FPT Education

FPT POLYTECHNIC

cdio™

Conceive Design Implement Operate

1

Spring Beans

Giảng viên:

www.poly.edu.vn

Agenda

- Basic Spring Beans
  - Understanding @Autowiared
  - Spring Beans, DI and IoC
  - Injecting and usering built-in Spring Beans
  - Building Utility Spring Beans (Upload, Cookie, Session)
- Advanced Spring Beans
  - Bean Scopes
  - Understanding deeper IoC
  - Building ShoppingCartService

Understanding Spring Beans

UNDERSTANDING @AUTOWIRED

```
@Controller
public class HomeController {
        @Autowired
        HttpServletRequest request;

        @RequestMapping("/index.php")
        public String index() {
                request.setAttribute("message", "Dependence Injection");
                return "index";
        }
}
```

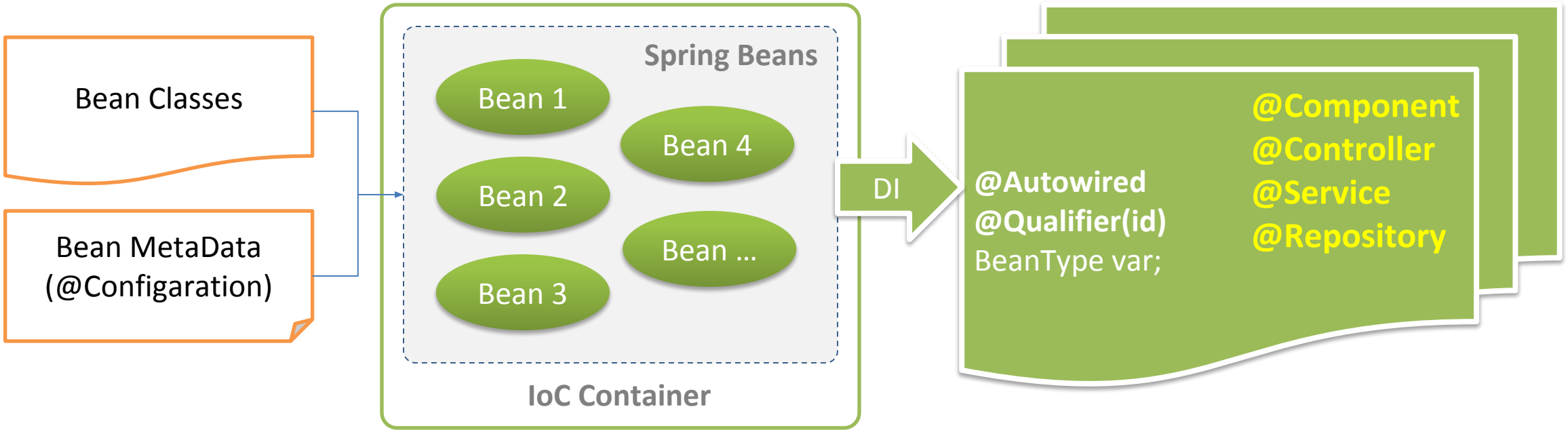
TẠI SAO BIẾN REQUEST KHÔNG NULL?

-

*Spring sẽ tìm kiếm đối tượng (bean) trong môi trường của nó **có kiểu tương ứng với biến và liến kết biến với đối tượng tìm thấy** hoặc báo lỗi nếu không tìm thấy.*

- Spring Bean là các bean được Spring quản lý. Các bean này gồm:
  - Bean hệ thống (built-in)
    - HttpServletRequest
    - HttpServletResponse
    - HttpSession
    - ServletContext
    - …
  - Bean do người sử dụng (user-defined) nạp vào.
- DI (Dependence Injection = "Tiêm các đối tượng cần thiết") là cách thức liên kết các Spring Bean vào các biến cần thiết của chương trình.
- Ví dụ:
  - @Autowired HttpSession session;
  - @Autowired ServletContext application;

Spring Beans



- Spring Beans = <Built-in beans> + <User-defined beans>
- ID (tiêm) = liên kết đến bean cần thiết
- IoC Container (Inversion of Control = "Đảo ngược điều khiển") là engine làm nhiệm vụ nạp bean từ bên ngoài vào hệ thống nhằm cho phép điều khiển theo mã tùy biến của bean bên ngoài.

