CS544 EA

Spring

Dependency Injection: Annotations

Annotation Config

- If you don’t use component scan you have to tell Spring to look for DI annotations (in XML)

```
<context:annotation-config/>
<bean id="customerService" class="cs544.spring12.di.anno.CustomerService" />
<bean id="customerDao" class="cs544.spring12.di.anno.CustomerDao"/>
```

```
@Configuration
public class Config {
	@Bean
	public CustomerService customerService() {
		return new CustomerService();
	}
	@Bean
	public CustomerDao customerDao() {
		return new CustomerDao();
	}
}
```

Java Config automatically looks for annotations inside any bean it knows about

@Autowired

- Tells Spring to autowire a property

```
package cs544.spring12.di.anno;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.stereotype.Service;

@Service
public class CustomerService {
	@Autowired
	private ICustomerDao customerDao1;

	public void sayHello() {
		System.out.println("Hello");
	}
}
```

Defaults to Autowiring by Type

3 possibilities

- There are **3 annotations** you can use to specify **dependency injection** (all give the same result)
  - @Autowired (original spring annotation)
  - @Resource (JSR250 annotation / J2EE answer)
  - @Inject (JSR330 annotation related to CDI)

For a comparison see: https://www.sourceallies.com/2011/08/spring-injection-with-resource-and-autowired/

@Qualifier

- In order to **specify a bean name** you can use the @Qualifier annotation

```
package cs544.spring12.di.anno;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.stereotype.Service;

@Service
@Qualifier("customerService")
public class CustomerService {
	@Autowired
	@Qualifier("customerDao")
	private ICustomerDao customerDao1;

	public void sayHello() {
		System.out.println("Hello");
	}
}
```

Works to specify the id /name of a bean when declaring

Works to specify the id / name of a bean when injecting

Bean Lookup

- **`@Autowire`** and **`@Inject`**
  1) Matches by Type
  2) Restricts by Qualifiers
  3) Matches by Name

- **`@Resource(name="bean")`**
  1) Matches by Name
  2) Matches by Type
  3) Restricts by Qualifiers (ignored if match is found by name)

Property

- You can specify DI on a property
  - Spring **uses reflection** to inject

```
package cs544.spring12.di.anno;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.stereotype.Service;

@Service
@Qualifier("customerService")
public class CustomerService {
	@Autowired
	@Qualifier("customerDao")
	private ICustomerDao customerDao1;

	public void sayHello() {
		System.out.println("Hello");
	}
}
```

No setter or constructor needed to inject this property

Setter

- Setter injection is used if the annotation is on a **setter method**

```java
package cs544.spring13.di.anno;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

@Service
public class CustomerService {
	private ICustomerDao customerDao;

	@Autowired
	public void setCustomerDao(ICustomerDao customerDao) {
		this.customerDao = customerDao;
	}

	public void sayHello() {
		System.out.println("Hello");
	}
}
```

Constructor

- Constructor injection if the annotation is placed **on a constructor**

```
package cs544.spring14.di.anno;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

@Service
public class CustomerService {
	private ICustomerDao customerDao;

	@Autowired
	public CustomerService(ICustomerDao customerDao) {
		this.customerDao = customerDao;
	}

	public void sayHello() {
		System.out.println("Hello");
	}
}
```