CS544 EA

Spring:

Beans & Application Context

Application Context

- Reads the given config (file) on startup
- Creates the specified objects (beans)
- Connects them together with DI (as indicated)
- Creates Proxy object when needed (for AOP)

The Application Context = The Spring Container

Basic Spring Application

Application.java

```
package cs544.spring01.helloworld;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class App {
  public static void main(String[] args) {
    ApplicationContext context = new
                    ClassPathXmlApplicationContext("cs544/spring01/helloworld/springconfig.xml");
    CustomerService customerService = context.getBean("customerService", CustomerService.class);
    customerService.sayHello();
  }
}
```

CustomerService.java

```
package cs544.spring01.helloworld;

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

springconfig.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="customerService" class="cs544.spring01.helloworld.CustomerService" />
</beans>
```

Spring Bean

- A Spring Bean is an object created by Spring
  - By default Spring creates a **single object** for a class (like a singleton)
  - By default all beans are created **right away** at startup (eager, not lazy)

Creating an Application Context

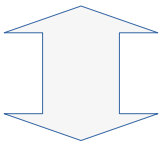
- Config files can be written in **XML or Java**
  - XML can be loaded from ClassPath or FileSystem
  - Java uses a class on the ClassPath

```
package cs544.spring02.context;

import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import org.springframework.context.support.FileSystemXmlApplicationContext;

public class App {
  public static void main(String[] args) {
    ApplicationContext context;
//    context = new ClassPathXmlApplicationContext("cs544/spring02/context/springconfig.xml");
//    context = new FileSystemXmlApplicationContext("//home/mzijlstra/springconfig.xml");
    context = new AnnotationConfigApplicationContext(Config.class);

    CustomerService customerService = context.getBean("customerService", CustomerService.class);
    customerService.sayHello();
  }
}
```

Java Configuration

springconfig.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="customerService" class="cs544.spring01.helloworld.CustomerService" />
</beans>
```



Config.java

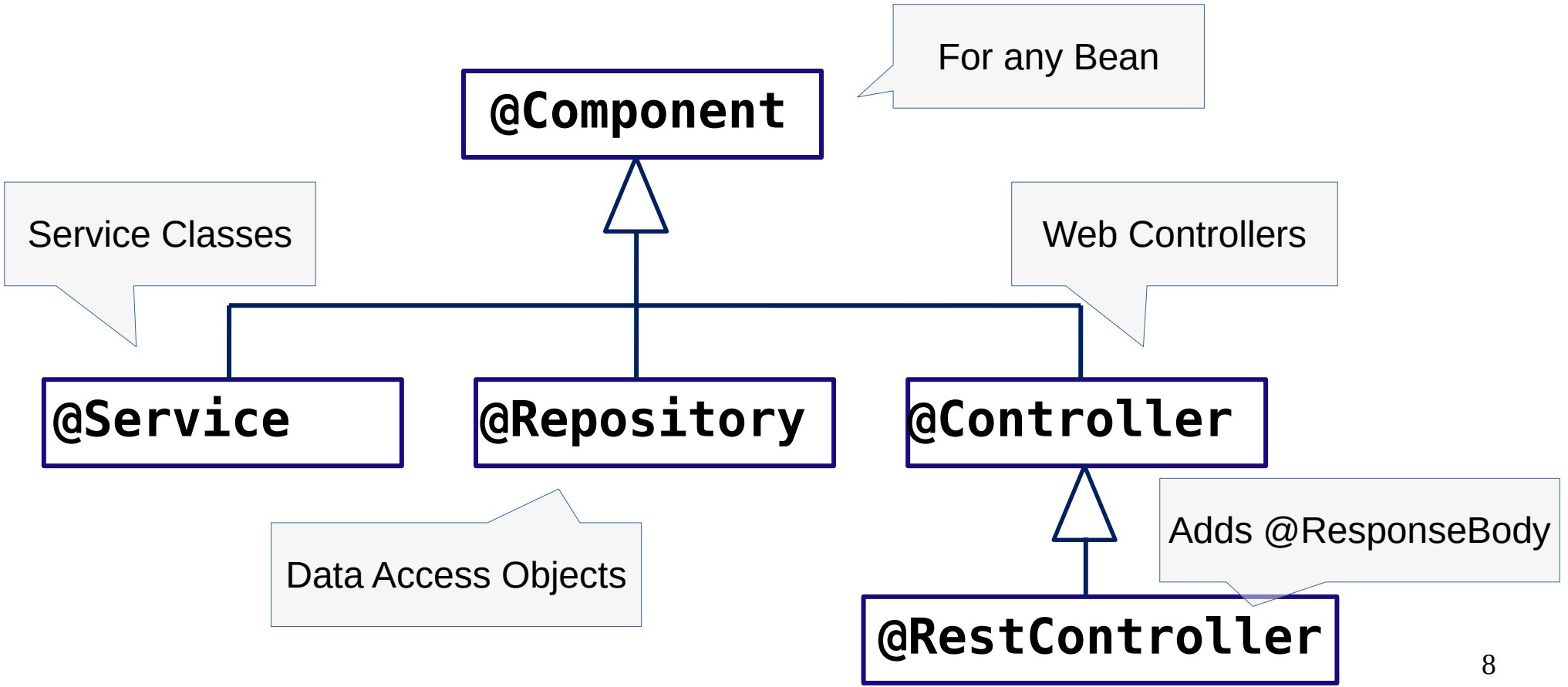
```java
package cs544.spring02.context;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

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

Component Scan

- You have to start with **XML** or **Java**, but:
  - You can tell spring to look for config **annotations**

- How will it find these annotations?
  - Reading all classes on the classpath takes too long!
  - Scan for ‘components’ (beans) in a certain package

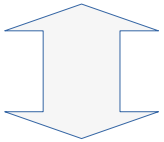
Component Scan



Component Scan 1/2 (Config)

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

       <context:component-scan base-package="cs544.spring03.scan" />
</beans>
```



To use context tags you have to add the context namespace

```java
package cs544.spring03.scan;

import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;

@Configuration
@ComponentScan("cs544.spring03.scan")
public class Config {
}
```

Component Scan 2/2

Application.java

```
package cs544.spring03.scan;

import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class App {
  public static void main(String[] args) {
    ApplicationContext context;
//    context = new ClassPathXmlApplicationContext("cs544/spring03/scan/springconfig.xml");
    context = new AnnotationConfigApplicationContext(Config.class);

    CustomerService customerService = context.getBean("customerService", CustomerService.class);
    customerService.sayHello();
  }
}
```

CustomerService.java

```
package cs544.spring03.scan;

import org.springframework.stereotype.Service;

@Service
public class CustomerService {
      public void sayHello() {
            System.out.println("Hello from CustomerService");
      }
}
```

Summary

- The application context reads a configuration (XML or Java) and creates objects (beans)
- The config file can indicate that further configuration is found in annotations.
  - For bean creation this works with classpath scanning, where you give the base package

Science of Consciousness

- Unity in Diversity. You can get the same result in many different ways