# The IoC container

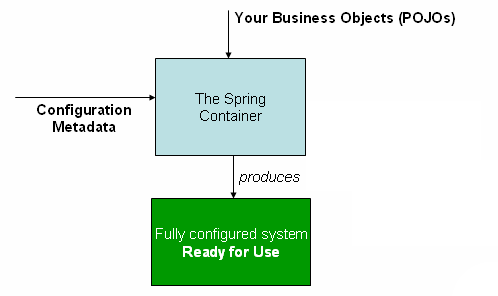
## Introduction to the Spring IoC container and beans(介绍Spring IOC容器和beans)

org.springframework和org.springframework.context是IOC容器的基础的集成包。

[BeanFactory](http://docs.spring.io/spring/docs/current/javadoc-api/org/springframework/beans/factory/BeanFactory.html)接口提供了管理所有对象类型的配置和原理. [ApplicationContext](http://docs.spring.io/spring/docs/current/javadoc-api/org/springframework/context/ApplicationContext.html) 是 BeanFactory.的接口，它很容易集成到Spring AOP、消息订阅、甚至WebApplicationContext。

## Container overview(容器预览)

org.springframework.context.ApplicationContext代表着整个IOC容器，ApplicationContext做为接口，一般的实现主要有[ClassPathXmlApplicationContext](http://docs.spring.io/spring/docs/current/javadoc-api/org/springframework/context/support/ClassPathXmlApplicationContext.html) 或者 [FileSystemXmlApplicationContext](http://docs.spring.io/spring/docs/current/javadoc-api/org/springframework/context/support/FileSystemXmlApplicationContext.html" \t "_top)，下图就显示，在ApplicationContext作用下，你应用的类在捆绑配置的原数据类型后，产生你系那个要的。



### Configuration metadata

这里介绍下采用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"

xsi:schemaLocation="http://www.springframework.org/schema/beans

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

<bean id="..." class="...">

*<!-- collaborators and configuration for this bean go here -->*

</bean>

<bean id="..." class="...">

*<!-- collaborators and configuration for this bean go here -->*

</bean>

*<!-- more bean definitions go here -->*

</beans>

### Instantiating a container

 Spring IoC container非常直接容易，只需告知资源所在路径，

ApplicationContext context =

**new** ClassPathXmlApplicationContext(**new** String[] {"services.xml", "daos.xml"});

<?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">

*<!-- services -->*

<bean id="petStore" class="org.springframework.samples.jpetstore.services.PetStoreServiceImpl">

<property name="accountDao" ref="accountDao"/>

<property name="itemDao" ref="itemDao"/>

*<!-- additional collaborators and configuration for this bean go here -->*

</bean>

*<!-- more bean definitions for services go here -->*

</beans>

为了简化，也可以直接采用多个xml导入

<beans>

<import resource="services.xml"/>

<import resource="resources/messageSource.xml"/>

<import resource="/resources/themeSource.xml"/>

<bean id="bean1" class="..."/>

<bean id="bean2" class="..."/>

</beans>

### Using a container

当你拿到ApplicationContext后，可以通过下面方法T getBean(String name, Class<T> requiredType)，检索到实体Bean类

*// create and configure beans*

ApplicationContext context =

**new** ClassPathXmlApplicationContext(**new** String[] {"services.xml", "daos.xml"});

*// retrieve configured instance*

PetStoreService service = context.getBean("petStore", PetStoreService.**class**);

*// use configured instance*

List<String> userList = service.getUsernameList();

## Bean overview(Bean预览)

### Naming beans

Xml配置时，id必须唯一性

在bean定义的外面取别名

<alias name="fromName" alias="toName"/>

### Instantiating beans

**构建构造函数的bean**

<bean id="exampleBean" class="examples.ExampleBean"/>

<bean name="anotherExample" class="examples.ExampleBeanTwo"/>

**通过工作方法来构建bean**

<bean id="clientService"

class="examples.ClientService"

factory-method="createInstance"/>

**public** **class** ClientService {

**private** **static** ClientService clientService = **new** ClientService();

**private** ClientService() {}

**public** **static** ClientService createInstance() {

**return** clientService;

}

}