FPT Education

FPT POLYTECHNIC

cdio™

Conceive Design Implement Operate

SPRING BEANS

SLIDE 5.1

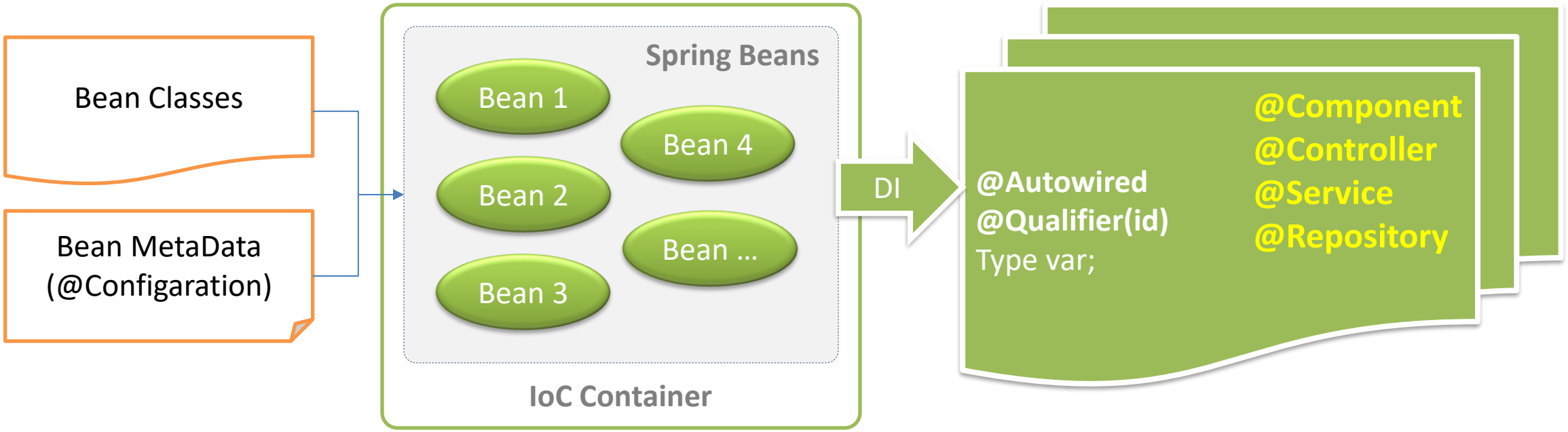
www.poly.edu.vn

AGENDA

- PHẦN I:
  - GIỚI THIỆU SPRING BEANS
  - SỬ DỤNG SPRING BEANS DỰNG SẴN
  - XÂY DỰNG VÀ CẤU HÌNH SPRING BEAN
- PHẦN II
  - TÌM HIỂU BEAN SCOPES
  - TÌM HIỂU VỀ DI AND IOC
  - XÂY DỰNG CÁC BEAN TIỆN ÍCH THƯỜNG DÙNG

Spring Bean Introduction

SPRING BEAN INTRODUCTION



- Spring Bean là các bean (đối tượng) được Spring quản lý. Các bean này gồm các bean do hệ thống tạo sẵn (built-in) hoặc do người dung định nghĩa (user-defined) và cấu hình yêu cầu hệ thống nạp vào.
- Khi cần sử dụng một bean nào đó chỉ cần sử dụng @Autowire để yêu cầu hệ thống tìm kiếm bean và gán cho biến

SỬ DỤNG BUILT-IN BEAN

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

        @RequestMapping("/my-url")
        public String method(Model model) {
                String uri = request.getParameter("fullname");
                return "/demo/page";
        }
}
```

Tại sao biến request không null?

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

User-defined Spring Beans

User-defined Spring Bean

- Có thể yêu cầu IoC Container nạp bean bằng 2 cách
  - Cách 1: Tạo bean class được chú thích bởi các annotation sau
    - @Controller
    - @Component
    - @Service
  - Cách 2: Tạo lớp bất kỳ và sử dụng @Bean khai báo trong file cấu hình @Configuration
- Khi khởi động ứng dụng, Spring sẽ tự động tìm kiếm các lớp đã thực hiện theo 2 cách trên để tạo đối tượng và nạp vào hệ thống để quản lý.

CÁCH 1: TẠO VÀ NẠP SPRING BEAN VỚI ANNOTATION

```
@Service
public class CookieService {
        public Cookie create(String name, String value, int expiry) {
                return null;
        }
        public void delete(String name) {
        }
        public String getValue(String name) {
                return null;
        }
}
```

CÁCH 2: TẠO VÀ CẤU HÌNH NẠP SPRING BEAN

```
public class CookieService {
        public Cookie create(String name, String value, int expiry) {
                return null;
        }
        public void delete(String name) {
        }
        public String getValue(String name) {
                return null;
        }
}
```

```
@Configuration
public class BeanConfig {
        @Bean
        public CookieService getCookieService() {
                return new CookieService();
        }
}
```

SỬ DỤNG SPRING BEAN

```
@Controller
public class MyController {
        @Autowired
        CookieService cookieService;

        @RequestMapping("/my-url")
        public String method(Model model) {
                cookieService.create("name", "Poly", 30*60);
                return "/demo/page";
        }
}
```

XUNG ĐỘT NHIỀU BEAN CÙNG KIỂU

- Khi nạp nhiều bean cùng kiểu hệ thống sẽ báo lỗi. Trường hợp sau là nạp 2 bean cùng kiểu CookieService.

