# 基础加强

## 泛型

### 泛型类

具有或多个类型变量的类，称之为泛型类。

例如：

Class A<T>{}

### 赋值

在创建泛型类实例时，需要为其类型变量赋值。例如：

A<String> a = new A<String>();

注意：如果创建实例时，不给类型变量赋值，会产生一个警告！

### 泛型方法

具有一个或多个类型变量的方法，称之为泛型方法！

注意：

Class A<T>{

Public T fun(T t1){}

}

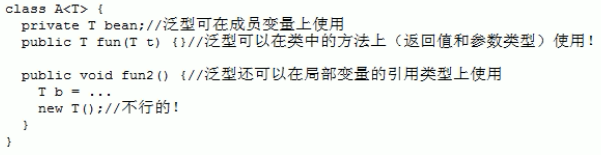
其中 fun() 方法不是泛型方法，它是泛型类中的一个方法！

Public <T> T fun(T t1){} – 是泛型方法。

泛型方法与泛型类没什么关系，泛型方法不一定非要在泛型类中。

### 泛型在类中或方法中的使用

* 泛型类中使用泛型
* 成员类型
* 返回值和参数类型
* 局部变量的引用上
* 例如：



### 泛型的继承和实现

* 例子：

Class A<T> {}

Class AA extends A<String> {}

* 继承泛型类
* 子类不是泛型类

\*- 当给父类传递的类型常量为String时，那么在父类中所有的T 都会被String 替换！

* 子类是泛型类

可以给父类传递类型常量，也可以传递类型变量。

Class AA<E> extends A<String> {}

Class AA<E> extends A<E> {}

都可以！

## 通配符



## 注解

* 作用

替代配置文件 web.xml

Servlet3.0 中就可以不再使用 web.xml 文件，而是所有配置都是用注解！

* 语法

@注解名称

* 使用
* 定义注解类：框架的工作
* 使用注解：我们的工作
* 读取注解（反射）：框架的工作
* 定义注解类

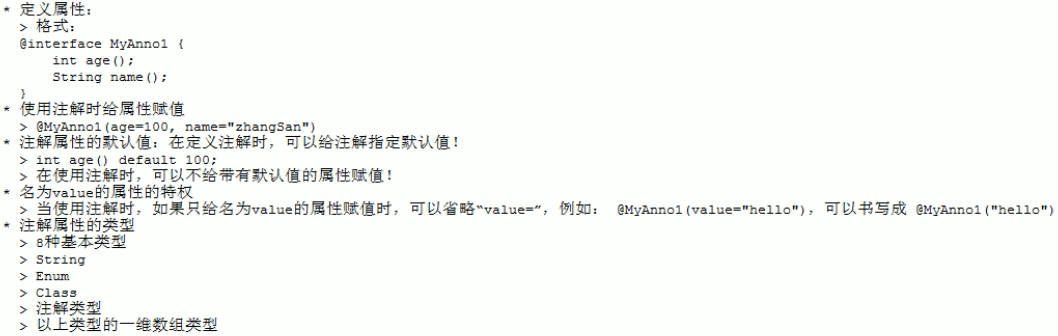
//所有的注解都是Annotation的子类！  
@interface annot1{}

