1安装环境：tomcat服务器的安装（tomcat能识别jsp标签而阿帕奇服务器只能识别html）

所以在javaweb开发中需要这个服务器

安装是要注意在环境变量中设置CATALINA\_HOME是服务器的目录

2 tomcat的目录结构其中的webapp是存放web应用文件存放在此目录中

Work目录是将jsp生产的sevlert放到这个文件夹中。

3 第一个web应用程序：

先要在webapps中新建项目的名字（如hello）

然后在这个目录下也是有规则的：要有主页index.jsp还要有文件加WEB-INF

4关于web-inf目录详解

其只能是服务器访问。也就是说客户端是不能访问的（一般情况下客户端是可以访问项目目录下的文件的）

其中还有web.xml文件用来项目部署（比如可以修改和配置直接访问项目名的时候打开的是主页名字默认是index.jsp），还要有classes文件夹用来存java的字节码，还要有lib存放需要的jar包

5在javaee（eclipse中集成tomcat服务器但这样做有什么好处呢？）

Jsp语法：

Jsp（java server pages）其可以包含html标签

语法：（1）page指令：<% @ 属性=“属性值”%>

其中属性有：language（默认java），import（可以导入java包），contentType（字符编码）

（2）jsp脚本：在jsp页面执行的java代码<%java代码%>

（3）jsp声明：在jsp页面定义变量或方法<%!java代码%>

（4）jsp表达式：在jsp页面中执行的表达式<%=表达式%>表达式不要以封号结束；

（5）jsp页面的生命周期： 当客户端第一次访问index.jsp的时候jsp会转化成sevlert（实质也是java类）和sevlert字节码文件，并会执行sevlert中的方法jspinit（），然后用jspservers（）方法来处理用户的请求（多线程实现）

就我目前的理解是jsp是sevlert的一个抽象。便于动态界面的书写。不然还要用javax.sevlert类中的一大堆的request，response啊什么的来做

（总的来说jsp就是可以把java代码插入到html中便于输出动态页面，必须要理解sevlert类。。。不然又要走到学flask的老路了。。。）

Jsp中的内置对象：out，request，response，session，application，page，pagecontext，exception，config总共9个对象，常用的是前5个。

这些内置对象是由web容器创建的（既然是对象就可以直接使用不用new 一个类了）？

问题什么是web容器？

1. out对象：是jspwrite类的实例对象

作用：向客户端输出内容

其中常用的方法：

Void println（）：向客户端打印字符串。Void clean（）：清除缓冲区的内容void cleanbuffer（）。Void flush（）将缓存区的内容输出到客户端

1. request对象：是类HttpSevlertRequest的对象

客户端的请求会封装在里面

常用的方法：String getParameter(String name)

String[] getParameterValues(String name)

Void setCharacterEncoding(“utf-8”)请求内容的编码（但这种方法只能解决post提交表单的信息乱码问题）

对于从url中获得的信息乱码可以通过配置服务器中的server.xml文件的标签<connector>(就是修改端口好的标签)加属性URLEncoding=“utf-8”

1. response对象：是类httpservletrequest的对象：包含了响应客户请求的相关信息（很少使用）
2. session对象：表示客户端与服务器的一次对话。（用户在这个网站（多个页面）上花的一段时间）它保存在服务器的内存当中（不同的用户有不同的session）

session对象的产生：在第一个jsp页面被加载的时候自动创建完成回话管理。

而且在客户端向服务器发送请求的时候会把这个sessionId发到服务器

生命周期：从客户打开浏览器连接到服务器到关闭浏览器断开与服务器的连接；

服务器通过session对象来判断客户在切换页面的时候是不是同一个人。

Session对象的销毁：可以直接通过他的方法来销毁，也可以设置一个时间在时间结束后销毁，或者当所有页面关闭的时候若在另外打开一个页面的时候他会创建一个新的session对象（老的session对象并没有销毁）；也可以在web.xml中设置销毁时间（《session config》）。

