
- @GetMapping(url)
  - Chỉ thực hiện khi method là GET
- @PostMapping
  - Chỉ thực hiện khi method là POST

# PHÂN BIỆT POST | GET

- Chỉ rõ đối số method của @RequestMapping để xác định request với method cụ thể
  - GET(url): @RequestMapping(value = "url", method = RequestMethod.GET)
  - POST(url): @RequestMapping(value = "url", method = RequestMethod.POST)
- Spring Boot MVC cung cấp @GetMapping và @PostMapping để tối giản dòng lệnh ánh xạ
  - GET(url): @**GetMapping**("url")
  - POST(url): @**PostMapping**("url")
- Chú ý:
  - Request có method POST khi bạn submit form với method là POST
    - <form action="*url*" method="**POST**">
  - Tất cả các request còn lại (link, submit form, nhập vào ô địa chỉ trình duyệt) đều có method là GET

# TÁCH URL THÀNH 2 PHẦN

- Spring Boot cho phép bạn tách URL thành 2 phần
  - Phần chung: Map mức class
  - Phần riêng: Map mức method

```
@Controller
@RequestMapping("account")
public class AccountController {
        @RequestMapping("sign-in") // URL:/account/sign-in
        public String signIn() {
                return "/account/sign-in.html";
        }
        @RequestMapping("sign-up") // URL:/account/sign-up
        public String signUp() {
                return "/account/sign-up.html";
        }
}
```

# @PostMapping, @GetMapping

```
@Controller
@RequestMapping("account")
public class AccountController {
        @GetMapping("login")
        public String login() {
                return "login";
        }
        @PostMapping("login")
        public String login(MyBean bean) {
                return "login";
        }
}
```

GET: /account/login

POST: /account/login

- Xử lý thông thường
  - GET: hiển thị form
  - POST: xử lý form

# User Data Handling

# @RequestParam

- @RequestParam được sử dụng để đọc giá trị tham số vào đối số của phương thức điều khiển.

```
@PostMapping("/account/login")
public String login(
        @RequestParam("username") String un,
        @RequestParam("password") String pw,
        @RequestParam(name="remember", defaultValue = "false") boolean rm){
        // Xử lý tham số
        return "account/login";
}
```

# Optional Parameter

- Tham số tùy chọn (có hoặc không)

```
@RequestMapping("url")
public String method(@RequestParam("page") Optional<Integer> optPage) {
        int pageNo = optPage.orElse(0);
        return "view-path.html";
}
```

- Ví dụ:

| URL | pageNo |
|---|---|
| http://localhost:8080/url | 0 |
| http://localhost:8080/url?page=0 | 0 |
| http://localhost:8080/url?page=3 | 3 |

- @RequestPart được sử dụng để đọc file upload từ client

```
@PostMapping("/upload/image")
public String upload(@RequestPart("photo_file") MultipartFile file){
        // Xử lý file upload
        return "upload/image";
}
```

- MultipartFile API

| Method | Return | Description |
|---|---|---|
| isEmpty() | boolean | Có upload file hay không |
| getOriginalFilename() | String | Tên file tải lên |
| getContentType() | String | Kiểu file tải lên |
| getSize() | long | Kích thước file tải lên |
| getBytes() | byte[] | Byte[] chứa dữ liệu file tải lên |
| transferTo(File) | | Lưu file tải lên vào vị trí mới |

# UPLOAD FILE EXAMPLE

```
@Autowired
ServletContext servletContext;

@PostMapping("/upload")
public String upload(@RequestPart("photo_file") MultipartFile photoFile) {
        if(photoFile.isEmpty()) {
                String dir = servletContext.getRealPath("/photos");
                String filename = photoFile.getOriginalFilename();
                try {
                        photoFile.transferTo(new File(dir, filename));
                } catch (IllegalStateException | IOException e) {
                        e.printStackTrace();
                }
        }
        ...
}
```

```
<form action="/upload"
      method="post" enctype="multipart/form-data">
      <input type="file" name="photo_file">
      <button>Upload</button>
</form>
```

