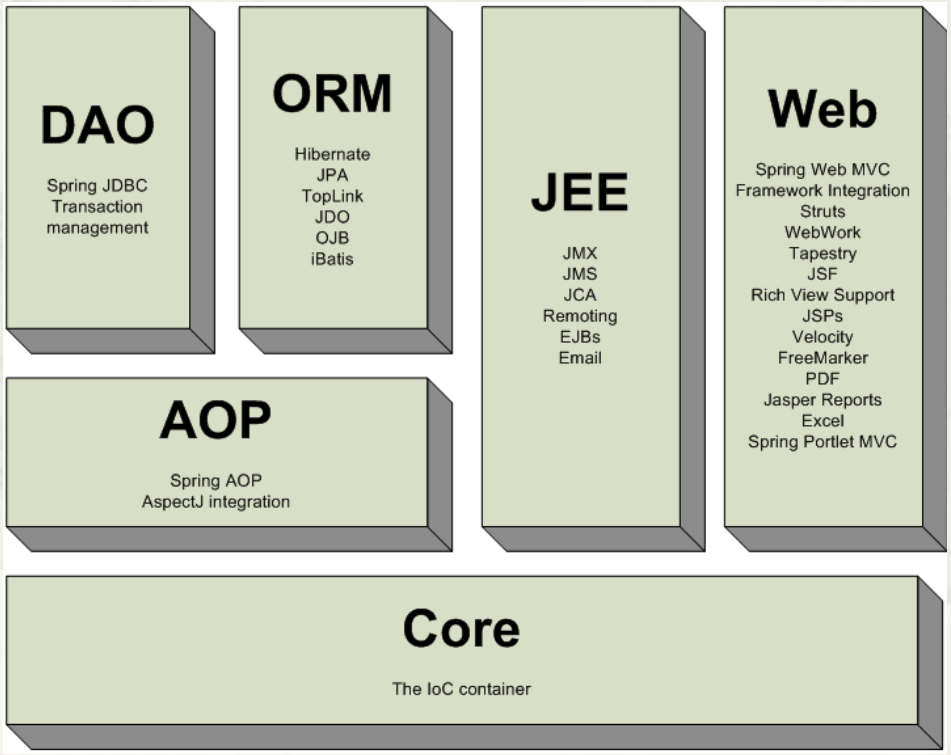
企业级框架 Spring

Unit01

Spring简介

Spring简介

- Spring是一个开源框架，为JavaEE应用提供多方面的解决方案，用于简化企业级应用的开发。



Spring简介

Spring包含功能如下：

- IOC：控制反转，Spring的核心功能
- AOP：面向方面编程
- Web：MVC结构的实现、与其他Web技术整合
- DAO：与JDBC整合和事务管理
- ORM：与ORM框架整合
- JEE：与JavaEE服务整合

Spring IOC

Sping IOC

全称是Inversion of Control，控制反转。是指程序中对象的获取方式发生反转，由最初的new方式创建，转为由框架创建、注入，这样可以降低对象之间的耦合度。

IOC主要作用是管理程序组件，创建组件对象和维护组件对象之间的关系。

Spring容器

- 在Spring中，任何的Java类都被当成Bean组件，通过容器管理和使用。
- Spring容器实现了IOC和AOP机制
- Spring容器有ApplicationContext和BeanFactory两种类型

Spring容器

- ApplicationContext继承自BeanFactory，提供了更多的方法，建议使用ApplicationContext
- 从classpath下加载配置文件实例化

```
String conf = "applicationContext.xml";
ApplicationContext ac =
        new ClassPathXmlApplicationContext(conf);
```

- 从文件系统中加载配置文件实例化

```
String conf = “D:\\applicationContext.xml";
ApplicationContext ac =
        new FileSystemXmlApplicationContext(conf);
```

Spring容器

- 从classpath下加载配置文件实例化

```
String conf = "applicationContext.xml";
Resource resource = newClassPathResource(conf);
BeanFactory bean = new XmlBeanFactory(resource);
```

- 从文件系统中加载配置文件实例化

```
String conf = “D:\\applicationContext.xml";
Resource resource = new FileSystemResource (conf);
BeanFactory bean = new XmlBeanFactory(resource);
```