它是httpsession的对象

1. application对象：实现了用户间的数据共享，可以存放全局变量。

生命周期：开始于服务器启动结束于服务器关闭。

它是servletcontext的对象。

1. page对象：指向jsp页面本身，其是object类的实例。
2. pagecontext对象：提供了对jsp页面所有对象及名字空间的访问（如他可以访问session的属性，application的属性等等）
3. config对象是在servlet初始化的时候jsp向他传递信息（初始化参数）用的。
4. exception对象在页面产生异常的时候产生

在做登录系统遇到的问题：页面重新定向：有两种方式

1:请求转发：服务器行为requset.getrequestdispatch().forword()对于浏览器是未知的；是一次请求转发后请求对象会保存（比如前一次的请求中包含了表单信息，在下转发的页面也是可以访问的），地址栏的url地址不变

2：请求重定向：客户端行为Response.sendredirect();本质上是两次请求，前一次的请求对象不会保存，然后服务器会告诉你去请求别的url地址，url会变

Javabean的使用：

他的实质是符合特定规范的java类

作用是解决代码的重复编写，减少代码的冗余，功能区分明确，提高代码的可维护性

Javabean的设计原则：

首先要是公有类，属性必须是私有的（通过set，get方法来访问），无参的构造方法。

他存放在web工程的src文件夹下新建包类。

Javabean的使用

1可以在page指令中import这个类

2 jsp动作元素

1. 在jsp页面实例化或指定范围内使用javabean：<jsp:useBeans id=”唯一标识符”（对象名） class=”类名” scope=“作用范围”/>
2. 给已经实例化的对象的属性赋值，共有4种形式。

First：<jsp:setProperty name=”javabean对象的名字” property=”\*”/>(其会自动匹配有由客户端提交上来的表单，如果提交的表单的信息的名字和javabean类中定义的私有化field属性一样，则自动调用这个对象的set（）函数给相关的对象之赋值)

如果不是全部匹配就在property中指定javabean的属性名字。

Second：如果不想自动的由表单赋值，也可以手动的给javabean对象属性赋值<jsp:setProperty name=”javabean对象的名字” property=”javabean属性名” value=““/>

Third：<jsp:setProperty name=”javabean对象的名字” property=”\*” param=“request对象中的参数名“/>

1. 获取javabean对象中属性的值(是直接显示的)<jsp:getProperty name=”javabean对象的名字” property=”\*”/>

