MVC设计模式：
M：model，模型 ：一个功能。用JavaBean来实现
V：view，视图：用于展示、以及与用户交互 使用 html js css jquery
C：Controller，控制器：接受各种请求，将请求跳转到模型处理；处理完毕之后再将处理结果返回给请求处 。 可以用Jsp实现，但一般建议使用Servlet实现控制器。

jsp->Java(Servlet)->JSP 先翻译成一个Java编码再翻译成jsp

Servlet:
Java类必须附和一定的规范：
a.必须继承 javax.servlet.http.HttpServlet
b.重写其中的 doGet()或doPost()方法
doGet()：接受并处理所有get 提交方式的请求
doPost()：接受并处理所有post提交方式的请求

Servlet想要使用，必须配置
Servlet2.5：web.xml
Servlet3.0：@WebServlet

Servlet2.5：web.xml ：
项目的根目录：WebContent、构建路径（例如src）
<a href="WelcomeServlet">所在的jsp是在WebContent目录中，因此发出WelcomeServlet是取请求项目的根目录

Servlet流程：
qingqiu-><url-pattern>(拦截)-》根据<servlet-mapping>中的<servlet-name>取匹配<servlet>中的<servlet-name>再去匹配<servlet-class>

1.纯手工创建Servlet
步骤;
编写一个类继承HttpServlet
重写doGet() doPost()方法
编写web.xml中的servlet映射关系

2.借助eclipse快速生成servlet

直接新建Servlet

Servlet3.0：与2.5的区别

Servlet3.0不需要在web.xml中配置，但需要在servlet类的定义处之上编写注解

@WebServlet("url-pattern的值")

匹配流程：请求地址与@WebServlet中的值进行匹配，如果匹配成功，则说明请求的就是该注解所对应的类

web.xml中的/：代表项目根路径

http://localhost:8888/Servlet30Project/

jsp中的/：服务器根路径

http://localhost:8888/

4.Servlet生命周期：5个阶段

加载

初始化：inti() 该方法会在Servlet被加载并实例化以后执行

服务：service()抽象方法->doGet() doPost()

销毁：destory() Servlet被系统回收时执行

卸载

//修改完web.xml必须重启

inti() ：

a.第一次访问可以执行(只执行一次)

b.可以修改为Tomcat启动时自动执行

i.Servlet2.5： <servlet>

....

<load-on-startup>1</load-on-startup>

</servlet>

其中这个1代表执行顺序

ii.Servlet3.0：

@WebServlet(value="/WelcomeServlet", loadOnStartup = 1)//当只有第一个值得时候value可以省略

service()抽象方法->doGet() doPost():调用几次就执行几次
destory()：关闭tomcat服务时，执行一次。

5.Servlet API：

由两个软件包组成：对应于HTTP协议的软件包，对应于除了HTTP协议以外的其他软件包
即Servlet API可以适用于任何通信协议
我们学习的Servlet，是位于javax.servlet.http包中的类和接口，是基础HTTP协议

6.Servlet继承关系
ServletConfig：接口
ServletContext getServletContext()：获取Servlet上下文对象 application
String getInitParameter(String name)：在当前Servlet范围内，获取名为name的参数值（初始化参数）

a.ServletContext中的常见方法：

getRealPath():获取绝对路径
getContextPath()：获取相对路径
setAttribute()、getAttribute()：getAttribute(key)则是取出作用域中相应的值
-->

String getInitParameter(String name)：在当前web容器范围内，获取名为name的参数值（初始化参数）

Servlet3.0方式给当前Servlet设置初始值：

@WebServlet(value="/WelcomeServlet", loadOnStartup = 1 ,initParams = {@WebInitParam(name="servletparaname30",value = "servletparavalue30") })
注意此注解只隶属于某一个具体的Servlet，因此无法为整个web容器设置初始化参数，（如果用3.0方式和2.5是一样的）

HttpServletRequest（）中得方法和request一致
HttpServletResponse（）中得方法和Response一致

Servlet在使用层面：

Eclipse中在src创建一个Servlet，然后重写doGet()和doPost()就可以了（doGet()和doPost()只需要编写一个）

1.三层架构

与MVC设计模式的目标一致，都是为了 了解耦合，提高代码复用；

区别，二者对项目理解的角度不同

2.三层组成：

表示层:View（视图层）

-前台：对应于MVC中的view：用于和用户交互，界面显示

jsp,js,html,css;

代码位置：webcontent

-后台：对应于MVC中的Controller，用于控制跳转、调用业务逻辑层

Servlet位于包中

业务逻辑层:Service层

-接收表示层的请求、调用

-组装数据访问层，带逻辑性的操作（增删改查，删：查+删）

一般位于xxx.service包中

数据访问层：Dao层

-直接访问数据库的操作，原子性的操作（增删改查）

在Dao包

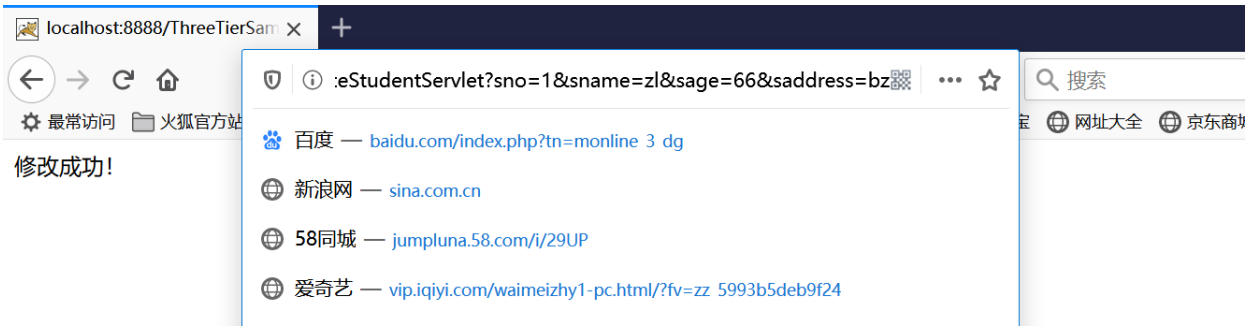
3.三层间的关系

上层将请求传递给下层，下层处理后返回给上层

上层依赖下层　　就是持有成员变量，有A的前提是先有B

Servlet：一个Servlet对应于一个功能，因此，如果有增删改查4个功能就写4个Servlet

如果没有写jsp可以通过这种方式传参



7.三层优化

1.加入接口

建议面向接口开发：先接口-再实现类

--service、dao加入接口

--接口与实现类的命名规范

接口：interface　　起名　　I实体类Service　　IStudentService

IStudentDao

实现类：implements　　起名　　实体类ServiceImpl　　StudentServiceImpl

StudentDaoImpl

接口：I实体类层所在的包名

接口所在的包：xxx.service　　xxx.dao

实现类：实体类层所在包名Impl

实现类所在的包：xxx.service.impl　　xx.dao.impl

以后使用接口/实现类时，推荐写法

接口 x = new 实现类;

IStudentDao studentDao = new StudentDaoImpl();

2.DBUtil　　通用的数据库帮助类，可以简化Dao层的代码量

帮助类一般建议写在xxx.util包

nike m2k tekno