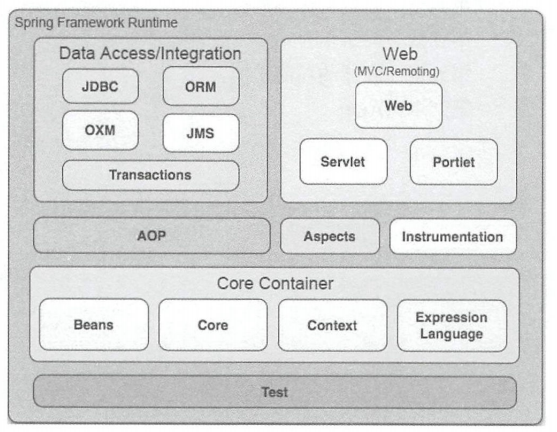
Spring源码

001 Spring的整体架构



1.Core Container（核心容器），包含有Core、Beans、Context、和Expressing Language

Core和Beans模块是框架的基础部分，提供IOC和DI特性。这里的基础概念是BeanFactory

,它提供对Facotry模式的经典实现来消除对程序单例模式的需要，并真正允许你从程序

逻辑中分离出依赖关系和配置。

-Core Core模块主要包含Spring框架基本的核心工具类，Spring的其他组件都要用到

这个包里的类

-Beans Beans模块也是其他所有应用都要用到的，它包含访问配置文件、创建和管理bean

以及进行IOC Inversion OF Control和DI Dependency Injection 操作相关的所有类

-Context 它构建于Core和Beans模块之上，是Spring对外提供的应用程序上下文，它

提供了一种对象访问方法，Context模块继承了Beans模块的特性，为Spring核心提

供了大量的扩展，添加了对国际化、事件传播、资源加载和对Context的透明创建支持。

Context模块同时也支持J2EE的一些特性，例如EJB、JMX和基础的远程处理。

ApplicationContext接口是Context模块的关键。

-Expression Language 模块提供了强大的表达式语言，用于运行时查询和操纵对象。它

支持设置/获取属性的值、属性的分配，方法的调用，访问数组上下文、容器和索引器、  
 逻辑和算术运算符、命名变量以及从IOC容器中根据名称检索对象。它也支持list投