Spring容器

- 在容器配置文件applicationContext.xml中添加Bean定义，等价于将Bean组件放入Spring容器，后续由Spring负责创建对象。

<bean id="标识符" class="Bean类型"/>

Bean对象的创建

- Spring容器创建Bean对象的方法有以下3种
  - 使用构造器来实例化
  - 使用静态工厂方法实例化
  - 使用实例工厂方法实例化

Spring容器

- 从ApplicationContext容器获取Bean组件对象的方法如下

类型 变量名 = 容器对象.getBean(“标识符”,组件类型)

Bean对象的创建

* Bean组件定义格式如下

```
//构造器配置
<bean id="标识符" class="包名.类名"/>
//静态工厂方法配置
<bean id="标识符" class="包名.类名"
      factory-method="静态方法名" />
//动态工厂方法配置
<bean id="标识符" factory-bean="对象id"
      factory-method="方法名" />
```

Bean对象的作用域

- 指定Spring容器创建Bean对象的作用域

作用域	描述
singleton	在每个Spring IoC容器中一个bean定义对应一个对象实例，默认项
prototype	一个bean定义对应多个对象实例
request	在一次HTTP请求中，一个bean定义对应一个实例，仅限于Web环境
session	在一个HTTP Session中，一个bean定义对应一个实例，仅限于Web环境
global Session	在一个全局的HTTP Session中，一个bean定义对应一个实例；仅在基于portlet的Web应用中才有意义，Portlet规范定义了全局Session的概念

Bean对象的初始化

* 指定Spring容器创建Bean对象的初始化方法

```
<beans default-init-method=“初始化方法名">
   <bean id="标识符" class="类“/>
 </beans>
```

或者

```
 <beans >
    <bean id="标识符" class="类“
              init-method =“初始化方法名"/>
  </beans>
```

Bean对象的销毁

- 指定Spring容器创建Bean对象的销毁方法

```
<beans default-destroy-method=“销毁方法名">
   <bean id="标识符" class="类“/>
 </beans>
```

或者

```
 <beans >
    <bean id="标识符" class="类“
              destroy-method =“销毁方法名"/>
  </beans>
```

Bean延迟实例化

- 默认情况下，容器对象创建时Bean对象就会创建，可以指定实例化延迟

```
<beans default-lazy-init=“true">
   <bean id="标识符" class="类“/>
 </beans>
```

或者

```
 <beans >
    <bean id="标识符" class="类” lazy-init=“true"/>
  </beans>
```

DI依赖注入

Bean对象的注入

- 容器可以建立Bean对象之间的关系，实现技术途径就是DI注入。
- Spring DI注入有setter注入和构造器注入两种

Bean对象的注入-setter注入

- 容器在创建完对象后，通过调用set方法完成信息的注入。

```
<bean id=“ds"
        class="com.xdl.util.DBCPUtil">
    <property name=“dataSource" ref=“dbcp"/>
</bean>
```

Bean对象的注入-构造器注入

* 容器在创建完对象时，通过构造器参数完成信息的注入。

```
<bean id=“ds"
        class="com.xdl.bean.EmpDao">
    <constructor-arg index="0" ref=“dbcp"/>
</bean>
```

Bean对象的注入-自动装配

- 在Bean对象注入时，可以通过自动装配简化XML注入配置。
- <bean/>元素中使用autowire属性指定自动装配规则，属性值如下

属性值	描述
no	禁用自动装配，默认值
byName	根据属性名自动装配。此选项将检查容器并根据名字查找与属性完全一致的bean，并将其与属性自动装配
byType	如果容器中存在一个与指定属性类型相同的bean，那么将与该属性自动装配
constructor	与byType的方式类似，不同之处在于它应用于构造器参数
autodetect	通过bean类来决定是使用*constructor*还是*byType*方式进行自动装配。如果发现默认的构造器，那么将使用*byType*方式

Bean参数注入

Bean参数注入

- Bean对象注入的信息类型可以为字符串、集合、bean对象。

注入字符串

```
<bean id="msg"
class="com.xdl.bean.OracleDataSource">
    <property name=“username" value=“scott"/>

    <property name=“password“ >
        <value>tiger</value>
    </property>

    <property name=“msg“ >
        <null/>
    </property>
</bean>
```