* 使用注解

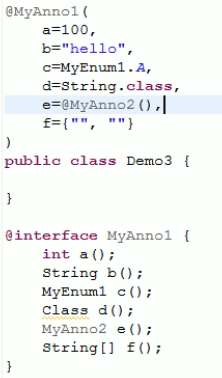
注解可以加在：

类，方法，构造器，参数，局部变量，包上；

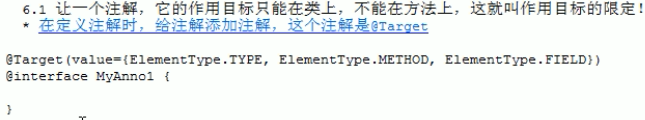
* 注解的属性



* 注解成员参数示例

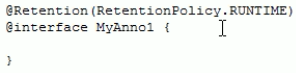


* 注解的作用目标限定



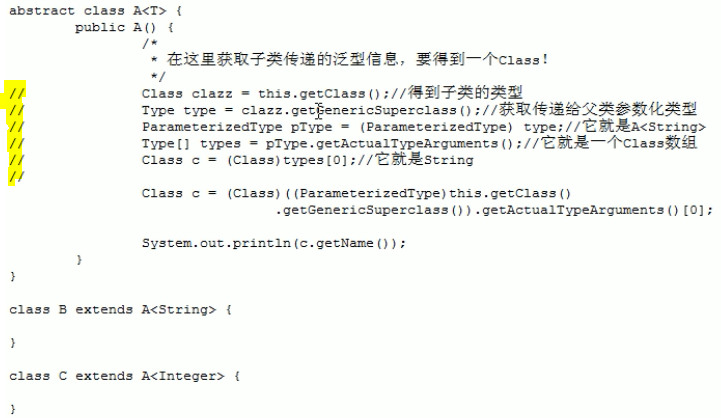
* 注解的保存策略限定



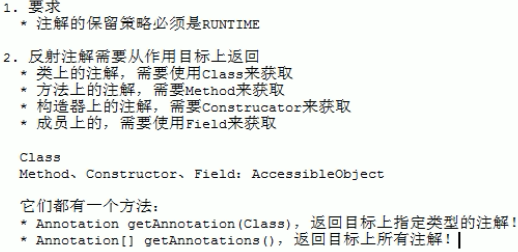


## 反射注解

* 反射泛型信息

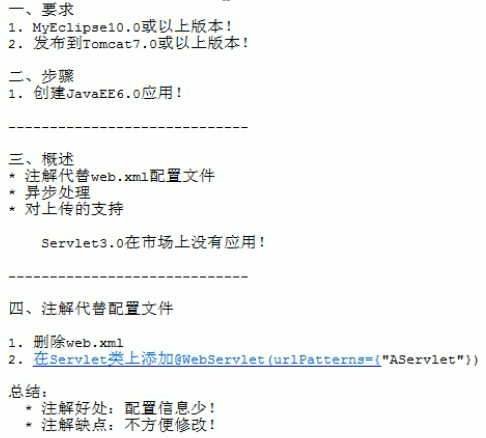


* 反射注解



## Servlet 3.0

### WebServlet注解



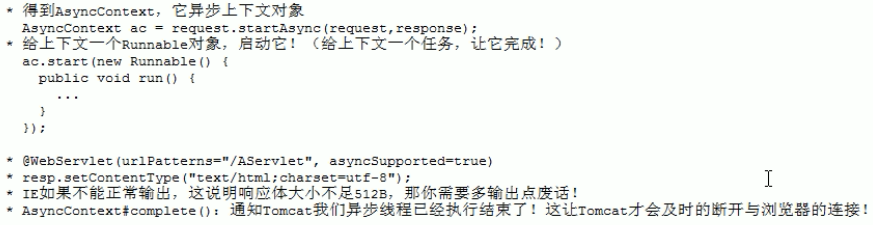
同样可以添加到过滤器、监听器上。

### 异步处理

在服务器没有结束相应之前，浏览器时看不到响应内容的！只有相应结束时，浏览器才能显示结果！

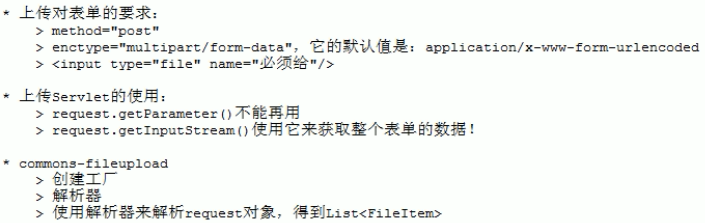
异步处理的作用：在服务器开始相应之后，浏览器就可以看到相应内容，而不用等待响应结束！

