Web-based programming with Java

SPRING BEANS

ThS. Nguyễn Nghiệm
0913.745.789 - NghiemN@fpt.edu.vn

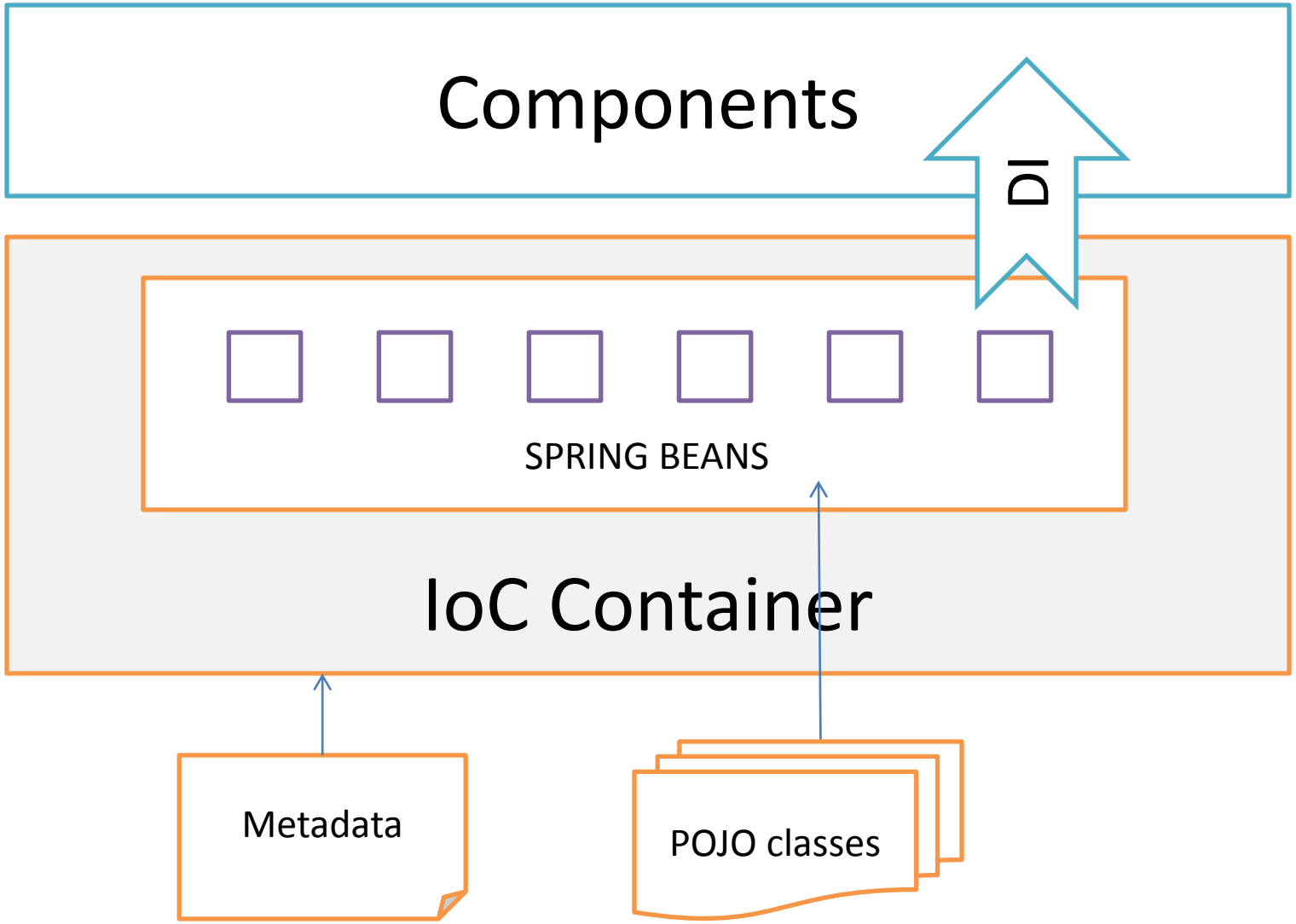
Mục tiêu

- Hiểu được cơ chế IoC
- Tạo, khai báo và sử dụng được Spring BEAN
- Sử dụng các annotation khai báo Spring BEAN
- Xây dựng được các bean tiện ích

NỘI DUNG

- Hiểu được dependence và non-dependence
- Tìm hiểu cơ chế IoC của Spring
- Xây dựng, khai báo và sử dụng Spring Bean
- Xây dựng bean CookieService
- Sử dụng JavaMailSender
- Sử dụng các annotation tự khai báo bean

SPRING BEANS



Spring Beans

- Spring Beans là các bean được Spring tạo ra và quản lý.
- Spring Beans được tạo ra từ các đối tượng của hệ thống hoặc từ bên ngoài thông qua các cấu hình.
- Các bean này sẵn sàng phục vụ các thành phần khác trong ứng dụng khi cần bằng cách "tiêm" chúng vào và sử dụng.

