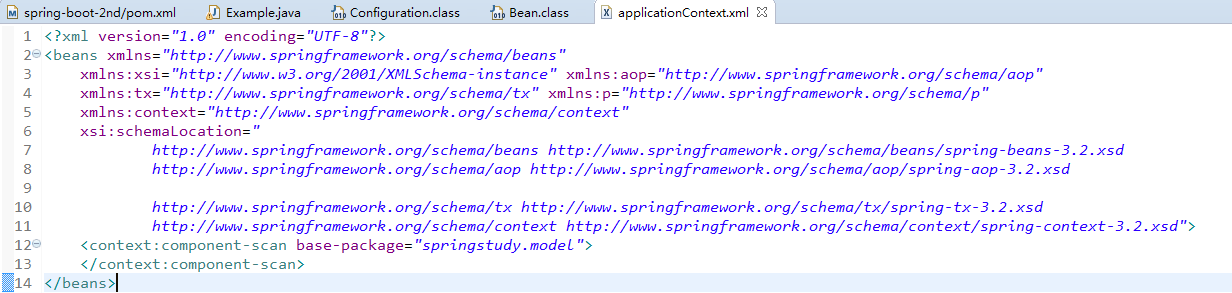
Spring中为了减少xml的配置 ---- 可以声明一个配置类 来对bean进行配置

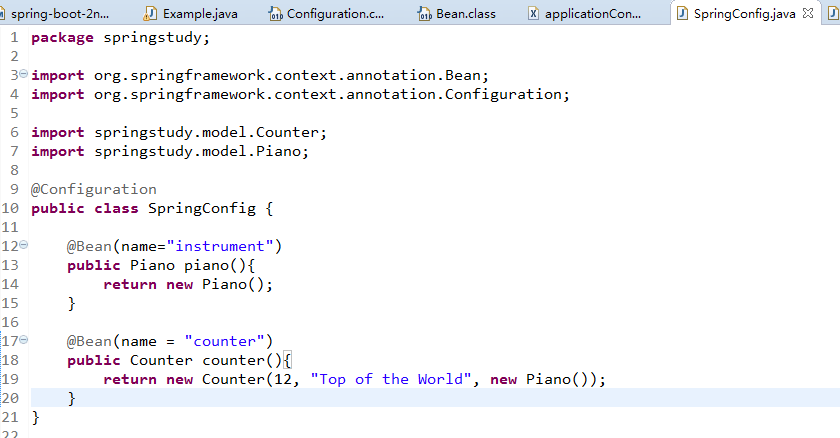
(1). 进行简单的xml配置



定义一个配置管理类

用@Configuration注解这个类 等价于xml中配置beans

@Bean标注的方法等价于 xml中配置的bean



这个就相当于 (大概)

<beans>

<bean id=”instrument” class=”studyspring.model.Piano” />

<bean id=”counter” class=”studyspring.model.Counter”>

<constructor-arg index=”0” value=”12” />

<constructor-arg index=”1” value=”Top of the World” />

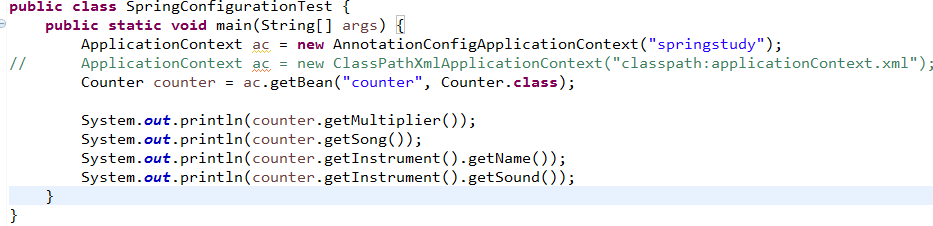
<constructor-arg index=”2” ref=” instrument” />