User-defined Spring Beans

User-defined Spring Bean

- Có thể yêu cầu IoC Container nạp bean bằng 2 cách
  - Cách 1: Tạo lớp có đính kèm các annotation sau
    - @Controller
    - @Component
    - @Service
    - @Repository
  - Cách 2: Tạo lớp bất kỳ và viết phương thức tạo bean kèm @Bean khai báo trong file cấu hình @Configuration
- Khi khởi động ứng dụng, Spring sẽ tìm kiếm các annotation đã được định nghĩa trên các lớp và các phương thức cùng với các @Scope để tạo và nạp các đối tượng vào thời điểm phù hợp

User-defined Spring Bean – Way #1

```java
@Controller
public class DIController {
        @Autowired
        Company company;

        @ResponseBody
        @RequestMapping("test")
        public String test() {
                return company.getName();
        }
}
```

use

```java
@Getter @Setter
@NoArgsConstructor @AllArgsConstructor
@Component
public class Company {
        String name = "FPT Polytechnic";
        String logo = "poly.png";
}
```

- Spring IoC Container sẽ tạo đối tượng từ lớp Company (vì có @Component) và nạp vào hệ thống. Vì vậy DIController có thể sử dụng @Autowired để tiêm vào và sử dụng

User-defined Spring bean – Way #2

```java
@Controller
public class DIController {
        @Autowired
        Company company;

        @ResponseBody
        @RequestMapping("test")
        public String test() {
                return company.getName();
        }
}
```

use

```java
@Getter @Setter
@NoArgsConstructor @AllArgsConstructor
public class Company {
        String name = "FPT Polytechnic";
        String logo = "poly.png";
}
```

use

```java
@Configuration
public class BeanConfig {
        @Bean
        public Company getCompany() {
                Company company = new Company();
                company.setName("FPT Education");
                return company;
        }
}
```

Phương thức getCompany() được chú thích bởi @Bean bên trong lớp được chú thích bởi @Configuration vì vậy đối tượng trả về của phương thức này đượng Spring nạp vào Spring Beans.

Tìm hiểu thêm về Injection

- @Authowired
  - Dựa vào kiểu dữ liệu của biến để tìm kiếm và liên kết đến Spring Bean có kiểu phù hợp.
  - Nếu có nhiều hơn một Spring Bean có kiểu phù hợp với biến thì hệ thống sẽ báo lỗi
- Giải quyết giải quyết xung đột này theo 2 cách sau
  - Cách 1: Cấu hình @Bean với @Primary để báo cho hệ thống biết đó là bean chính
  - Cách 2: Đặt id cho bean và sử dụng @Qualifier(id) bên cạnh @Autowired

Cách 1: @Primary

- @Primary được sử dụng kèm với @Bean để báo cho IoC biết đây là bean chính. Nếu có các bean cùng kiểu thì Spring sẽ ưu tiên liên kết với bean này.

```
@Autowired
Account account;
```

Liên kết với bean này

```
@Configuration
public class HelloConfig {
    @Bean
    public Account getAccount1() {
        return new Account();
    }

    @Primary @Bean
    public Account getAccount2() {
        return new Account();
    }
}
```

Cách 2: @Qualifier(id)

- @Qualifier được sử dụng kèm với @Autowired để liên kết đến Spring Bean theo kiểu và id.

```
@Autowired
@Qualifier("bean1")
Account account;
```

Liên kết với bean này

```
@Configuration
public class HelloConfig {
    @Bean("bean1")
    public Account getAccount1() {
        return new Account();
    }

    @Bean("bean2")
    public Account getAccount2() {
        return new Account();
    }
}
```

3 CÁCH TIÊM

- Tiêm vào field

```
@Autowired
@Qualifier("bean1")
Account account;
```

- Tiêm vào setter

```
@Autowired
public void setAccount(@Qualifier("bean1") Account account){}
```

- Tiêm vào constructor

```
@Autowired
public MyController(@Qualifier("bean1") Account account){}
```