INVERSION OF CONTROL

- IoC là nguyên tắc đảo ngược luồng điều khiển của chương trình.

  Thay vì người lập trình điều khiển luồng của chương trình thì các thành phần bên ngoài thực hiện điều đó. Tương tự chúng ta cắm một cái gì đó vào cái gì khác.

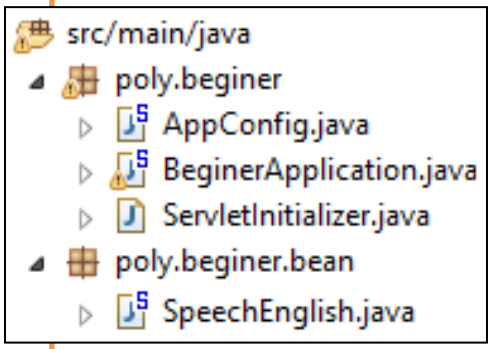
- IoC container thu thập thông tin cấu hình (xml, annotation, code java) để tạo và quản lý các SPRING BEAN sẵn sàng phục vụ cho ứng dụng.

IoC Benifits

- Tránh sự phụ thuộc giữa các đối tượng
- Tạo môi trường mở
- Dễ dàng tháo lắp các thành phần mở rộng
- Dễ dàng kiểm thử

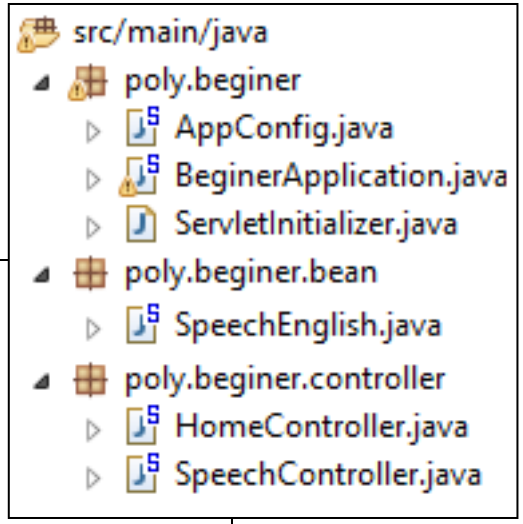
IoC Example

```java
@Configuration
public class AppConfig {
	@Bean
	public SpeechEnglish getSpeechEn()
		return new SpeechEnglish();
	}
}
```




```java
public class SpeechEnglish{
	@Override
	public String getGreeting() {
		return "Hello everybody";
	}
}
```

DEPENDENCE INJECTION




```
@Controller
public class SpeechController {
	@Autowired
	SpeechEnglish speech;

	@ResponseBody
	@RequestMapping("say")
	public String say() {
		return speech.getGreeting();
	}
}
```

DEPENDENCE INJECTION

- DI là cách để "tiêm" một thành phần được quản lý bởi IoC container vào một thành phần khác
- Bạn có thể tiêm bằng XML hoặc annotation. Spring cung cấp 2 annotation để tiêm
  - @**Autowired**
    - Được sử dụng để phân biệt theo kiểu
  - @**Qualifier**
    - Được sử dụng để phân biệt theo id
- Có 3 cách tiêm: vào field, constructor và setter

DEPENDENCE INJECTION

- @Autowired được sử dụng để tiêm bean dựa vào kiểu. Sau đây là 3 cách tiêm:

```
@Autowired
SpeechEnglish speech;

@Autowired
public SpeechController(SpeechEnglish speech) {
	this.speech = speech;
}

@Autowired
public void setSpeech(SpeechEnglish speech) {
	this.speech = speech;
}
```