```
@Configuration
public class BeanConfig {
        @Bean
        public CookieService getCookieService1() {
                return new CookieService();
        }
        @Bean
        public CookieService getCookieService2() {
                return new CookieService();
        }
}
```

XUNG ĐỘT NHIỀU BEAN CÙNG KIỂU

```
@Configuration
public class BeanConfig {
        @Bean("CS1")
        public CookieService getCookieService1() {
                return new CookieService();
        }
        @Bean("CS2")
        public CookieService getCookieService2() {
                return new CookieService();
        }
}
```

Giải pháp 1: Sử dụng bean id để phân biệt các bean cùng kiểu

```
@Controller
public class MyController {
        @Qualifier("CS2")
        @Autowired
        CookieService cookieService;
        ...
}
```

XUNG ĐỘT NHIỀU BEAN CÙNG KIỂU

```
@Configuration
public class BeanConfig {
        @Bean
        public CookieService getCookieService1() {
                return new CookieService();
        }
        @Primary
        @Bean
        public CookieService getCookieService2() {
                return new CookieService();
        }
}
```

Giải pháp 2: Cấu hình bean với @Primary để xác định bean chính

```
@Controller
public class MyController {
        @Autowired
        CookieService cookieService;
        ...
}
```

BEAN INJECTION

```
@Controller
public class MyController {
        @Autowired
        CookieService cookieService;

        public MyController(@Autowired CookieService cookieService) {
                cookieService.add(null, null, 0);
        }

        @RequestMapping("/my-url")
        public String method(@Autowired CookieService cookieService) {
                return "/demo/page";
        }
}
```

Inject bean to field

Inject bean to constructor

Inject bean to argument

SỬ DỤNG BEAN TRONG THYMELEAF

```
@Service
public class CartService{
    List<CartItem> getItems(){
        return null;
    }
    …
}
```

Truy xuất bean bằng tên class

```
${@cartService.getItems()}
hoặc
${@cartService.items}
```

```
@Service(“cart”)
public class CartService{
    List<CartItem> getItems(){
        return null;
    }
    …
}
```

Truy xuất bean bằng id

```
${@cart.getItems()}
hoặc
${@cart.items}
```

FPT Education

FPT POLYTECHNIC

cdio™

Conceive Design Implement Operate

SPRING BEANS

SLIDE 5.2

www.poly.edu.vn

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

SỬ DỤNG @SESSIONSCOPE

SỬ DỤNG @SERVICE HOẶC @COMPONENT

```
@SessionScope
@Service
public class CookieService {
        ...
}
```

TẠO VÀ CẤU HÌNH NẠP SPRING BEAN

```
public class CookieService {...}

@Configuration
public class BeanConfig {
        @SessionScope
        @Bean
        public CookieService getCookieService() {
                return new CookieService();
        }
}
```

- Bean được tạo và nạp vào hệ thống tại thời điểm mỗi session được tạo ra và bị giải phóng khỏi hệ thống khi session hết hạn (timeout)

Inversion of Control (IoC)

IoC là gì?

- Đảo ngược điều khiển (Ioc) là sự thay đổi điều khiển của chương trình một cách tùy biến bằng cách thay thế một vài thành phần nào đó của chương trình mà không cần phải xây dựng lại chương trình.
- Trong Spring sử dụng interface thay vì class để tham chiếu đến bean được tiêm vào, làm như vậy sẽ tránh được sự phụ thuộc của chương trình vào bean class.
- Khi cần thay thế bean class này bằng bean class khác chúng ta chỉ việc cấu hình nạp thay thế một cách dễ dàng.

IoC Example – Tiêm (tham chiếu đến) bean

```
@Controller
public class CrudController {
        @Autowired
        CrudService crudService;

        @RequestMapping("/crud/create")
        public String create(Model model) {
                crudService.create();
                return "/crud";
        }
}
```

```
public interface CrudService {
        void create();
}
```

IoC Example – Nạp bean vào hệ thống

```
@Service
public class FileCrudService implements CrudService{
        @Override
        public void create() {
                System.out.println("Data is added to file");
        }
}
```

