3.1基于 Java 的配置选项，可以使你在不用配置 XML 的情况下编写大多数的 Spring。

@Configuration 和 @Bean 注解

@Configuration 的注解类表示这个类可以使用 Spring IoC 容器作为 bean 定义的来源。@Bean 注解告诉 Spring，一个带有 @Bean 的注解方法将返回一个对象，该对象应该被注册为在 Spring 应用程序上下文中的 bean。例子：

```
package com.tutorialspoint;
import org.springframework.context.annotation.*;
@Configuration
public class HelloWorldConfig {
   @Bean
   public HelloWorld helloWorld(){
      return new HelloWorld();
   }
}
```

等同于下面的 XML 配置：

```
<beans>
   <bean id="helloWorld" class="com.tutorialspoint.HelloWorld" />
</beans>
```

带有 @Bean 注解的方法名称作为 bean 的 ID，它创建并返回实际的 bean。使用AnnotationConfigApplicationContext来加载到Spring容器。

```
   ApplicationContext ctx =
   new
AnnotationConfigApplicationContext(HelloWorldConfig.class);
   HelloWorld helloWorld = ctx.getBean(HelloWorld.class);
```

@Import 注解

@import 注解允许从另一个配置类中加载 @Bean 定义。例子：

```
@Configuration
public class ConfigA {
   @Bean
   public A a() {
      return new A();
   }
}
```

可以在另一个 Bean 声明中导入上述 Bean 声明，如下所示：

```
@Configuration
@Import(ConfigA.class)
public class ConfigB {
   @Bean
   public B a() {
      return new A();
   }
}
```

生命周期回调

属性：

```
public class Foo {
   public void init() {
      // initialization logic
   }
   public void cleanup() {
      // destruction logic
   }
}

@Configuration
public class AppConfig {
   @Bean(initMethod = "init", destroyMethod = "cleanup" )
   public Foo foo() {
      return new Foo();
   }
}
```

指定 Bean 的范围：

```
@Configuration
public class AppConfig {
    @Bean
    @Scope("prototype")
    public Foo foo() {
        return new Foo();
    }
}
```