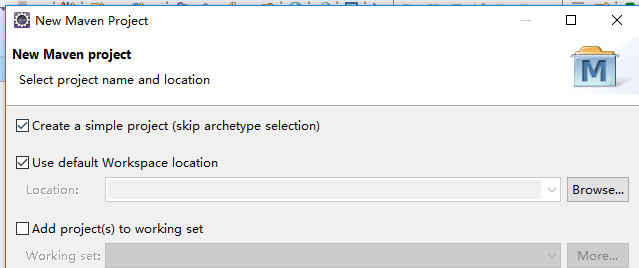
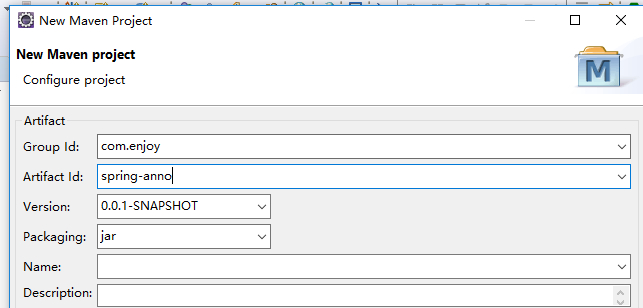
**一 CAP1章节 将你的工程从XML配置到注解**

1. 创建工程
2. 创建Maven工程：
3. 
4. 
5. pom.xml引入spring-context jar和Junit测试用例包

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.0.6.RELEASE</version>

</dependency>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

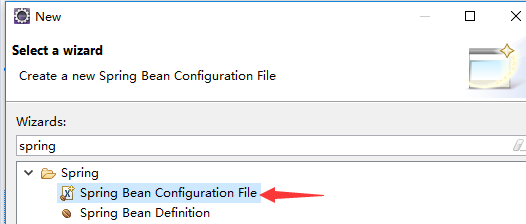
<version>4.12</version>

<scope>test</scope>

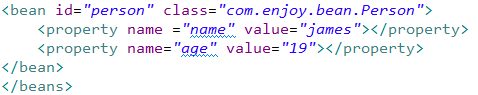
</dependency>

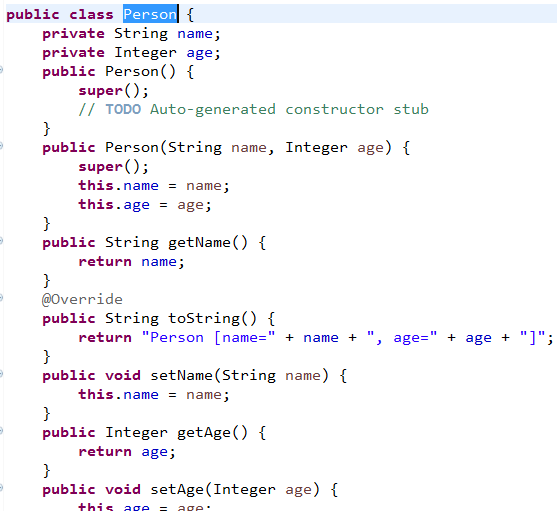
</dependencies>

1. 如果是以前,我们应该建立spring的beanx.xml

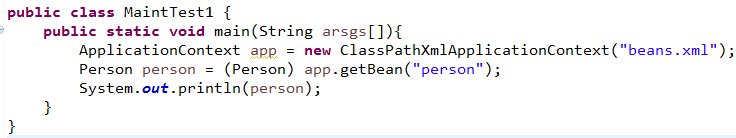


beans.xml内容如下,使用bean标签注册一些组件<新建Person.java>:



3,新建cap1包名，新建Person.java类  


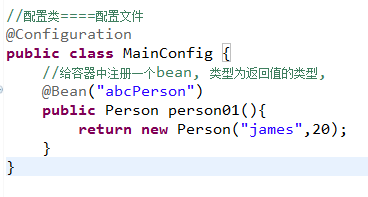
4，新建MainTest1测试类：ClassPathXmlApplicationContext:类路径下的XML



