

Spring_day1

Spring

春天

项目管理框架

整理者:小陈

1. spring框架的引言

spring(春天),生于在2002年,由Rod Johnson创作。Spring框架是一个集**众多设计模式**于一身的**开源的、轻量级**的项目管理框架。致力于JAVAE轻量级解决方案。

轻量级解决方案:提供一个以简单的、统一的、高效的方式构造整个应用,并且可以将单层框架以最佳的组合揉和在一起建立一个连贯的体系

特点: 相对于原来学过的框架而言,spring框架和之前学习的struts2、mybatis 框架有了本质的区别,不是替换原来的某个框架,**而是对其进行整合管理**。

2. spring框架的作用

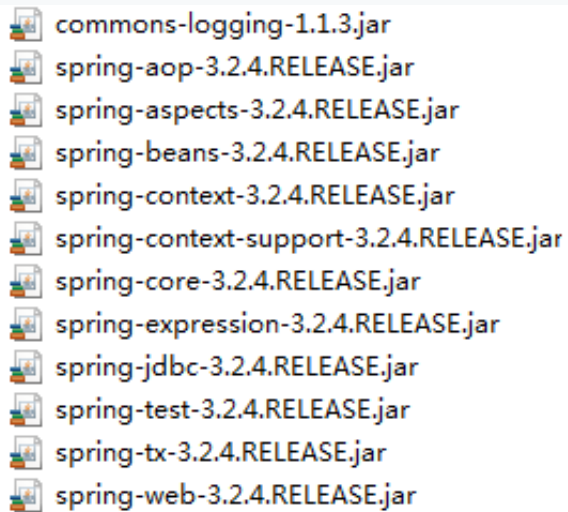
Spring 框架用来管理**创建|使用|销毁**项目中的组件,由于spring 框架可以帮我们生产项目中组件对象,因此也习惯称spring是一个**工厂|容器**。

组件: 项目中的service,dao,action,都是项目中的组件

注意: spring框架通常**不管理对实体类对象创建**

2. spring框架的环境搭建

- 导入spring框架的依赖包



A screenshot of a file explorer window displaying a list of Spring Framework JAR files. Each file is preceded by a small icon representing a JAR file. The files listed are:

- commons-logging-1.1.3.jar
- spring-aop-3.2.4.RELEASE.jar
- spring-aspects-3.2.4.RELEASE.jar
- spring-beans-3.2.4.RELEASE.jar
- spring-context-3.2.4.RELEASE.jar
- spring-context-support-3.2.4.RELEASE.jar
- spring-core-3.2.4.RELEASE.jar
- spring-expression-3.2.4.RELEASE.jar
- spring-jdbc-3.2.4.RELEASE.jar
- spring-test-3.2.4.RELEASE.jar
- spring-tx-3.2.4.RELEASE.jar
- spring-web-3.2.4.RELEASE.jar

- maven项目导入的依赖

```
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-core</artifactId>
  <version>4.3.2.RELEASE</version>
</dependency>
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-beans</artifactId>
  <version>4.3.2.RELEASE</version>
</dependency>
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-web</artifactId>
  <version>4.3.2.RELEASE</version>
</dependency>
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-expression</artifactId>
  <version>4.3.2.RELEASE</version>
</dependency>
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-aop</artifactId>
  <version>4.3.2.RELEASE</version>
</dependency>
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-context</artifactId>
  <version>4.3.2.RELEASE</version>
```

```

</dependency>
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-context-support</artifactId>
    <version>4.3.2.RELEASE</version>
</dependency>
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-aspects</artifactId>
    <version>4.3.2.RELEASE</version>
</dependency>
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-jdbc</artifactId>
    <version>4.3.2.RELEASE</version>
</dependency>

```

- 引入spring框架的配置文件

配置文件名称: **任意名称**

配置文件位置: **项目中根下任意位置**

配置文件的头:

```

<?xml version="1.0" encoding="UTF-8"?>
    <beans xmlns="http://www.springframework.org/schem
a/beans"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema
a-instance"
        xsi:schemaLocation="http://www.springframew
ork.org/schema/beans
        http://www.springframework.org/schema/bean
s/spring-beans-3.2.xsd ">
    </beans>

```

- 配置spring框架配置文件管理组件对象

创建组件对象:

```

UserDAO.java
1 package com.baizhi.dao;
2
3
4 public interface UserDao {
5
6     public void delete(String id);
7
8     public void save();
9
10 }
11
12
13
UserDAOImpl.java
1 package com.baizhi.dao;
2
3 public class UserDaoImpl implements UserDao{
4
5     public void delete(String id) {
6         System.out.println("删除成功:"+id);
7     }
8
9     public void save() {
10         System.out.println("保存一个...");
11     }
12
13 }