DEPENDENCE INJECTION

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

	@ResponseBody
	@RequestMapping("say")
	public String say() {
		return speech.getGreeting();
	}
}
```

```java
public interface Speech {
	String getGreeting();
}
```

```java
public class SpeechEnglish implements Speech{
	@Override
	public String getGreeting() {
		return "Hello everybody";
	}
}
```

DI BY BEANID

```java
@Configuration
public class AppConfig {
	@Bean("en")
	public SpeechEnglish getSpeechEn() {
		return new SpeechEnglish();
	}

	@Bean("vi")
	public SpeechVietnamese getSpeechVi() {
		return new SpeechVietnamese();
	}
}
```

```java
@Autowired
@Qualifier("vi")
Speech speech;

@Autowired
public SpeechController(@Qualifier("vi") Speech speech) {
	this.speech = speech;
}

@Autowired
public void setSpeech(@Qualifier("en") Speech speech) {
	this.speech = speech;
}
```

BEAN TỰ KHAI BÁO

- Sử dụng @Component([id]), @Service, @Repository cho bean class và không cần khai báo trong file cấu hình

```
@Component("en")
public class SpeechEnglish implements Speech{
	@Override
	public String getGreeting() {
		return "Hello everybody";
	}
}
```

```
@Component("vi")
public class SpeechVietnamese implements Speech{
	@Override
	public String getGreeting() {
		return "Xin chào mọi người";
	}
}
```

HttpCookie

ĐỀ MÔ

Bean Scope

- Spring Beans được tạo ra khi nào? Duy trì được bao lâu? Chia sẽ cho những thành phần nào?
- Bean Scopes
  - Singleton: mặc định
  - @ApplicationScope
  - @SessionScope
  - @RequestScope

Bean Scope

- Mã cấu hình

```
@SessionScope
@Bean("vi")
public SpeechVietnamese getSpeechVi() {
	return new SpeechVietnamese();
}
```

- Bean tự khai