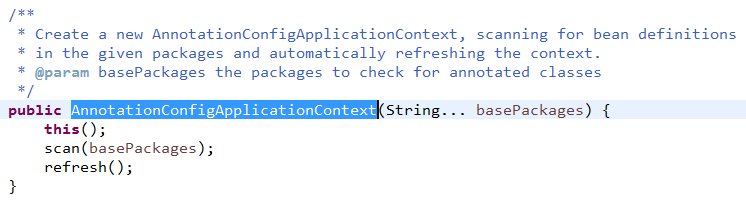
</bean>

</beans>

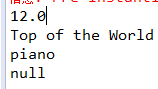
这样 以前在xml中的配置文件就转移到了SpringConfig这个类中了 这个类必须要@Configuration进行标注

最后的测试类





======= 运行结果



======= 以前的用法 使用刚才配置的applicationContext.xml

需要注意的是

要在Bean的每一个java文件上面为这个类打注解

@Component

@Component + xml自动扫描配置 就可以将Bean管理到Spring容器中

------ 联系

现在 把各个Bean文件中@Component汇总到一个SpringConfig（使用了@Configuration进行了配置）-----> 原有的每一个Bean文件中的@Component去掉

这样 就在一个java文件中实现了对应的Bean的配置和管理

使用@Component的缺点 就是只能使用默认的无参构造函数进行实例化对象 如果没有就会抛出异常 ---- 优点就是不需要对每一个Bean进行实例化构建

使用@Configuration + @Bean进行配置 管理很清楚 但是 要繁琐写出每一个Bean的配置情况 同时可以根据任意类型构造方式进行构造

--------------- 能不能结合这两种方式呢？还是不使用xml文件进行配置

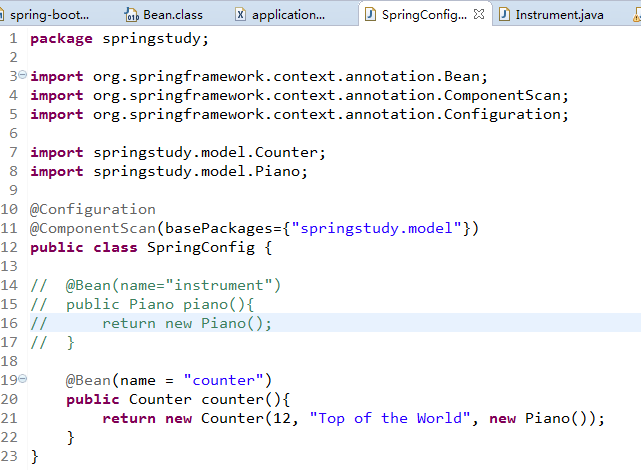
也就是

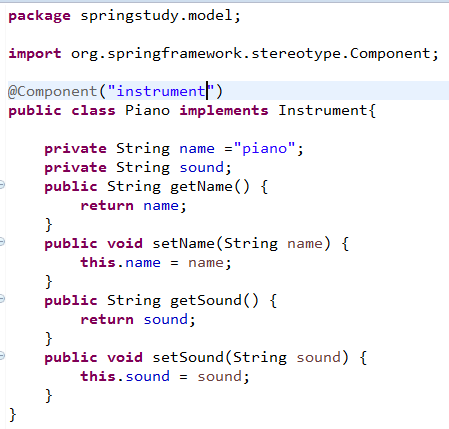
{1}. 对特殊的bean 我们使用@Configuration标记的类进行显示配置 也就是使用@Bean进行标注 ---- 然后对指定的构造方法进行适配

{2}. 对需要使用@Component这种默认方式进行配置的bean 我们在@Configuration上面再打上一个注解@ComponentScan(basePackages=””)标签 这个标签就相当于<context:component-scan base-package>进行的配置

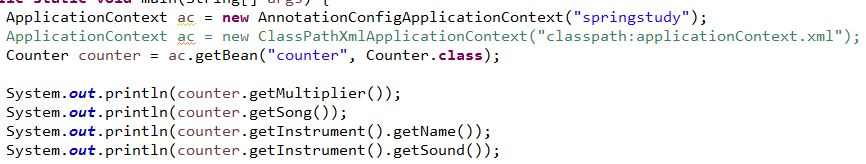
这样就相当于在xml中使用了<context:component-scan base-package>和指定的<bean><constructor-arg>进行配置

这样就都放在一个java文件中进行管理



 就是加上了一个注解@Component 用来被自动扫描

最后在测试中



这样xml中的标签在java类中就有了各种对应的注解进行对应!!!!