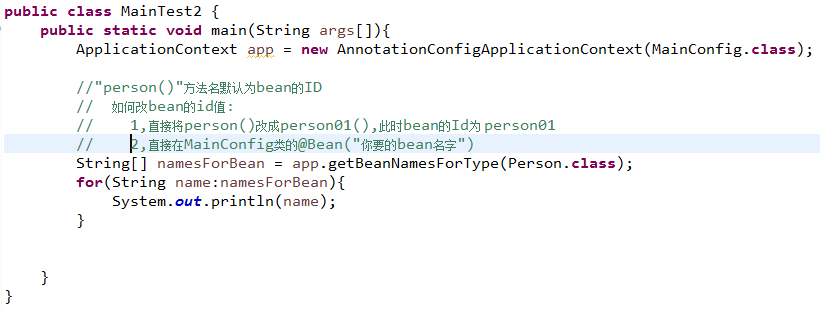
如果我们用注解开发, 很明显是不需要XML的

5，**注解测试：**如何使用注解（去掉配置文件）开发

新建MainConfig类



6，**注解测试：**新建MainTest2注解测试，用来测试//AnnoatationConfigApplicationContext: 注解配置来获取IOC容器



**二 CAP2章节 ComponentScan扫描规则**

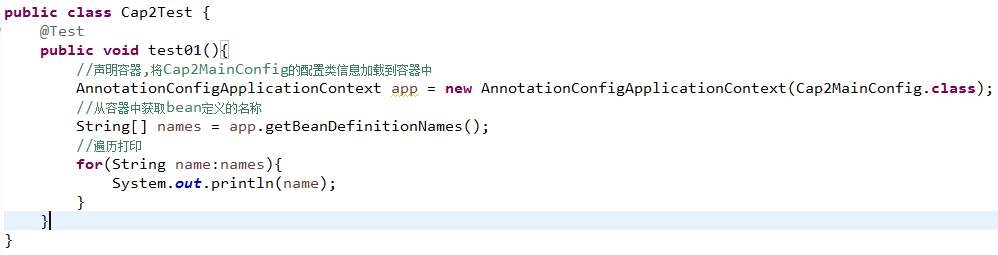
2.1 操作:新建cap2文件夹,新建Cap2MainConfig.java配置类

作用:指定要扫描的包



1,@ComponentScan(value="com.enjoy.cap2")表示扫描此目录下的包

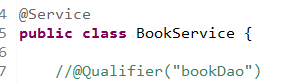
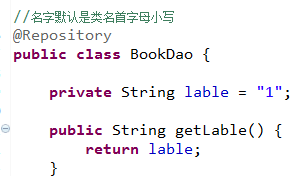
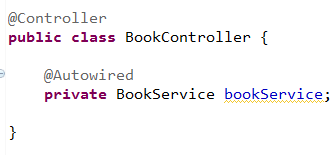
2,建立测试用例方法;



2.2 新建Cap2MainConfig2配置类

作用:定制包扫描时的过滤规则

新建dao, service,controller



在Cap2MainConfig2加入配置: @Filter: 扫描规则

@ComponentScan(value="com.enjoy.cap2",**includeFilters**={ @Filter(type=FilterType.ANNOTATION,classes={Controller.class}), @Filter(type=FilterType.ASSIGNABLE\_TYPE,classes={BookService.class})