注入集合

- 通过<list/>、<set/>、<map/>及<props/>元素可以定义和设置与Java类型中对应List、Set、Map及Properties的属性值。

注入集合-List

- 注入List集合

```
<bean id=“demo" class="com.xdl.bean.DemoBean">
    <property name=“subjects">
        <list>
            <value>php</value>
            <value>大数据</value>
            <value>html5</value>
        </list>
    </property>
</bean>
```

注入集合-Set

* 注入Set集合

```
<bean id=“demo" class="com.xdl.bean.DemoBean">
    <property name=“subjects">
        <set>
            <value>php</value>
            <value>大数据</value>
            <value>html5</value>
        </set>
    </property>
</bean>
```

注入集合-Map

- 注入Map集合

```
<bean id="demo" class="com.xdl.bean.DemoBean">
    <property name="score">
        <map>
            <entry key=“1001" value=“99"/>
            <entry key=“1002" value=“100"/>
            <entry key="1003" value="98"/>
        </map>
    </property>
</bean>
```

注入集合-Properties

* 注入Properties集合

```
<bean id="demo" class="com.xdl.bean.DemoBean">
    <property name=“dbParams">
        <props>
            <prop key="user">scott</prop>
            <prop key="password">tiger</prop>
        </props>
    </property>
</bean>
```

注入集合-引入

* List、Set、Map、Properties集合也可以先独立定义，再注入的方式使用，这样便于重复利用。

```
<util:list id=“subjectList">
    <value>php</value>
    <value>java</value>
</util:list>
<bean id=“demo" class="com.xdl.bean.DemoBean">
        <property name=“subjects" ref=" subjectList"/>
</bean>
```

注入集合-引入

- List、Set、Map、Properties集合单独定义，使用的标记分别为<util:list >、<util:set> 、<util:map>、<util:properties > 。

Spring表达式注入

- Spring引入了一种表达式语言，它和EL在语法上很相似，可以读取一个bean对象/集合中的数据。

```
<util:properties id=“db“
      location="classpath:db.properties"/>
<bean id="demo" class="com.xdl.bean.OracleDataSource">
    <property name=“username" value="#{db.name}"/>
    <property name=“password" value="#{db.password}"/>
    <property name=“url" value="#{db.url}"/>
</bean>
```

组件扫描

组件扫描

Spring提供了一套基于注解配置的使用方法。使用该方法可以大大简化XML配置信息。

组件扫描

开启组件扫描，可以利用注解方式应用IOC。使用方法如下：

- 在applicationContext.xml中添加启用标记

  <context:component-scan base-package="包路径"/>

- 在组件类中追加以下标记

注解标记	描述
@Component	通用注解
@Repository	持久化层组件注解
@Service	业务层组件注解
@Controller	控制层组件注解

组件扫描

* 扫描组件后，默认id值为类名首字母小写，也可以自定义id，例如

```
@component(“empDao”)
```

* 扫描组件后，默认scope为singleton单例，也可以进行指定，例如

```
@component(“empDao”)
@Scope(“prototype”)
```

组件扫描

* 也可以指定初始化和销毁的方法，例如

```
@Component
public class DemoBean {
    @PostConstruct
    public void init() {
        //初始化回调方法
    }
    @PreDestroy
    public void destroy() {
        //销毁回调方法
    }
}
```

组件扫描

- 将所有Bean组件扫描到Spring容器后，可以使用以下注解指定注入关系
- @Autowired/@Qualifier
  - 可以处理构造器注入和Setter注入
- @Resource
  - 只能处理Setter注入，但大部分情况都是Setter注入

组件扫描

@Value注解可以注入Spring表达式值

- 首先读取db.properties文件，封装成Properties对象

```
<util:properties
id=“db“ location="classpath:db.properties"/>
```

- 然后在属性变量或Setter方法前使用@Value注解

```
@Component
public class MyDataSource{
    @Value("#{db.url}")
    private String url;
}
```

总结和答疑