Spring Breakout Session
Spring and IoC

by Vlad Costel Ungureanu
for FII UAIC,
Learn Stuff and Endava Pass It On

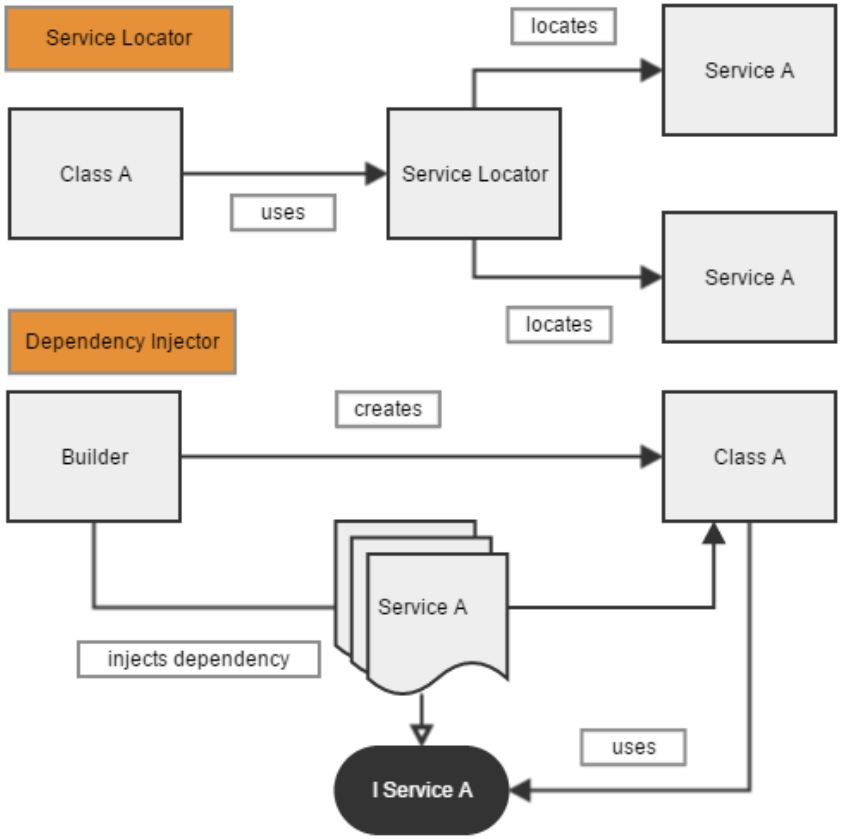
Spring Breakout Session

Spring and Inversion of Control

- Inversion of Control
- IoC Container
- Spring Beans
- Spring Configuration with XML
- Spring Annotations
- Java Annotations
- Overview

Inversion of Control

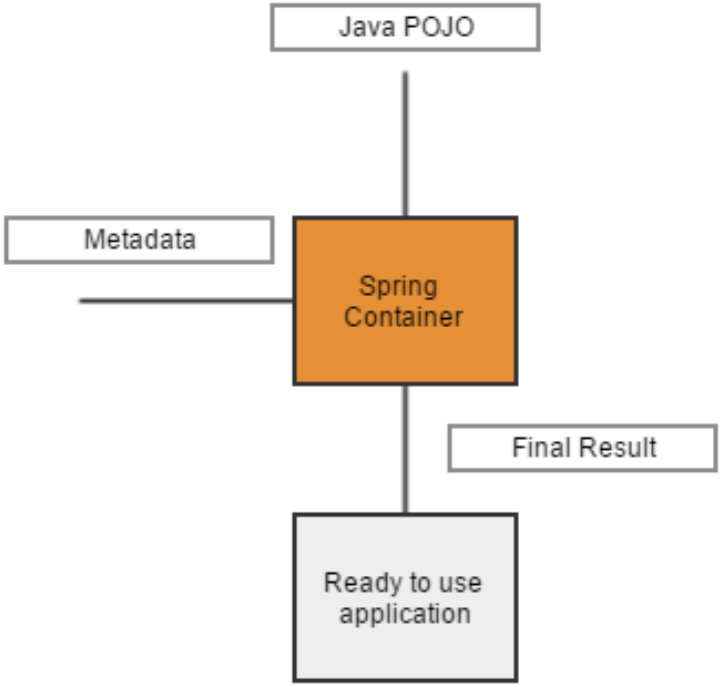
- Generic term referring to assigning the responsibility of choosing an appropriate implementation for an interface at runtime instead of letting a developer specify a particular implementation before runtime.



Inversion of Control

- Instead of a class specifying it's dependencies, they are injected in that class at runtime.
- Implementations can be injected through constructors, setters and through service look-up.
- This ensures that implementations and configurations are decoupled
- This way you can change implementation just by changing configuration
- In testing, implementations can be substituted with Mocks (object instances that simulate functionality)
- Dependencies can be easier to identify

IoC Container



Spring BeanFactory Container

- simplest container
- standard apps
- facilities bean operations

Spring ApplicationContext Container

- web applications
- property files management
- publish application events

Spring Beans

Scope	Description
singleton	A single instance within the container.
prototype	Any number of instances in the container.
request	A single instance per request (only for web apps).
session	A single instance per session (only for web apps).
global-session	A single instance per deployment (only for web apps).

LEARN stuff IO

Spring Configuration with XML

```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"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
           http://www.springframework.org/schema/beans/spring-beans.xsd">

  <bean id="..." class="...">
    <!-- collaborators and configuration for this bean go here -->
  </bean>

  <bean id="..." class="...">
    <!-- collaborators and configuration for this bean go here -->
  </bean>

  <!-- more bean definitions go here -->

</beans>
```

```java
// create and configure beans
ApplicationContext context =
    new ClassPathXmlApplicationContext(new String[] {"services.xml", "daos.xml"});

// retrieve configured instance
PetStoreServiceImpl service = context.getBean("petStore", PetStoreServiceImpl.class);

// use configured instance
List userList = service.getUsernameList();
```

Spring Annotations

- They can be used alongside XML configurations.
- Requires the following configuration in the XML file:

<context:annotation-config />

- For Dependency Injection:
- @Required for setter injection; makes instance injection mandatory
- @Autowired general annotation for dependency injection
- @Qualifier when there are multiple implementations for a single interface and we need to specify the one we want
- @Resource for injecting a specific instance (Autowired + Qualifier).

- Spring components that can be injected:
- @Component
- @Repository
- @Service
- @Controller

Java Annotations

- @Bean indicates that a method is responsible for creating a bean instance (xml tag <bean />)
- @Scope specifies the scope of the bean
- @Configuration indicates the fact that a class hold configurations and bean definitions

Assignments

- ✓Using the Spring container create different types of beans, with different scopes
- ✓Add logging to the constructors of each bean to see how they are created and used within the application
- ✓Use different types of Dependency Injection on the created beans to understand how Spring can be used, and especially how Spring features can be used in your application
- ✓Experiment with Maven and Maven plugins to create a WAR, EAR and JAR from your application

THANK YOU!

Vlad Costel Ungureanu
ungureanu_vlad_costel@yahoo.com

This is a free course from LearnStuff.io
– not for commercial use –