# SỬ DỤNG JAVABEAN ĐỂ ĐỌC GIÁ TRỊ CÁC THAM SỐ

```
<form action="/save" method="post">
        <input name="fullname">
        <input name="age">
        <button>Save</button>
</form>
```

```
@Controller
public class MyController {
        @RequestMapping("/save")
        public String method(Staff bean) {
                return "view-path.html";
        }
}
```

```
@Data
public class Staff {
        String fullname;
        int age;
}
```

- *Chú ý:*
  - *Bean property phải cùng tên với các parameter*
  - *Sử dụng @Data để sinh các cặp getter và setter cho các field.*

# ĐỌC GIÁ TRỊ COOKIE

- Sử dụng @CookieValue để đọc giá trị của cookie

```
@RequestMapping("/url")
public String method(@CookieValue(name="cki", defaultValue = "poly") String val) {...}
```

- Ví dụ:
  - Đọc giá trị của cookie có tên cki vào đối số val nếu cookie cki tồn tại
  - Nếu cookie cki không tồn tại thì đối số val có giá trị là poly

# ĐỌC GIÁ TRỊ BIẾN ĐƯỜNG DẪN

- Sử dụng @PathVariable để đọc giá trị của biến đường nằm trong dẫn của URL.

```
@RequestMapping("/page/{size}/{number}")
public String method(
            @PathVariable("size") int pageSize,
            @PathVariable("number") int pageNumber) {...}
```

- Với request có url là "/page/10/5" thì giá trị của các đối số:
  - pageSize là 10
  - pageNumber là 5

# ĐỌC GIÁ TRỊ BIẾN ĐƯỜNG DẪN TÙY CHỌN

```
@RequestMapping({"/page", "/page/{size}", "/page/{size}/{number}"})
public String method(
            @PathVariable("size") Optional<Integer> optPageSize,
            @PathVariable("number") Optional<Integer> optPageNumber) {
    int pageSize = optPageSize.orElse(8);
    int pageNumber = optPageNumber.orElse(0);
    ...
}
```

- Các tình huống có thể xảy ra với các request

| URL | pageSize | pageNumber |
|---|---|---|
| /page | 8 | 0 |
| /page/10 | 10 | 0 |
| /page/10/5 | 10 | 5 |

FPT Education

FPT POLYTECHNIC

cdio™

Conceive Design Implement Operate

# SPRING BOOT CONTROLLER

## SLIDE 2.2

www.poly.edu.vn

# Data Sharing

# Model & @ModelAttribute

- Model là nơi chứa dữ liệu do Controller tạo ra để chia sẻ với View
- Trong Controller có 3 cách để bổ sung dữ liệu chia sẻ vào Model:
  - Model.addAttribute(name, value)
    - Bổ sung 1 biến vào model
  - MappingMethod(@ModelAttribute(name) Type value)
    - Bổ sung giá trị đối số vào model
  - @ModelAttribute(name) Type method(){...return value;}
    - Bổ sung kết quả trả về của một phương thức vào model

# BỔ SUNG BIẾN VÀO MODEL

```
@RequestMapping("/sharer/index")
public String index(Model model){
        model.addAttribute("message", "Hello Spring");
        model.addAttribute("Hello Spring");
        model.addAttribute("now", new Date());
        model.addAttribute(new Date());
        model.addAttribute("user", new Account());
        model.addAttribute(new Account());
        return "sharer/index";
}
```

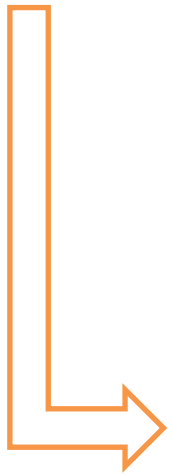
## Model

| name | value |
|---|---|
| message | “Hello Spring” |
| string | “Hello Spring” |
| now | Date object |
| date | Date object |
| user | Account object |
| account | Account Object |

- *Chú ý: Nguyên tắc đặt tên mặc định: lấy tên kiểu của biến và đổi ký tự đầu tiên sang ký tự thường*

