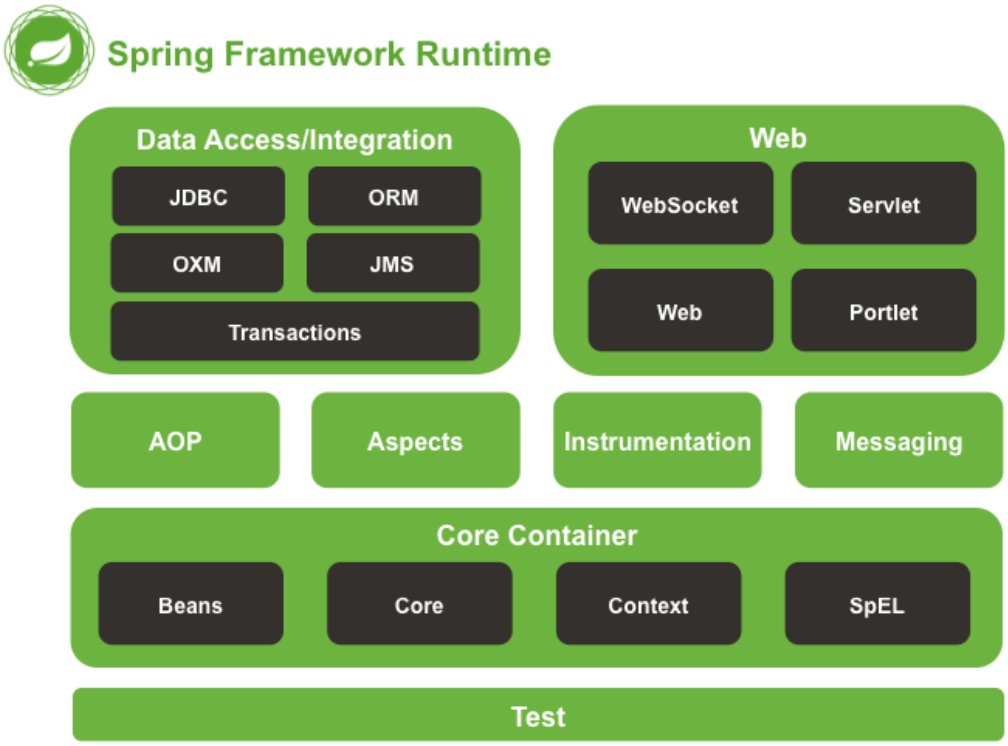
Spring 5

笔记本：	Spring		
创建时间：	2020.08.20 22:00	**更新时间：**	2020.08.21 10:36
作者：	195330205@qq.com		

- https://spring.io

Spring概述

1. Spring 是轻量级的开源的 JavaEE 框架
2. Spring 可以解决企业应用开发的复杂性
3. Spring 有两个核心部分：IOC 和 AOP
    - IOC：控制反转，把创建对象过程交给 Spring 进行管理
    - AOP：面向切面编程，不修改源代码进行功能增强
4. Spring 特点
    - 方便解耦，简化开发
    - AOP 编程支持
    - 方便程序测试
    - 方便和其他框架进行整合
    - 方便进行事务操作
    - 降低 API 开发难度



Spring IOC

1. 什么是 IOC
    - IOC (Inversion of Control)：控制反转，把对象创建和对象之间的调用过程，交给 Spring 进行管理
    - DI (Dependency Injection)：依赖注入，就是注入属性
    - IoC是一种思想，DI是对IOC思想的具体实现
    - 使用 IOC 目的：为了耦合度降低

2. IOC 底层原理
    - XML解析
    - 反射
    - 工厂模式

3. BeanFactory 接口
    - IOC 思想基于 IOC 容器完成，IOC 容器底层就是对象工厂
    - Spring 提供 IOC 容器实现两种方式：（两个接口）
        - BeanFactory：IOC 容器基本实现，是 Spring 内部的使用接口，不提供开发人员进行使用

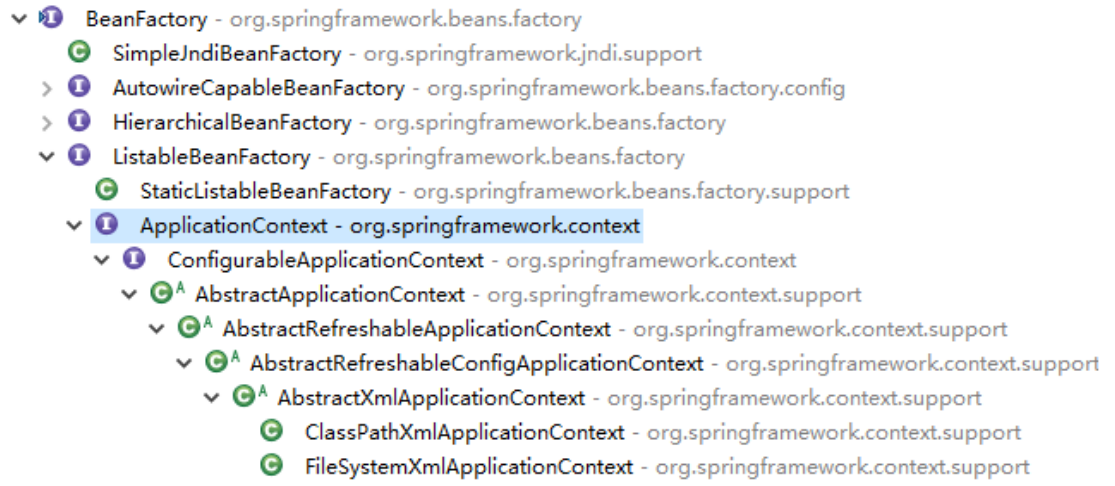
* 加载配置文件时候不会创建对象，在获取对象（使用）才去创建对象

- ApplicationContext：BeanFactory 接口的子接口，提供更多更强大的功能，一般由开发人员进行使用

  * 加载配置文件时候就会把在配置文件对象进行创建

- ApplicationContext 接口的实现类

Type hierarchy of 'org.springframework.beans.factory.BeanFactory':



