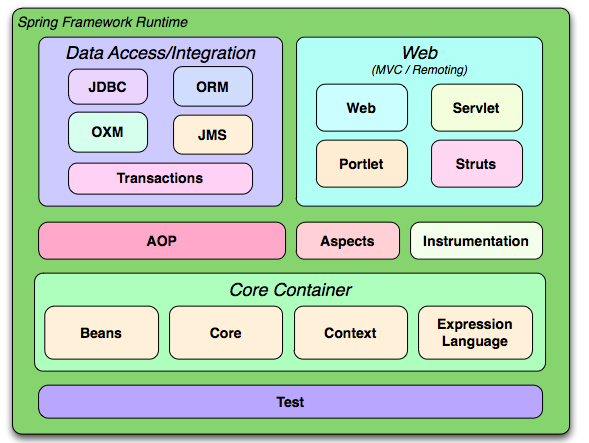
**Spring结构图**



**Spring三种配置**

* xml显示配置bean：
* java代码配置bean：
* 自动化装配bean：适用于自己编写的JavaBean，当需要注入第三方类库时，需要适用上面两种方式

**1、自动化装配**

首先配置扫描包：xml配置和java代码配置

1）xml配置：

<?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"

xmlns:mvc="http://www.springframework.org/schema/mvc"

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-4.1.xsd

http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc-4.1.xsd">

<!-- 配置扫描 -->

<context:component-scan base-package="com.wangdh.spring.demo"/>

</beans>

Java代码配置：

@Configuration

@ComponentScan(basePackages = "com.wangdh.spring.demo")

public class SpringJavaConfig {

}

2）为需要装载的bean加@Component注解

3）在需要实例化的地方加@Autowired注解，可以加载构造函数、setter、方法和字段上

**2、java代码配置**

1）兴建一个JavaConfig类,该类需要添加注解：@Configuration

2）为每个需要注入的对象创建一个方法，该方法需要添加注解：@Bean并返回注入对象的实例

@Bean

**public** Cat cat(){

**return** **new** Cat();

}

3、xml配置：对旧的项目的xml进行维护，新的项目建议适用java代码配置和自动装置

**Spring 常用配置**

**1.Bean的作用域：@Scope**

Singleton：单例模式，Spring的默认配置

Prototype：每次调用新建一个Bean实例

Request：web项目中，每个http请求新建一个Bean实例

Session：web项目中，每个http会话新建一个Bean实例

GlobalSession：portal应用中有用

**2.Bean的初始化和销毁**

Java配置方式：initMethod和destroyMethod

xml配置方式：init-method和destroy-method

注解方式：@PostConstruct和@PreDestroy

**Spring 注解注入**

在Java代码中使用@Autowired或@Resource注解方式进行装配，@Autowired默认按类型装配, @Resource默认按名称装配, 当找不到与名称匹配的Bean才会按类型匹配. @Autowired注解是按类型装配依赖对象, 默认情况下它要求依赖对象必须存在, 如果允许null值, 可以设置required=false。如果想使用按名称装配, 可以结合@Qualifier注解一起使用

spring2.5为我们引入了组件自动扫描机制, 它可以在类路径下寻找标注了@Component、@Controller、@Service、@Reponsitory注解的类， 并把这些类纳入进spring容器中管理.

@Controller通常用于标注控制层组件(如struts中的action);

@Service通常用于标注业务层组件;

@Repository通常用于标注数据访问组件, 即DAO组件;

@Component泛指组件, 当组件不好归类的时候, 我们可以使用这个注解进行标注;

**注入bean的注解**

@Autowired:Spring提供的注解

@Inject:JSR-330提供的注解

@Resource:JSR-250提供的注解

**Spring 容器**

* ClassPathXmlApplicationContext:从类路径中加载。
* FileSystemXmlApplicationContext:从文件系统加载。
* XmlWebApplicationContext:从web系统中加载。

bean工厂:最简单的容器，提供了基础的依赖注入支持。创建各种类型的Bean.

应用上下文:建立在bean工厂基础之上，提供系统架构服务。ApplicationCotext,spring更加高级的容器。功能强大：

1.提供文本信息解析工具，包括对国际化支持。

2.提供载入文件资源的通用方法，如图片。

3.可以向注册为监听器的bean发送事件

在web应用程序中，需要在web.xml文件中添加下面配置：

<context-param>

<param-name>contextConfigLocation</param-name>

<param-value>classpath\*:spring-mybatis.xml</param-value>

</context-param>

<listener>

<listener-class>

org.springframework.web.context.ContextLoaderListener

</listener-class>

</listener>

**Spring Aware接口**

Spring的依赖注入的最大亮点是所有的Bean对Spring容器没有意识，即可以将Spring容器替换成其他容器。

当需要用到Spring容器本身的功能资源，这是Bean就需要知道Spring容器的存在，才能使用Spring容器提供的资源，这就是所谓的Spring Aware

Spring提供的Aware接口：

BeanNameAware：获得容器中Bean的名称

BeanFactoryAware：获得当前Bean factory

ApplicationContextAware：获得当前的Application context，集成了下面三个服务

MessageSourceAware：获得message source

ApplicationEventPublisherAware：应用事件发布器，可以发布事件

ResourceLoaderAware：获得资源加载器，可以获得外部资源文件

**Spring 多线程**

**@Enable\*注解**

@EnableAspectJAutoProxy：开启对AspectJ自动代理的支持

@EnableAsync：开启异步方法的支持

@EnableScheduling：开启计划任务的支持

@EnableWebMvc：开启web MVC的配置支持

@EnableConfigurationProperties：开启对@ConfigurationProperties注解配置bean的支持

@EnableJpaRepositories：

@EnableTransactionManagement：开启注解式的事务支持

@EnableCaching：开启注解式的缓存支持