Javabean的四大作用范围（就是在前面创建javabean对象中useBeans的scope）

1page（在当前页面有效）

2request（可以通过request。Getattribute来获取javabean对象）

3session

4application（全局的）

Web开发的模型：

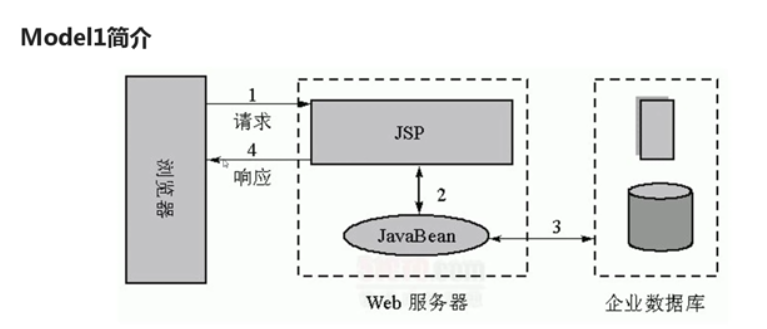
开发方式的历史：

之前的开发状态是全部是jsp页面

这样做的弊端是：在界面层有大量的业务逻辑的代码和数据访问层的代码。可扩展性维护性差

然后是javabean的出现可以把业务逻辑的代码放在javabean里，然后数据可以封装在javabean里

Model1：jsp+javabean由javabean（业务逻辑层）来和数据库打交道



Jsp的状态管理：

http的无状态性：通过http协议发送的信息服务器并不能识别客户端。（服务器不会记得你）

保存用户状态的两大方式：

Session（服务器端）

Cookie（客户端）：服务器保存在客户端的一系列文本信息。

典型的应用：判断注册用户是否已经登录网站。购物车的处理

作用：对特定对象的追踪，保存用户网页浏览的记录，简化登录

但是容易泄露用户信息。

在jsp中使用cookie：

创建cookie对象cookie newcookie=new cookie（String ，object）；

写入cookie对象：response。Addcookie（）；

读取cookie对象：cookie【】 cookies=request。Getcookie（）；

Cookie的方法：setmaxage（）；设置有限期，setvalue（）；对cookie赋值。

但是在cookie中无法保存中文的解决：java.net.URLEcoding.encode();来编码成utf-8；

Session和cookie的区别：

Session保存的是object类型cookie保存的是string

Session随着会话结束销毁，cookie会长期保存在客户端。

Session保存重要信息。Cookie保存不重要的信息。

在使用cookie时遇到的问题：

1 String的是否相对一定要通过equels方法

2 html里的checkbox里在不勾选的是够是什么？返回null？

3在修改了cookie的期限的时候一定要把cookie对象在保存到response里。

Jsp指令

Include指令<%@ include file=”url”%>

使用include在当前页面包含其他的页面的内容。

Include动作<jsp:include page=”url” flush=”boolean”（是否用缓存？）/>

效果和指令一样

但是还是有很多区别的：

发生作用的时间：指令在页面转换期间。 请求期间

包含内容：文件的源代码 页面的输出

转换成的servlet ：主页面和包含的页面在一个servlet中 独立

编译时间：较慢 较快

执行时间 ：较快 较慢

Forward指令：语法<jsp:forward page=”url”/>

等同于request.getRquestDispatcher(‘url”).forward();

页面的跳转。

Param动作：<jsp:param name=”参数名” value=”参数值”>

设置添加参数，也可以从request对象提取。

也可以修改原有的参数。

与forward一起使用作为其子标签

Servlet基础；

servlet容器：tomcat容器分为4个等级，servlet容器管理context容器，一个context对应一个web工程

编写servlet的过程：

1继承HttpServlet

2重写doGet（）doPost（）方法

3在web.xml中注册servlet



关于编写自己定意的servlet，doget（）是处理get请求，dopost处理post请求

关于注册到webxml中，

<servlet>

<servlet-name> </servlet-name>

<servlet-class> 包名.类名 </servlet-class>

</servlet>

<servlet-mapping>（映射）

<servlet-name> </servlet-name>

<url-pattern> / 包名/类名 </url-pattern>(相当于jsp页面的url路径)

</servlet-mapping>

Servlet是执行流程：

在客户端点击超链接href=servlet/helloServlet，在服务器就会在web.xml下查看对应的servlet-mapping找到这个路径对应的servlet名字，然后在标签servlet中找到这个servlet名字对应的类名字，然后根据是get或post执行对应的方法。

Servlet的生命周期：1开始是执行构造方法创建对象（当用户请求的servlet不存在时）

2调用init（）方法（servlet配置）

3响应客户端请求阶段，调用service（）方法,由其根据客户端的请求方式来选择执行doGet还是doPost（）

4在服务器关闭是调用destroy

在什么时候servlet容器会装载servlet

1. 在用户请求这个servlet而这个servlet并没有实例化的时候
2. 可以在servlet容器启动时自动装载，web.xml中配置servlet标签中加入<loadon-startup>1</loadon-startup>(数字越小级别越高)
3. 在servlet被更新后。

Jsp中的9大内置对象和servlet中对应是是

关于servlet中的路径问题：

1在超链接和表单中的要跳转的路径上，可以用相对路径：

Servlet/helloservlet.不要在前面加“/“表示服务器根目录（webcontent）。

也可以用绝对路径：path=request.getcontextpath();path+ /Servlet/helloservlet.path是项目的根目录（与服务器的根目录不同）

2在配置文档中的路径web.xml。在urlpattern要加“/”表示项目的根目录

3 在servlet类中的路径问题（跳转到别的页面）：这里的路径是servlet

注意：这里的webcontent文件夹下的jsp页面默认的路径是在根目录下的（工程目录，不是在webcontent）所以在servlet类中路径跳转可以直接用相对路径../jsp页面