4. IOC 操作 Bean 管理

   - Bean 管理指的是两个操作：
     - Spring 创建对象
     - Spring 注入属性
   - Bean 管理操作有两种方式
     - 基于 xml 配置文件方式实现
     - 基于注解方式实现

基于 xml 配置文件方式实现

基于 xml 方式创建对象

创建对象时候，默认是执行无参数构造方法完成对象创建

```
<!-- User user = new User(); -->
<bean id="user" class="com.clps.spring5.bean.User">
</bean>
```

基于 xml 方式注入属性

1. **使用 set 方法进行注入**

Step 1 : 创建类，定义属性和对应的 set 方法

```java
public class User {

    private String username;

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

}
```

Step 2 : 在 spring 配置文件配置对象创建，配置属性注入

```xml
<!-- User user = new User(); -->
<bean id="user" class="com.clps.spring5.bean.User">
    <property name="username" value="Tom"></property>
</bean>
```

2. **使用有参构造方法进行注入**

Step 1 : 创建类，定义属性，创建属性对应有参数构造方法

```java
public class Order {

    private String address;

    public Order(String address) {
        this.address = address;
    }

}
```

Step 2 : 在 spring 配置文件中进行配置

```xml
<bean id="order" class="com.clps.spring5.bean.Order">
    <constructor-arg name="address" value="DaLian"></constructor-arg>
</bean>
```

3. **p 名称空间注入**

```xml
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:p="http://www.springframework.org/schema/p"
    xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd">


    <!-- User user = new User(); -->
    <bean id="user" class="com.clps.spring5.bean.User" p:username="Lucy">
        <!-- <property name="username" value="Tom"></property> -->
    </bean>
```

---

xml 注入其他类型属性

1. 字面量

- null

```xml
<property name="address">
    <null/>
</property>
```

- 属性值包含特殊符号

```xml
<property name="address">
    <value><![CDATA[<<大连>>]]></value>
</property>
```

2. 注入属性 - 外部 bean

- 创建两个类 service 类和 dao 类
- 在 service 调用 dao 里面的方法
- 在 spring 配置文件中进行配置

3. 注入属性 - 内部 bean

- 一对多关系：部门和员工
- 一个部门有多个员工，一个员工属于一个部门
- 部门是一，员工是多
- 在实体类之间表示一对多关系，员工表示所属部门，使用对象类型属性进行表示

4. 注入属性 - 级联赋值

---

xml 注入集合属性

1. 注入数组类型属性

2. 注入 List 集合类型属性

3. 注入 Map 集合类型属性

4. 在集合里面设置对象类型值

5. 把集合注入部分提取出来

FactoryBean

Spring 有两种类型 bean：

1. 普通 bean：在配置文件中定义 bean 类型就是返回类型
2. 工厂 bean：在配置文件定义 bean 类型可以和返回类型不一样

Step 1：创建类，让这个类作为工厂 bean，实现接口 FactoryBean
Step 2：实现接口里面的方法，在实现的方法中定义返回的 bean 类型

bean 作用域

- 在 Spring 中，可以设置创建 bean 实例是单实例或多实例
- 默认情况下，bean 是单实例对象
- 设置单实例还是多实例
    - 在 spring 配置文件 bean 标签里面有属性"scope"用于设置单实例还是多实例
    - scope 属性值及区别
        - singleton：单实例，加载 spring 配置文件时候就会创建单实例对象
        - prototype：多实例，不是在加载 spring 配置文件时候创建对象，而是在调用 getBean 方法时候创建多实例对象
        - request：web域对象，表示一次请求中
        - session：web域对象，表示一次会话中

bean 生命周期

从对象创建到对象销毁的过程

1. 通过构造器创建 bean 实例（无参数构造）
2. 为 bean 的属性设置值和对其他 bean 引用（调用 set 方法）
3. 调用 bean 的初始化的方法（需要进行配置初始化的方法）
4. bean 可以使用了（对象获取到了）
5. 当容器关闭时候，调用 bean 的销毁的方法（需要进行配置销毁的方法）

bean 的后置处理器，bean 生命周期有7步

实现 BeanPostProcessor 接口

1. 通过构造器创建 bean 实例（无参数构造）
2. 为 bean 的属性设置值和对其他 bean 引用（调用 set 方法）
3. 把 bean 实例传递 bean 后置处理器的方法 postProcessBeforeInitialization
4. 调用 bean 的初始化的方法（需要进行配置初始化的方法）
5. 把 bean 实例传递 bean 后置处理器的方法 postProcessAfterInitialization
6. bean 可以使用了（对象获取到了）
7. 当容器关闭时候，调用 bean 的销毁的方法（需要进行配置销毁的方法）

基于注解方式实现

Spring AOP