```

通过工厂管理:

```

<?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">

    <!-- 管理DAO对象 -->
    <bean id="userDAO" class="com.baizhi.dao.UserDAOImpl"/>
</beans>

```

- 通过启动工厂测试

```

public static void main(String[] args) {

    //加载工厂
    ApplicationContext context = new ClassPathXmlApplicationContext("spring.xml");
    //根据id获取指定对象
    UserDao userDAO = (UserDao) context.getBean("userDAO");
    userDAO.save();

}

```

3. spring中的核心思想

IOC[控制反转]

- **IOC**(inversion of controll)控制反转|**DI**(dependency Injection)依赖注入
将对象的创建由原来(new)的方式转移到配置文件中,交给spring工厂来创建对象spring不仅要创建对象,还要建立类与类之间的关系,因此在控制反转的基础上又提出了依赖注入的概念。

AOP[面向切面编程]

- **AOP**(Aspect Oriental Programing) 面向切面的编程

4. set 注入

- 八种基本类型、日期类型、和数组类型注入

```
<!-- 基本类型+日期类型 -->
<property name="id" value="1"/>
<property name="name" value="张三"/>
<property name="price" value="5.5"/>
<property name="sex" value="false"/>
<property name="birthday" value="2015/12/12 12:12:12"/>
<!-- 数组类型 -->
<property name="strs">
    <array>
        <value>小刘</value>
        <value>小丽</value>
        <value>小好</value>
    </array>
</property>
```

- 引用类型,集合类型注入

```
<!-- 引用类型注入 -->
<property name="userDAO" ref="userDAO"/>
<!-- 集合类型 -->
<property name="list">
    <list>
        <value>小张</value>
        <value>小好</value>
    </list>
</property>
<property name="maps">
    <map>
        <entry key="123" value="小三"></entry>
        <entry key="456" value="小四"></entry>
        <entry key="789" value="九妹"></entry>
    </map>
</property>
<!-- properties -->
<property name="props">
    <props>
        <prop key="driver">oracle.jdbc.OracleDriver</prop>
        <prop key="url">jdbc:oracle:thin@localhost:1521:xe</prop>
        <prop key="username">hr</prop>
        <prop key="password">hr</prop>
    </props>
</property>
```

注意: 引用类型使用`ref`属性注入,基本类型使用`value`属性注入

5. 构造注入

```
<bean id="userDAO" class="zpark.dao.UserDAOImpl">
  <constructor-arg name="id" value="123"/>
  <constructor-arg name="name" value="zhangsna"/>
  <constructor-arg name="price" value="21111.98"/>
  <constructor-arg name="bir" value="2011/10/10 12:23:23"/>
  <constructor-arg name="strs">
    <array>
      <value>张三</value>
      <value>小三</value>
      <value>小白</value>
      <value>小紫</value>
    </array>
  </constructor-arg>
</bean>
```

注意:构造注入并不常用,不过在一些框架类中必须使用构造注入,这里先了解其注入语法即可。

6. 自动注入

- **autowire="byName"**

根据注入的属性名与配置文件中bean的id匹配,一致则注入,不一致报错

- **autowire="byType"**

根据注入的属性类型,与配置文件中的类型匹配,类型一致注入(在多个实现类时,会产生歧义)

注意: 无论使用以上那种方式注入都需要为属性提供**set方法**

7. bean的创建模式[重点]

```
singleton: 单例    默认
           : 全局唯一，一个工厂只创建一次。

prototype: 非单例
           : 全局不唯一，每当使用时，都要重新创建一个新对象

<bean id="xx" class="xx" scope="singleton|prototype"/>
Service, DAO          singleton
struts2的Action       prototype
```

注意:在项目开发中service,dao组件**单例**,struts2的Action必须为**多例**

8. spring bean的生产原理[重点]

- **原理:** 反射 + 无参数构造方法

```
Class<UserDAO> forName = (Class<UserDAO>) Class.forName("zpark.dao.UserDAONewImpl");
UserDAO newInstance = forName.newInstance();
newInstance.save();
```

9. bean的生命周期[重点]

- 何时创建

随着工厂启动, **所有单例bean随之创建 非单例的bean,每次使用时创建**

- 何时销毁

工厂**关闭,所有bean随之销毁** (注意:**spring对多例bean管理松散,不会负责多例bean的销毁**)

10. spring工厂创建对象的好处

- 使用配置文件管理java类,再生产环境中更换类的实现时不需要重新部署,修改文件即可
 - spring默认使用单例的模式创建bean,减少内存的占用
 - 通过依赖注入建立了类与类之间的关系(使java之间关系更为清晰,方便了维护与管理)
-