# BỔ SUNG GIÁ TRỊ ĐỐI SỐ VÀO MODEL

```
@RequestMapping("/sharer/index")
public String index(@ModelAttribute Date now,
        @ModelAttribute("user") Account account){
    return "sharer/index";
}
```



## Model

| name | value |
|---|---|
| date | Date object |
| user | Account Object |

# BỔ SUNG KẾT QUẢ TRẢ VỀ CỦA MỘT PHƯƠNG THỨC VÀO MODEL

```
@ModelAttribute("now")
public Date getNow() {
    return new Date();
}
@ModelAttribute
public Account getUser() {
    return new Account();
}
```

## Model

| name | value |
|---|---|
| now | Date object |
| account | Account Object |

# Return of Controller Methods

# Overview of Controller Method Returns

```
@RequestMapping("url1") public String method1() {
        return "view-path";
}
@RequestMapping("url2") public void method2() {
}
@RequestMapping("url3") public String method3() {
        return "forward:/order-url";
}
@RequestMapping("url4") public String method4() {
        return "redirect:/other-url";
}
@ResponseBody
@RequestMapping("url5") public Object method5() {
        return raw-data;
}
```

Chuyển tiếp sang view (view-path)

Chuyển tiếp sang view (url2)

Chuyển tiếp sang url khác

Tạo request mới đến url khác

***Trả về dữ liệu thô (bất kỳ)***

# RETURN VIEW NAME

```
@RequestMapping("/demo/1")
public String method1(){
    return "demo/form";
}
```

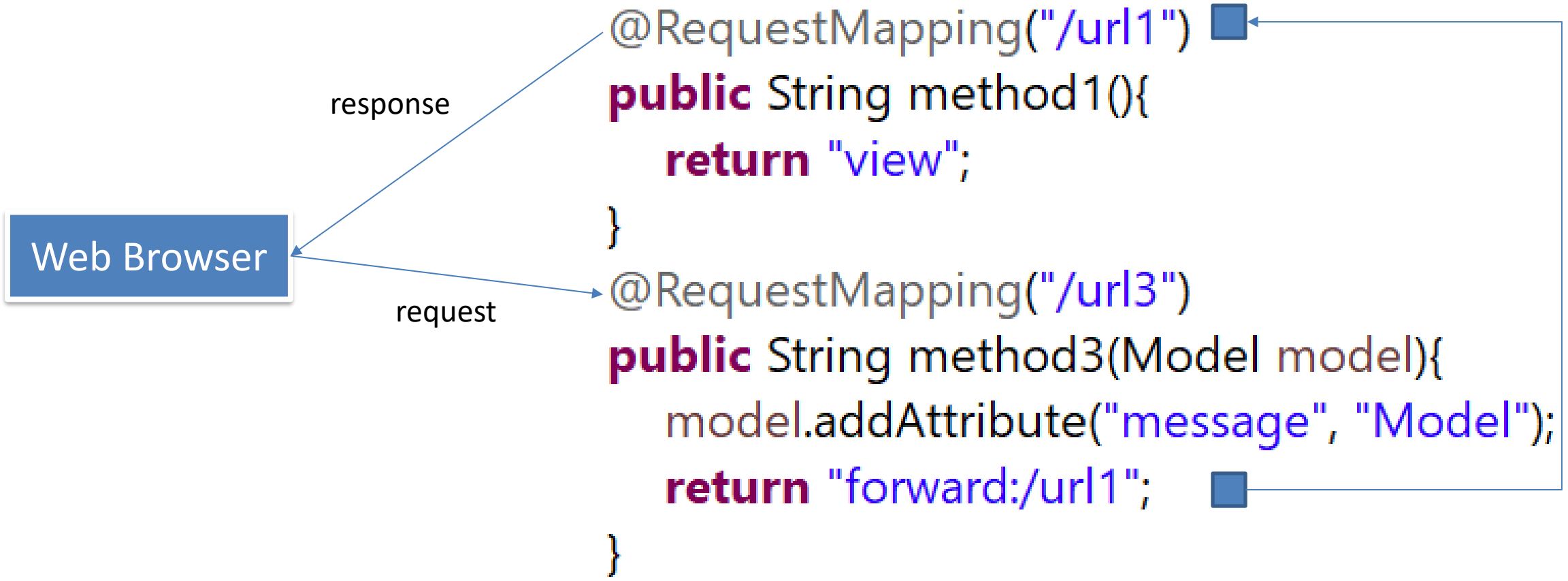
→ demo/form.jsp

```
@RequestMapping("/demo/2")
public void method2(){
}
```