```
@Service
public class OracleCrudService implements CrudService{
        @Override
        public void create() {
                System.out.println("Data is inserted to OracleDB");
        }
}
```

Bạn có thể nạp 1 trong 2 bean class này vào hệ thống mà controller không cần phải viết lại. Nghĩa là khi làm việc với file thì nạp FileCrudService và khi làm việc với CSDL Oracle thì nạp OracleCrudService và cứ thế bạn có thể nâng cấp, thay đổi tùy vào môi trường của khách hàng

THIẾT KẾ CÁC SPRING BEAN TIỆN ÍCH

GIỚI THIỆU CÁC BEAN TIỆN ÍCH

- FileService
  - Làm việc với file upload
- CookieService
  - Làm việc với cookie
- MailService
  - Gửi email
- AuthService
  - Làm việc với thông tin của người đăng nhập
- CartService
  - Quản lý giỏ hàng điện tử

THIẾT KẾ COOKIESERVICE

```
public interface CookieService {
        /** Tạo và gửi cookie về trình duyệt */
        Cookie create(String name, String value, int expiry);
        /** Đọc cookie */
        Cookie get(String name);
        /** Đọc giá trị cookie */
        String getValue(String name);
        /** Xóa cookie */
        void delete(String name);
}
```

- Tạo lớp CookieServiceImpl thực thi theo interface CookieService.
- Tiêm HttpServletRequest và HttpServletRessponse
- Cài đặt mã cho các phương phương thức

THIẾT KẾ FILESERVICE

```
public interface FileService {
        /** Lưu file upload vào thư mục của website */
        File save(MultipartFile file, String folder);
        /** Đọc file từ thư mục của website */
        byte[] read(String name, String folder);
}
```

- Tạo lớp FileServiceImpl thực thi theo interface FileService.
- Tiêm ServletContext để làm việc với đường dẫn ảo (virtual path) folder
- Cài đặt mã cho các phương phương thức

THIẾT KẾ AUTHSERVICE

```
public interface AuthService {
        /** Lấy tên đăng nhập */
        String getUsername();
        /** Lấy vai trò của người đăng nhập */
        List<String> getRoles();
        /** Kiểm tra đăng nhập hay chưa */
        boolean isAuthenticated();
        /** Kiểm tra vai trò của người đăng nhập */
        boolean hasAnyRoles(String...roles);
}
```

- Tạo lớp AuthServiceImpl thực thi theo interface AuthService.
- Tiêm HttpSession để đọc User đăng nhập được lưu trong session
- Cài đặt mã cho các phương phương thức

THIẾT KẾ MAILSERVICE

```
public interface MailService {
    /** Gửi email */
    void send(MailMessage mail);
    /** Gửi email đơn giản */
    default void send(String to, String subject, String body) {
        this.send(MailMessage.builder().to(to).subject(subject).body(body).build());
    }
    @Builder
    @Data
    public static class MailMessage{
        @Default
        String from = "FPT Polytechnic <poly@fpt.edu.vn>";
        String to, cc, bcc, subject, body, files;
    }
}
```

- Tạo lớp MailServiceImpl thực thi theo interface MailService.
- Tiêm JavaMailSender để gửi email
- Cài đặt mã cho các phương phương thức send(MailMesssage)

THIẾT KẾ CARTSERVICE

```
public interface CartService {
        /** Thêm vào giỏ hàng */
        CartItem add(Integer itemId);
        /** Lấy từ giỏ hàng */
        CartItem get(Integer itemId);
        /** Cập nhật số lượng */
        CartItem updateQuantity(Integer itemId, int newQuantity);
        /** Xóa khỏi giỏ hàng */
        void remove(Integer itemId);
        /** Xóa sách giỏ hàng */
        void clear();
        /** Lấy các mặt hàng trong giỏ */
        List<CartItem> getItems();
        /** Tính tổng số lượng */
        int getTotalQuantity();
        /** Tính tổng số tiền */
        double getTotalAmount();
}
```

- Tạo lớp CartServiceImpl thực thi theo interface CartService chú thích bởi **@SessionScope**
- Cài đặt mã cho các phương phương

```
@Data
@Builder
public class CartItem{
        Integer id;
        String name;
        double price;
        @Default
        int quantity = 1;
        public double getAmount() {
                return this.quantity*this.price;
        }
}
```

SUMMARY

- ☑ PHẦN I:
  - ☑ GIỚI THIỆU SPRING BEANS
  - ☑ SỬ DỤNG SPRING BEANS DỰNG SẴN
  - ☑ XÂY DỰNG VÀ CẤU HÌNH SPRING BEAN
- ☑ PHẦN II
  - ☑ TÌM HIỂU BEAN SCOPES
  - ☑ TÌM HIỂU VỀ DI AND IOC
  - ☑ XÂY DỰNG CÁC BEAN TIỆN ÍCH THƯỜNG DÙNG


FPT Education
FPT POLYTECHNIC
Thank you