Building Utility Spring Beans

BUILDING UPLOADSERVICE

```
@Service
public class UploadService {
    @Autowired
    ServletContext app;
    public File save(MultipartFile file, String folder) {
        File dir = new File(app.getRealPath(folder));
        if(!dir.exists()) dir.mkdirs();
        try {
            File saveFile = new File(dir, file.getOriginalFilename());
            file.transferTo(saveFile);
            return saveFile;
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
    }
}
```

BUILDING COOKIESERVICE

```
@Service
public class CookieService{
    @Autowired HttpServletRequest request;
    @Autowired HttpServletResponse response;
    public Cookie create(String name, String value, int days) {
        Cookie cookie = new Cookie(name, value);
        cookie.setMaxAge(days * 60 * 60);
        cookie.setPath("/");
        return cookie;
    }
    public Cookie get(String name) {
        Cookie[] cookies = request.getCookies();
        if(cookies != null)
            for(Cookie cookie: cookies)
                if(cookie.getName().equalsIgnoreCase(name)) return cookie;
        return null;
    }
}
```

BUILDING SESSIONSERVICE

```
@Service
public class SessionService{
    @Autowired
    HttpSession session;
    public void setAttribute(String name, Object value) {
        session.setAttribute(name, value);
    }
    public <T> T getAttribute(String name) {
        return (T) session.getAttribute(name);
    }
    public void removeAttribute(String name) {
        session.removeAttribute(name);
    }
}
```

FPT Education

FPT POLYTECHNIC

cdio™

Conceive Design Implement Operate

2

Spring Beans

Giảng viên:

www.poly.edu.vn

Understanding deeper IoC

Thực chất của IoC là để làm gì?

- Đối với các ứng dụng có độ tùy biến cao, có nhiều lời gọi phương thức trong chương trình mà không biết nó thực hiện chức năng gì vì còn phụ thuộc nhiều yếu tố trong tương lai.
- Để tránh sự phụ thuộc vào mã nguồn trong class, khi tiêm người ta sử dụng Interface. Điều này có nghĩa nếu chúng ta thay lớp này bằng lớp khác thì ứng dụng sẽ thực hiện chức năng theo mong muốn của lớp mới (đây chính là mục tiêu của IoC – đảo ngược sự điều khiển)
- Xét một ví dụ đơn gian:
  - Xây dựng một phương thức sayGreeting() để xuất ra một lời chào theo ngôn ngữ địa phương nơi cài đặt phần mềm.
  - Rõ ràng là lúc xây dựng phần mềm chúng ta không biết phải viết code cho phương thức này, mãi đến khi nào chúng ta bán và cài đặt cho một khách hàng nào đó thì chúng ta mới biết được phải viết mã như thế nào.

IoC IMPLEMENTATION

```java
@Controller
public class SpeechController {
    @Autowired
    Speech speech;

    @GetMapping("/speech/hello")
    public void hello(){
        speech.sayGreeting();
    }
}
```

use

```java
public interface Speech {
    void sayGreeting();
}
```

```java
@Component
public class VNSpeech implements Speech{
    @Override
    public void sayGreeting() {
        System.out.println("Xin chào");
    }
}
```

Thay thế lớp này ứng dụng sẽ hoạt động theo ý của bạn

Spring Bean Scopes

Spring Bean Scopes

- Mặc định khi khai báo một bean bằng cách cấu hình hoặc annotation thì bean sẽ được tạo ra một đối tượng duy nhất (singleton) và được nạp vào hệ thống ngay từ đầu và phục vụ cho toàn ứng dụng.
- Spring cung cấp một số annotation cho phép khai báo bean để yêu cầu hệ thống nạp và quản lý vòng đời của chúng.
  - @ApplicationScope
    - Tạo bean ngay từ đầu, chia sẻ trên phạm vi application và giải phóng khi ứng dụng kết thúc
  - @SessionScope
    - Tạo bean ngay từ lúc bắt đầu một phiên, chia sẻ trên phạm vi session và giải phóng khi session timeout
  - @RequestScope
    - Tạo bean ngay từ lúc bắt đầu một request, chia sẻ trên phạm vi request và giải phóng ngay khi request kết thúc