→ demo/2.jsp

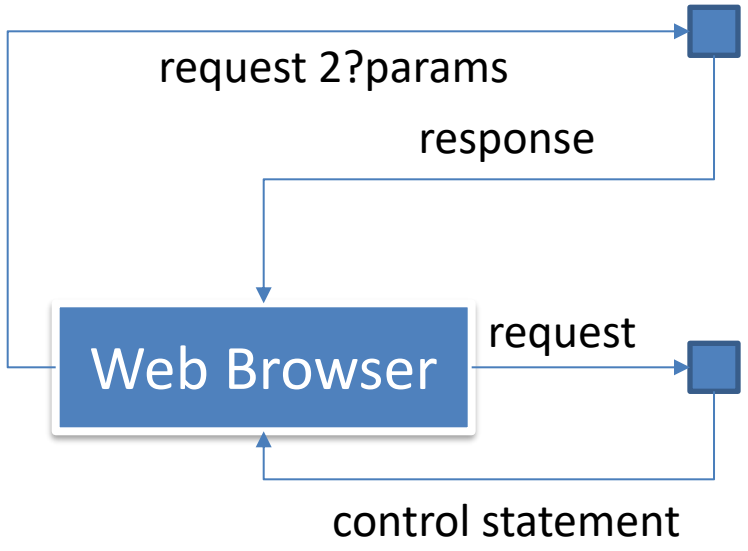
- *Chú ý:*
  - *Lệnh return được hiểu là chuyển tiếp (forward) sang view*
  - *Theo cấu hình mặc định (kiểu trả về là void) thì view-path của controller method là mapping-url*

# FORWARD




- *Forward xảy ra phía server trên cùng một request nên dữ liệu trong Model được tạo bởi url3 có thể chia sẻ với url1*