```
@SessionScope
@Component("vi")
public class SpeechVietnamese implements Speech{
	@Override
	public String getGreeting() {
		return "Xin chào mọi người";
	}
}
```

SENDING EMAIL

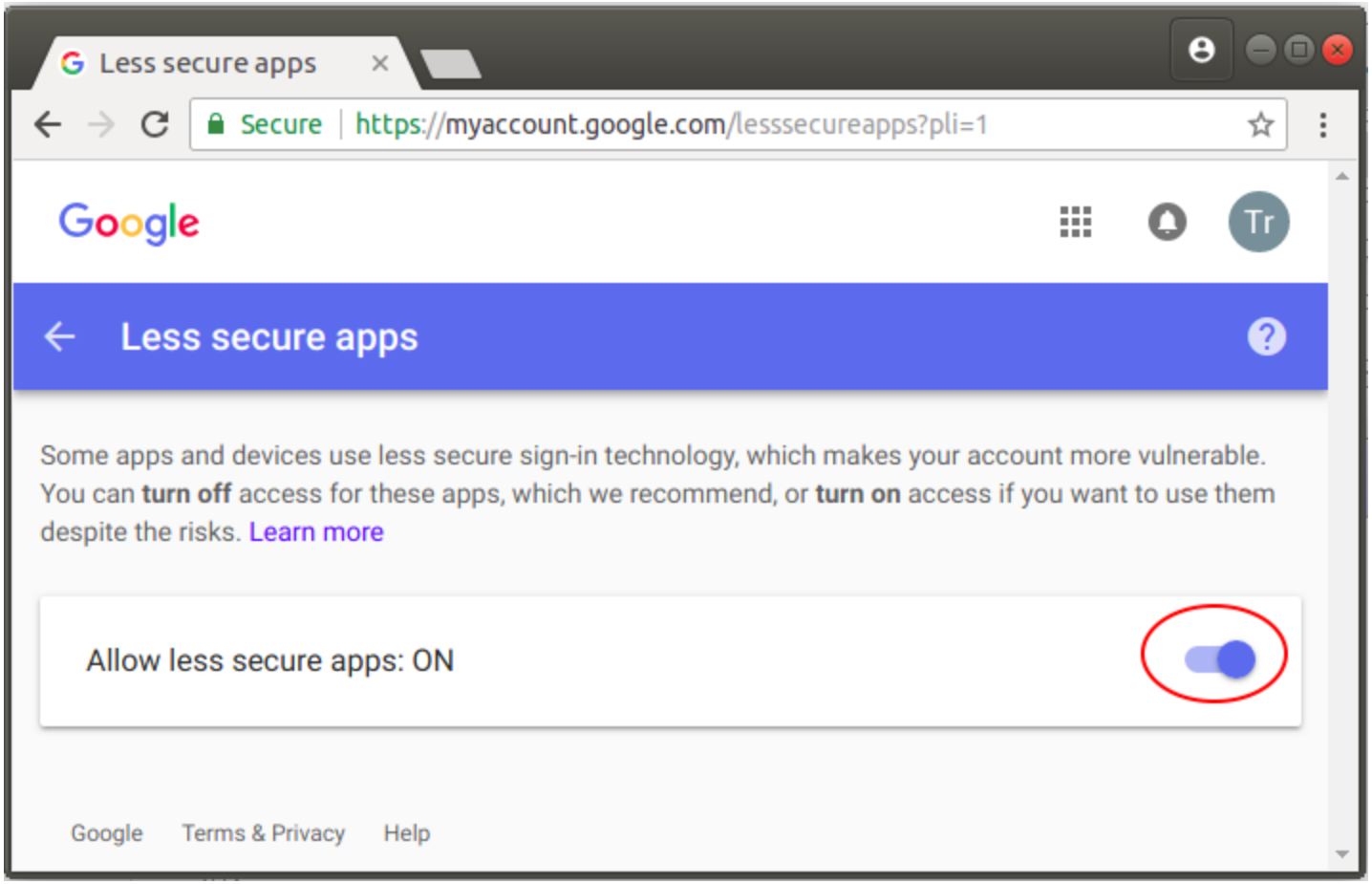
- Interface JavaMailSender và bean JavaMailSenderImpl được cung cấp để gửi email.
- SimpleMailMessage tạo mail văn bản
- MimeMessageHelper hỗ trợ gửi email HTML và đính kèm file.

CẤU HÌNH

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

@Bean
public JavaMailSender getMailSender() {
  JavaMailSenderImpl mailSender = new JavaMailSenderImpl();
  mailSender.setHost("smtp.gmail.com");
  mailSender.setPort(587);

  mailSender.setUsername("user@gmail.com");
  mailSender.setPassword("password");

  Properties props = mailSender.getJavaMailProperties();
  props.put("mail.transport.protocol", "smtp");
  props.put("mail.smtp.auth", "true");
  props.put("mail.smtp.starttls.enable", "true");
  props.put("mail.debug", "true");

  return mailSender;
}
```

KÍCH HOẠT EMAIL

- https://myaccount.google.com/lesssecureapps



SEND TEXT EMAIL

```
@Controller
@RequestMapping("mailer")
public class MailerController {
        @Autowired
        JavaMailSender mailer;

        @ResponseBody
        @RequestMapping("send-text")
        public String sendText() {
         SimpleMailMessage message = new SimpleMailMessage();
         message.setFrom("Sender <sender@gmail.com>");
         message.setTo("receiver@gmail.com");
         message.setReplyTo(message.getFrom());
         message.setSubject("Tiêu đề mail");
         message.setText("Chào quí vị các bạn!");

         mailer.send(message);
         return "OK";
        }
}
```

Send HTML Email with Attach

```
@Autowired
JavaMailSender mailer;

@ResponseBody
@RequestMapping("send-html")
public String sendHtml() throws MessagingException {
 MimeMessage message = mailer.createMimeMessage();

 MimeMessageHelper helper = new MimeMessageHelper(message, true, "utf-8");
 helper.setFrom("Sender <sender@gmail.com>");
 helper.setTo("receiver@gmail.com");
 helper.setReplyTo("Sender <sender@gmail.com>");
 helper.setSubject("Tiêu đề email");
 helper.setText("Chào <b>quí vị đại biểu</b>!", true);
 // Attachment 1
 File file = new File("C:\\temp\\logs.txt");
 helper.addAttachment(file.getName(), file);

 mailer.send(message);

 return "OK";
}
```

SEND EMAIL

ĐỀ MÔ

Lesson Summary

- IoC & DI
- Tạo, cấu hình và tiêm Spring Bean
- Một số cách tiêm
- Bean tự khai
- Xây dựng CookieService
- Mô hình MultiLayer
- Scope của bean
- Sử dụng JavaMailSender