Spring Bean Scopes

```
@Configuration
public class BeanConfig {
        @SessionScope
        @Bean("cart")
        public ShoppingCart getShoppingCart() {
                return new ShoppingCart();
        }
}
```

```
public class ShoppingCart {
        //...
}
```

```
@SessionScope
@Service("cart")
public class ShoppingCart {
        //...
}
```

session.getAttribute("scopedTarget.cart")
${sessionScope['scopedTarget.cart']}

BUILDING SHOPPING CART SERVICE

- Giỏ hàng điện tử là một thành phần không thể thiếu đối với các nhà phát triển ứng dụng web.
- Với tên gọi giỏ hàng (Shopping Cart) làm người học dễ nhầm tưởng chỉ khi bán hàng mới xây dựng giỏ hàng. Cần hiểu rộng hơn là một *kỹ thuật lập trình lưu dữ tạm thời* những dữ liệu được chọn. Nhờ vậy chúng ta sẽ dễ dàng ứng dụng vào một số lĩnh vực khác một cách hiệu quả
- Ví dụ
  - Chứa các mặt hàng đã chọn (web bán hàng)
  - Chứa các khóa học đã chọn (đào tạo)
  - Chứa các dịch vụ của một tour tự chọn (du lịch)
  - …

- Có nhiều cách để thực hiện việc lưu giữ tạm thời này, tùy thuộc vào sự phân tích tình huống và lựa chọn phù hợp
  - HttpSession (lưu phía server): kết thúc phiên sẽ bị giải phóng
  - sessionStorage (lưu phía client): kết thúc phiên sẽ bị giải phóng. Cần phải gửi lên server khi đặt hàng.
  - Database: duy trì cho đến khi xóa. Tuy nhiên cần tính đến giải pháp thu gom rác
  - localStorage (lưu phía client): duy trì cho đến khi xóa, chỉ xem lại trên cùng một máy. Cần phải gửi lên server khi đặt hàng.
  - Cookie (lưu phía client và tự động truyền thông với server khi có request): Duy trì theo thời hạn, kích thước lưu trữ rất hạn chế.
- Trong phạm vi môn học, chúng ta sử dụng HttpSession (server-side)

BUILDING SHOPPINGCARTSERVICE

- Nghiệp vụ của giỏ hàng:
  - Thêm vào giỏ
  - Thay đổi số lượng
  - Xóa khỏi giỏ
  - Xóa sạch giỏ
  - Tính tổng tiền
  - Tính tổng số lượng
  - Liệt kê các mặt hàng

```
public interface ShoppingCartService {
    List<CartItem> getItems();
    void add(int id);
    void remove(int id);
    void update(int id, int qty);
    void clear();
    int getCount();
    double getAmount();
}
```

SHOPPINGCARTSERVICEIMPL

```java
@SessionScope
@Service
public class ShoppingCartServiceImpl implements ShoppingCartService{
    Map<Integer, CartItem> map = new HashMap<>();
    @Override
    public Collection<CartItem> getItems() {return null;}
    @Override
    public void add(int id) {}
    @Override
    public void remove(int id) {}
    @Override
    public void update(int id, int qty) {}
    @Override
    public void clear() {}
    @Override
    public int getCount() {return 0;}
    @Override
    public double getAmount() {return 0;}
}
```

```java
public interface ShoppingCartService {
    List<CartItem> getItems();
    void add(int id);
    void remove(int id);
    void update(int id, int qty);
    void clear();
    int getCount();
    double getAmount();
}
```

Summary

- Understanding @Autowiared
- Spring Beans, DI and IoC
- Injecting and usering built-in Spring Beans
- Building Utility Spring Beans
  - UploadService
  - CookieService
  - SessionService
- Bean Scopes
- Understanding deeper IoC
- Building ShoppingCartService


FPT Education
FPT POLYTECHNIC
Thank you
TIN HỌC VĂN PHÒNG QUỐC TẾ THEO CHUẨN ICDL
Syllabus 5 - Office 2007
LUẬT KINH TẾ
KỸ THUẬT SOẠN THẢO VÀ TRÌNH BÀY VĂN BẢN