* 实现异步的步骤



## 上传支持

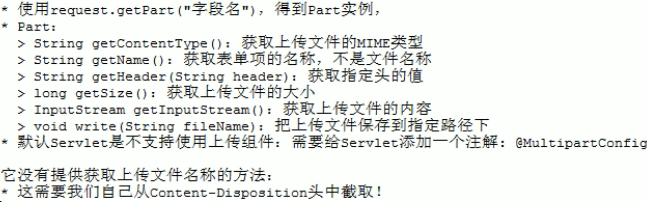
### 之前



### 使用Servlet 3.0

* 使用servlet 3.0 提供的上传组件接口替代 commons-fileupload

### 上传的步骤



## 动态代理

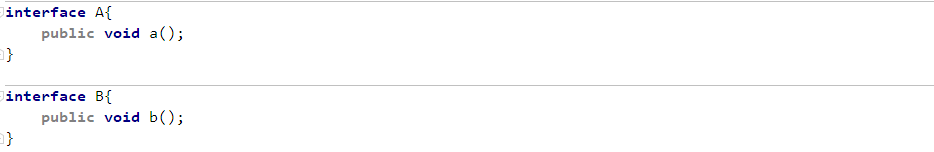
### 作用

动态创建一组指定接口的实现类对象！

（换言之，在运行时，创建实现了指定的一组接口的对象！）

### 实现代码





### InvocationHandler

* 包含的唯一方法

public Object invoke(Object proxy, Method method, Object[] args)

这个方法 仅在 调用代理对象所实现的接口中的 一些 方法时被调用！

* 方法的三个参数
* Object proxy

当前对象，即代理对象，再调用谁的方法！

* Method method

当前被调用的方法（目标方法！）

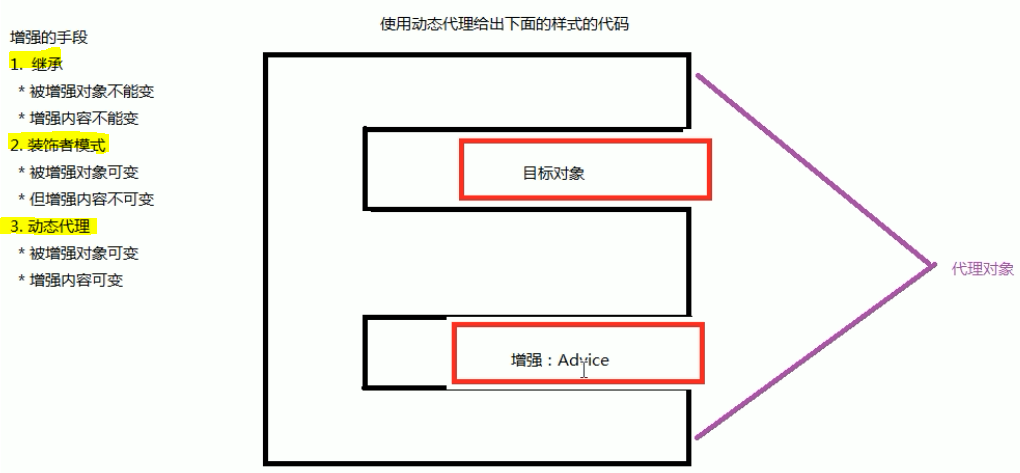
* Object[] args

给目标方法的实参！

* 返回值

为 invoke() 方法的返回值！

### 动态代理的用途



### 使用动态代理增强对象的代码



### 代理工厂的实现

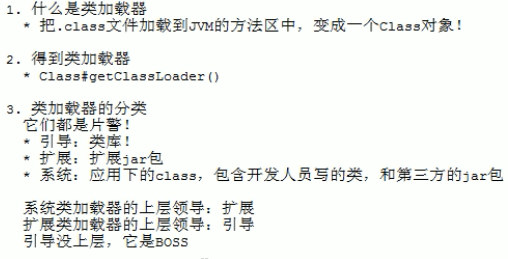
* 目标

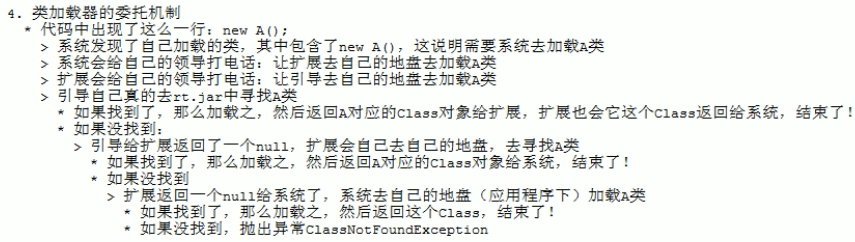
让目标对象和增强都可以更换，而代理工厂只负责装配。

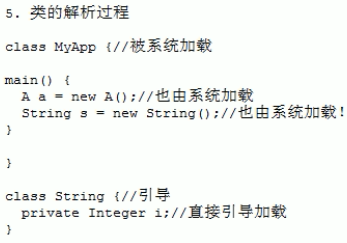
* 代码示例



## 类加载器







### Tomcat 类加载器