},**useDefaultFilters**=false) //默认是true,扫描所有组件，要改成false,使用自定义扫描范围

\*/

//@ComponentScan value:指定要扫描的包

//**excludeFilters** = Filter[] 指定扫描的时候按照什么规则排除那些组件

//**includeFilters** = Filter[] 指定扫描的时候只需要包含哪些组件

//**useDefaultFilters** = false 默认是true,扫描所有组件，要改成false

//－－－－扫描规则如下

//FilterType.ANNOTATION：按照注解

//FilterType.ASSIGNABLE\_TYPE：按照给定的类型；比如按BookService类型

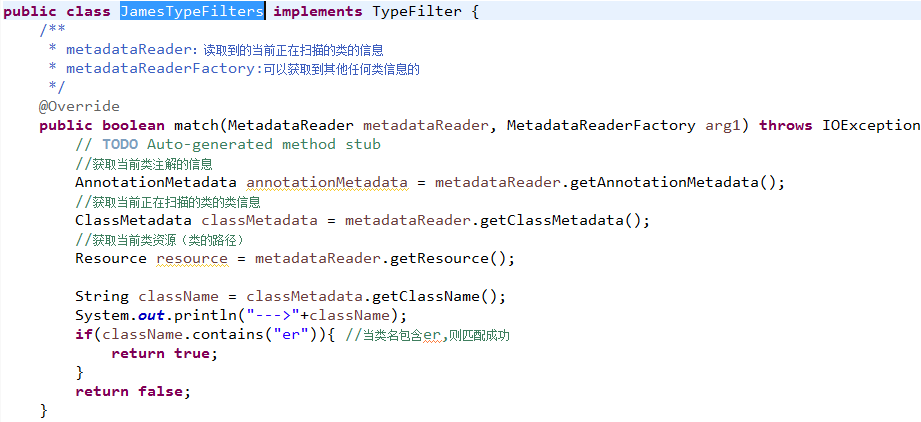
//FilterType.ASPECTJ：使用ASPECTJ表达式

//FilterType.REGEX：使用正则指定

//FilterType.CUSTOM：使用自定义规则，自已写类，实现TypeFilter接口

//FilterType.CUSTOM的例子,常用

先新增自定义过滤规则类:



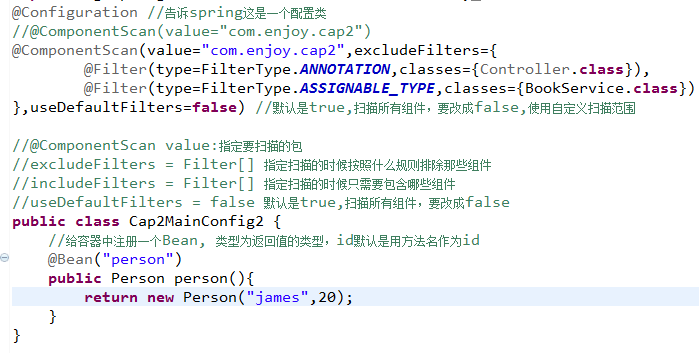
在Cap2MainConfig申明

@ComponentScan(value="com.enjoy.cap2",includeFilters={

@Filter(type=FilterType.***CUSTOM***,classes={JamesTypeFilters.**class**})

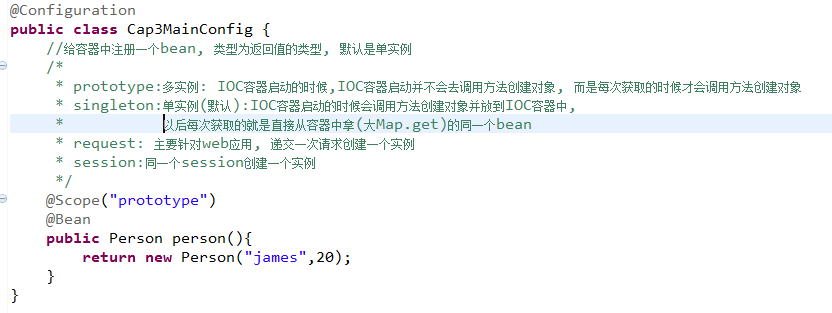
},useDefaultFilters=**false**)

**public** **class** Cap2MainConfig2 {}



**三 CAP3章节 scope扫描规则**

1,新建Cap3MainConfig.java



2,没加@Scope之前, 默认的bean是单实例的. 新建 test01()方法测试如下:



返回true, 证明取到的是同一个person bean,只实例化了一次.

3, 加入@Scope(“prototype”) //多实例

prototype: 多实例：IOC容器启动并不会去调用方法创建对象放在容器中，而是 每次获取的时候才会调用方法创建对象,见test02

singleton: 单实例（默认）：IOC容器启动会调用方法创建对象放到IOC容器中

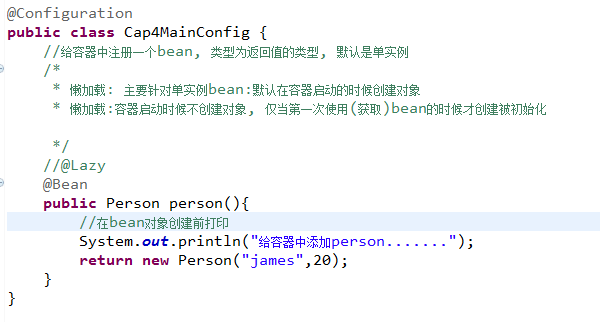
以后每交获取就是直接从容器（理解成从map.get对象）中拿

request: 主要针对WEB应用，同一次请求创建一个实例

session: 同一个session创建一个实例（**后面两个用得不多，了解即可**）

**四 CAP4章节 lazy懒加载**

1,新建Cap4MainConfig.java



2,建立测试用例test01();



当在Cap4MainConfig加入@Lazy时, 只有获取anno.getBean时才会加载到IOC容器中