影、选择和一般的list聚合。

2. Data Access/Integration

-JDBC 模块提供了一个JDBC抽象层，它可以消除冗长的JDBC编码和解析数据库提供商

特有的错误代码。这个模块包含了Spring对JDBC数据访问进行封装的所有类。

-ORM 模块为流行的对象-关系映射API，如JPA、JDO、Hibernate、IBatis等，提供了一

个交互层。利用ORM封装包，可以混合使用所有Spring提供的特性进行O/R映射，如

前面提到的简单生命性事务管理。

-OXM 模块提供了一个对Object/XML映射实现的抽象层，Object/XML映射实现包括JAXB、

Castor、XMLBeans、JiBX和XStream。

-JMS JAVA Message Service主要包含了一些制造和消费消息的工具封装。

-Transaction 模块支持编程和声明性事务管理，这些事务类必须实现特定的接口，并且

所有的POJO都实用。

3.WEB模块Web上下文模块建立在应用程序上下文模块之上，为基于web的应用程序提了

上下文。WEB模块还简化了处理大部分请求以及将请求参数绑定到域对象的工作。

-web模块：提供了基础的面向web的集成特性，例如多文件上传、使用servlet linstenerc

初始化IOC容器以及一个面向WEB的应用上下文。它还包含Spring远程支持中web的

相关部分。

-web-servlet模块，该模块包含Spring的model-view-controller（MVC）实现。Spring

的MVC框架是的模型范围内的代码和web form之间能够清楚地分离开来，并与Spring

的其他特性结合在一起。

-web-struts模块：该模块提供了对struts的支持，使得类在Spring应用中能够与一个

典型的struts web层集成在一起。Spring3.0之后已经不再提供对struts的支持。

-web-porlet模块：提供了对porlet环境和web-servlet模块的MVC的实现。

4.AOP AOP模块提供了一个符合AOP联盟标准的面向切面编程的实现，它让你可以定义例如

方法拦截和切点，从而将逻辑代码分开，降低它们之间的耦合性。利用source-level的

元数据功能，还可以将各种信息合并到你的代码中，这点像.NET技术里面的attribute。

Spring AOP模块为基于Spring的应用程序中的对象提供了事务管理服务。通过SpringAOP

不再依赖EJB组件，就能将声明性事务管理集成到应用程序中。

-Aspects模块：提供了对AspectJ的支持。

-Instrumentation模块：提供了class instrumentation支持和classloader实现，使

得可以在特定的应用服务器上使用。

5.Test模块：对JUNIT和TestNG框架的支持。

002 Spring的IOC容器加载Bean的过程

1. 获取对XML文件的验证模式
2. 加载XML文件，并得到对应的Document
3. 根据返回的Document注册Bean的信息

003 XML文件的验证模式

XML文件的验证模式有两种DTD和XSD。Spring用来检验验证模式的办法就是判断是否包

含DOCTYPE，如果包含就是DTD，否则就是XSD，这就是为什么我们的spring-mvc.xml不

包含DOCTYPE的原因。因为要使用的那些节点全部都存放在Spring的XML Schema中。所

以必须要使用XSD验证模式。

* DTD（Document Type Definition）即文档类型定

义，是一种XML约束模式语言，是XML文件的验证机制，同时也是XML文件的一部分。

DTD是一种保证XML文档格式正确的有效方法，可以通过比较XML文档和DTD文件来看文

档是否符合规范，元素和标签是否使用正确。一个DTD文档包含：元素的定义规则，元素  
间关系的定义规则，元素可以使用的属性，可使用的实体或符号规则。要使用DTD验证模

式的时候还需要在XML文件的头部进行声明，就像下面这样：



* XSD XML Schemas Definition。XML Schema描述了XML文档的接口。可以用一个指定

的XML Schema来验证一个某个XML文档。文档设计者可以通过XML Schema指定XML文

档所允许的结构和内容，并可据此检查XML文档是否是有效的。XML Schema本身就是

XML文档，它符合XML文档的语法结构。在使用XML Schema文档对XML实例文档进行

检验时，除了要声明名称空间外

例如：xmlns=<http://www.springframework.org/schema/beans>

还必须要指定该名称空间所对应的XML Schema文档的存储位置。通过schemaLocation

来指定，它包含两个部分，一个是命名空间的URI，另一个部分就是该名称空间所标识

的XSD文件URL

例如：<http://www.springframework.org/schema/beans> <http://www.springframework>

.org/schema/beans/spring-beans.xsd

004 获取Document

Spring也是通过SAX来解析XML文档的。遗留问题SAX，解析过后得到一组Document对

象。

005 解析及注册BeanDefinition

得到Document后，首先是检查是否定义了profile属性，即是否在当前环境需要解析某

一个bean,如果该bean的profile定义的是生产环境，但是现在环境变量配置的是开发

环境，那么就不会去解析该bean了。

006 解析并注册BeanFinition

处理了profile后就可以进行XML的读取了。在Spring的XML配置文件中有两类Bean的

声明，一个是自定义的，如：<bean id=“test”class=“com.shyfay.Test”/>，另一个

是Spring默认的例如：<context:annotation-config />。这两种bean的加载和解析是

有非常大的区别的。那么是怎么知道是自定义的节点还是Spring默认的节点呢，其实是

通过节点的命名空间来确定的，如果命名空间Spring中固定的命名空间一致即：

<http://www.springframework.org/schema/beans>

那么它就是Spring自带的节点，否则就是用户自定义的节点了。

007 BeanDefinition 它是XML配置文件中<bean>标签在IOC容器内部的存在形式。bean拥

有class scope lazy-init等配置属性，BeanDefinition也提供了相应的

beanClass scope lazyInit这些属性与bean对应。BeanDefinition是一个接

口，它有三个实现类RootBeanDefinition、ChildBeanDefinition和

GenericBeanDefinition。三种实现均继承了AbstractBeanDefinition。

008 bean的属性

-scope 指定bean的作用域

-singleton 指定bean是单例的

-abstract 当设置为true时，将该bean仅仅作为模板使用，应用程序上下文不会去预先

初始化它

-lazy-init 当设置为true时，延迟加载，该bean不会在上下文启动时提前被实例化，

而是在应用程序第一次去容器getBean时被实例化，只对singleton的bean

起作用

-autowired 自动装配，所谓自动装配就是Spring IOC容器可以自动装配互相协作的bean

之间的关联关系。

-dependency-check 检查依赖的bean

-depends-on 告知IOC在初始化该bean之前，先初始化depends-on指定的bean

-autowire-candidate 设置为false时，容器在查找自动装配对象时忽略该bean，但是

该bean本身可以使用自动装配其他bean

-primary 告知ioc容器先加载该bean

-init-method 初始化bean是调用的方法

-destroy-method 容器销毁之前掉好用的方法

-factory-method 当调用factory-method所指向的方法时，才开始实例化bean

-factory-bean 调用静态工厂方法来创建该bean

009 bean的子元素meta

<bean id=”myTestBean” class=”com.shyfay.MyTestBean”>

<meta key=”testStr” value=”aaaaa”>

</bean>