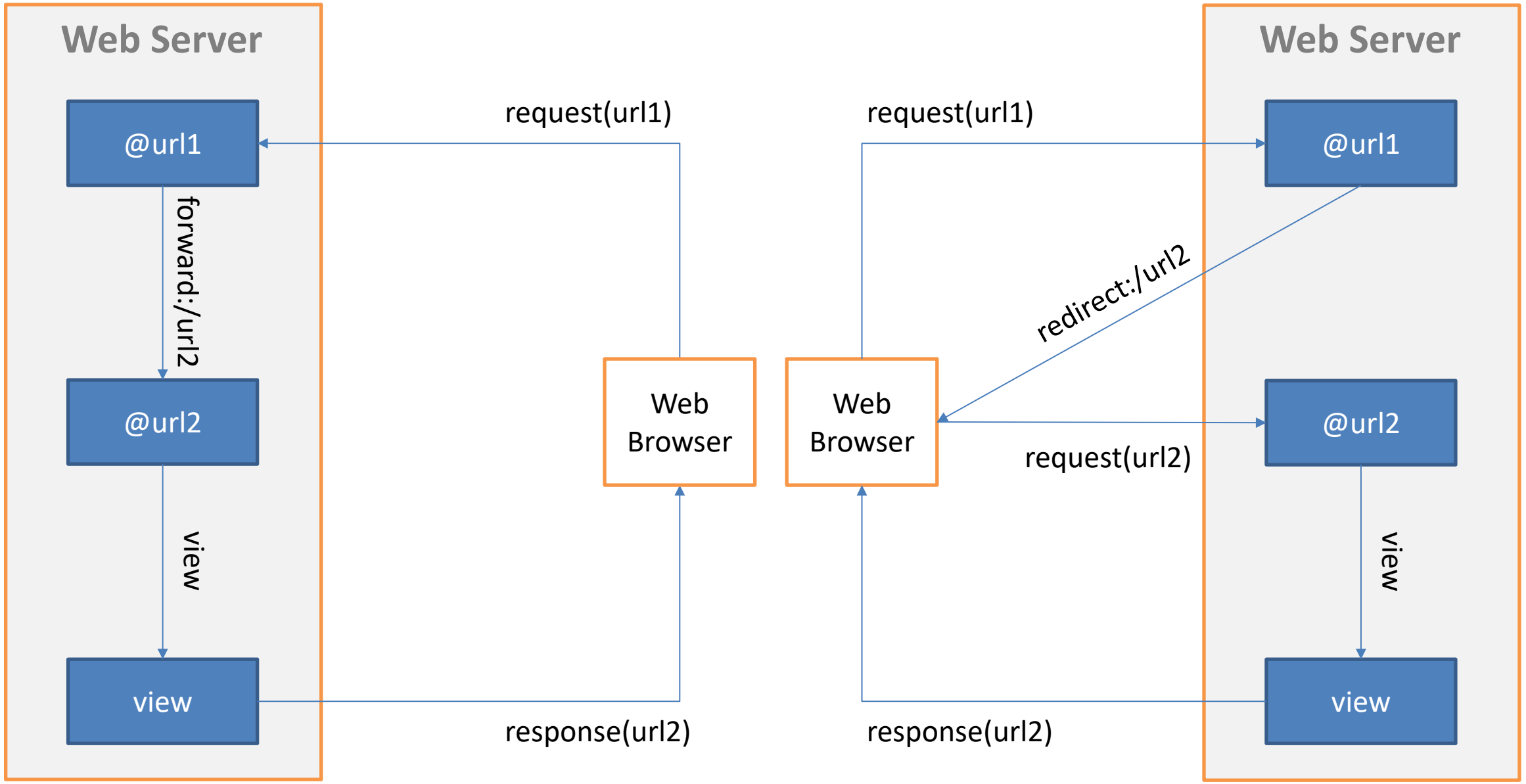
# REDIRECT




```
@RequestMapping("/url1")
public String method1(){
    return "view";
}
@RequestMapping("/url2")
public String method2(RedirectAttributes params){
    params.addAttribute("message", "RedirectAttributes");
    return "redirect:/url1";
}
```

- *@url1 và @url2 được khẩn cầu trên 2 request khác nhau nên Model không thể chia sẻ*
- *Sử dụng RedirectAttributes để chia sẽ dữ liệu text thay vì Model. RedirectAttributes sẽ được chuyển đổi thành các tham số của request mới*

# REDIRECT AND FORWARD



# @ResponseBody – Raw Data

```
@ResponseBody
@RequestMapping("/url1")
public String method1(){
    return "Chào quý vị đại biểu";
}
@ResponseBody
@RequestMapping("/url2")
public void method2(){
}
@ResponseBody
@RequestMapping("/url3")
public Account method3(){
    return new Account();
}
```

“Chào quý vị đại biểu”

empty

{JSON}

# Summary

- ☑ Phần I:
  - ☑ Ánh xạ Request với Method
    - ☑ @RequestMapping
    - ☑ @GetMapping
    - ☑ @PostMapping
  - ☑ Đọc dữ liệu người dùng
    - ☑ @RequestParam
    - ☑ @RequestPart
    - ☑ @PathVariable
    - ☑ @CookieValue
    - ☑ Optional<T>
- ☑ Phần II:
  - ☑ Chia sẻ dữ liệu với view
    - ☑ Model
    - ☑ @ModelAttribute
  - ☑ Điều hướng
    - ☑ return "view-path"
    - ☑ return "forward:<URL>"
    - ☑ return "redirect:<URL>"


FPT Education
FPT POLYTECHNIC
Thank you