在把jsp上做的登录系统转到servlet上的时候怎么都加载不了servlet类。但是发现第二天就变好了（神奇的是什么都没改。。。）

好转换成servlet测试成功。、。。

开启下一篇章。

Servlet高级内容：

1获取初始化参数

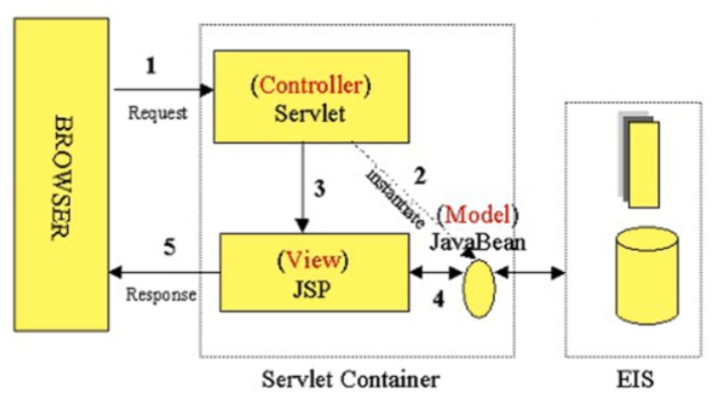
可以在xml文档中设置初始化参数，然后servlet通过接口servletconfig提供的方法getInitParameter（）来获取这些参数，一般在init（）中获取参数。

2mvc设计模式

Model（模型层）由javabean来实现和数据库打交道

Controller（控制层）是由servlet来实现（对用户提交的信息进行分析处理）用户请求的是这一层。

View（视图层）是由jsp页面来实现，来给用户以响应。



3model2简介

就是mvc

Web监听器（用的时候在学）

定义：servlet规范中的特殊类，用于监听servletcontext,httpsession和servletrequest等对象的创建销毁和改变

用途：

1统计在线人数

2系统启动时加载初始化信息

3统计网站访问量

4和spring结合

使用监听器：

1创建实现监听器接口的类（实现servletcontextlistener是对上下文application监听）

2在xml中注册

<listener>

<listener-class></listener-class>

</listener>

Web过滤器

定义：它可以截取客户端的请求和响应信息，并对这些数据过滤。



他的生命周期：在web服务器启动是就加载了

1实例化web.xml

2初始化调用init（）方法：这个方法可以读取xml中的过滤器的参数。

3过滤执行doFilter（）：当用户请求和过滤器关联的url时，web容器将先调用dofilter，然后就可以把请求转发到登录页面

4销毁在web容器关闭的时候调用destroy（）

先要新建一个filter类实现了filter接口

然后要进行配置在xml中

<filter>

<filter-name></filter-name>

<filter-class><filter-class>

<init-param>(初始化参数可以是0堆或多对)

</init-param>

</filter>

<filter-mapping>

<filter-name></filter-name>

<url-pattern>要过滤的url（/\*是过滤所有url）<url-pattern>

<dispatcher>配置什么类型的请求（REQUEST|INCLUDE|FORWARD|ERROR）

默认是request

</dispatcher>

</filter-mapping>

***过滤器链***

在有多个filter的时候如果这多个filter中配置的要拦截的url都不同。没有什么问题

当他们有交集的时候会按照顺序filter1到filter2到filter3然后到web资源

按照xml中定义的先后顺序（过滤器链）

Chain.dofilter（）是到下一个链没有就会到达用户请求的页面

Chain.dofilter()前面的代码是在请求的过程中执行，后面的代码是在服务器响应后执行

流程：

（一个方法总归是要执行完的，所以filter1的过滤代码遇到Chain.dofilter（）转到filter2的过滤代码遇到Chain.dofilter（）访问用户请求的页面，然后执行filter2中没完成的方法，filter2中的dofilter执行完了之后执行完filter1的）

好像不能直接在dofilter方法中直接重定向？？

重定向会导致死循环？sendredirect

过滤器的分类：

Request 用户直接访问页面是web容器直接调用

Forward转发目标资源通过requestdispatch的forward访问时，过滤器被调用

Include目标资源通过responsedispatch的include方法时调用

Error目标资源通过声明式异常处理机制时调用，

过滤器的应用：

对用户请求进行同一认证。

编码转换