1 注解 Annotation

@Override 告诉编译器这个方法是覆盖父类的方法。

@WebServlet("/test") 表示某个类是一个 Servlet，Web 容器就会识别这个注解，在运行的时候调用它。

@Controller("/test") 表示某个类是一个控制器，告诉 Spring 框架该类是一个控制器。

注解和注释是完全不同的两个东西，看起来有点类似，其实完全不同，注解会影响程序的运行。

注释是给开发人员看的，不会影响程序的编译和运行。注解并不是给开发人员看的，是用于给程序看的，会影响程序的编译和运行，编译器，Tomcat，框架。

1.1 注解的作用范围

自定义开发一个 Web 容器，基本功能是加载 Servlet，需要管理它的生命周期，所以必须先识别程序中的哪些类是 Servlet。

程序启动的时候，扫描所有的类，找出添加了 @WebServlet 注解的类，进行加载。

@WebServlet 是在程序运行的时候起作用的，Java 就把它的作用范围规定为 RUNTIME。

@Override 是给编译器看的，编译器工作的时候识别出包含了 @Override 注解的方法，就去检查它上层父类的相应方法，存在则通过，否则报错。

@Override 是编译时候起作用，Java 就把它的作用范围规定为 SOURCE。

1.2 @Target 指定注解针对的地方

ElementType：

ElementType.TYPE　针对类、接口

ElementType.FIELD　针对成员变量

ElementType.METHOD　针对成员方法

ElementType.PARAMETER　针对方法参数

ElementType.CONSTRUCTOR　针对构造器

ElementType.PACKAGE　针对包

ElementType.ANNOTATION_TYPE　针对注解

1.3 @Retention 指定注解的保留域

RetentionPolicy：

RetentionPolicy.SOURCE　源代码级别，由编译器处理，处理之后就不再保留

RetentionPolicy.CLASS　注解信息保留到类对应的 class 文件中

RetentionPolicy.RUNTIME　由 JVM 读取，运行时使用

1.4 自定义注解

```
package com.southwind.com.southwind.test3;

import java.lang.annotation.ElementType;
import java.lang.annotation.Retention;
import
java.lang.annotation.RetentionPolicy;
import java.lang.annotation.Target;

@Target(ElementType.METHOD)
@Retention(RetentionPolicy.RUNTIME)
public @interface InitMethod {
}
```

```java
package com.southwind.com.southwind.test3;

public class InitDemo {
    @InitMethod
    public void init(){
        System.out.println("init...");
    }

    @InitMethod
    public void test(){
        System.out.println("test...");
    }
}
```

```java
package com.southwind.com.southwind.test3;

import java.lang.reflect.Method;

public class Test {
    public static void main(String[] args)
throws Exception {
        Class clazz =
Class.forName("com.southwind.com.southwind.
test3.InitDemo");
        Method[] methods =
clazz.getMethods();
        if(methods!=null){
            for (Method method : methods) {
```

```
                boolean isInitMethod =
method.isAnnotationPresent(InitMethod.class
);
                if(isInitMethod){

 method.invoke(clazz.getConstructor(null).n
ewInstance(null), null);
                }
            }
        }
    }
}
```