7~9天课程目录

第七天（tomcat）

> 在小型的应用系统或者有特殊需要的系统中，可以使用一个 免费的Web服务器：Tomcat，该服务器支持全部SP以及Servlet规范

tomcat安装

> tomcat安装包有三种形式：
>
> > .zip .tar.gz .exe （安装文件）

tomcat启动

> 在[tomcat]/bin目录下双击startup.bat文件
>
> 启动时的问题：
>
> > tomcat窗口一闪而过：
> >
> > > 原因一：JAVA_HOME环境变量配置错误
> >
> > > 端口号8080被占用：
> > >
> > > 通过netstat -ano查看端口号及进程号

修改tomcat端口号

> [tomcat]/conf/server.xml文件中第71行，将8080修改为80.这是将tomcat服务器启动时占用的端口改为80端口。
>
> 80端口是一个缺省端口(在地址栏中可以不用书写)

Tomcat的目录结构

> bin --- 存放tomcat启动和关闭脚本文件的目录 conf --- 存放tomcat配置文件的目录 lib --- 存储tomcat在启动和运行过程中的jar包。 logs --- 存储tomcat日志信息的目录 temp --- 存储tomcat临时文件的目录 webapps --- 存储tomcat中的web应用的目录，这些应用可以被外界访问。 work --- 存储tomcat运行时产生的文件的目录

虚拟主机概念

> 一个网站就是一个虚拟主机
>
> 服务器也称之为主机

web应用概念

> 一个功能创建一个web应用目录

web应用的映射

第一种方式：

> 在conf/server.xml中添加
>
> > `<Context path="虚拟路径" docBase="真实路径"/>`

第二种方式：

> 在conf/Catalina/localhost中创建一个以虚拟路径为文件名称的.xml文件，在文件中添加标签。
>
> > `<?xml version='1.0' encoding='utf-8'?>`
> >
> > `<Context docBase="真实路径"/>`

第三种方式：

> 将webapps中的web应用名称改为ROOT，即可作为缺省的web应用使用。

web.xml文件

web.xml功能：

> 配置过滤器 配置监听器 配置servlet映射 配置缺省主页。全站首页。(conf/web.xml配置缺省首页)

web.xml应用：配置缺省主页

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<web-app xmlns="http://java.sun.com/xml/ns/javaee"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
 http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
 version="3.0">

 <welcome-file-list>
 <welcome-file>1.html</welcome-file>
 </welcome-file-list>

</web-app>
```

虚拟主机的配置(server.xml)

修改server.xml：

```
<Host name="www.baidu.com" appBase="baidu">
</Host>
```

修改hosts文件

> 修改文件的路径： C:\Windows\System32\drivers\etc\hosts

打war包

> 在某一个web应用的目录下访问cmd窗口，在窗口中书写：
>
> > `jar -cvf ROOT.war *`

> 进入到某一个web应用的目录中，将所有文件添加到一个.zip的压缩包中，并修改后缀为.war

HTTP协议（详见笔记）

HTTP/1.0

> 在浏览器和服务器建立连接之后，浏览器发出一次请求，服务器作出一次响应，连接就会断开

HTTP/1.1

> 在浏览器和服务器建立连接之后，浏览器可以依次发出多次请求，服务器会根据各个请求依次作出响应。在服务器等待一段时间之后，如果没有更多的请求到达，则会将连接断开。

HTTP请求

> 客户端连上服务器后，向服务器请求某个web资源，称之为客户端向服务器发送了一个HTTP请求

一个完整的HTTP请求包括如下内容：

> 一个请求行、若干请求头、一个空行、以及实体内容。

HTTP请求的细节——请求行

> 请求行中的GET称之为请求方式，请求方式有：
>
> > POST、GET、HEAD、OPTIONS、DELETE、TRACE、PUT
> >
> > 常用的有： GET、 POST

HTTP响应

> 一个HTTP响应代表服务器向客户端回送的数据，它包括：
>
> > 一个状态行、若干响应头(消息头)、一个空行、以及实体内容 。

第八天（servlet）

> servlet技术是一门由sun公司提供的动态资源开发的技术

```
package cn.tedu.servlet;
import java.io.*;
import javax.servlet.*;
public class FirstServlet extends GenericServlet{
    public void service(ServletRequest req, ServletResponse res) throws
ServletException, IOException{

            res.getWriter().write("first Servlet!!!");
    }
}
```

配置servlet映射

```
<servlet>
 <servlet-name>FirstServlet</servlet-name>
 <servlet-class>cn.tedu.servlet.FirstServlet</servlet-class>
 </servlet>

 <servlet-mapping>
 <servlet-name>FirstServlet</servlet-name>
 <url-pattern>/servlet/FirstServlet</url-pattern>
 </servlet-mapping>
```

Myeclipse开发Servlet(配置见笔记)

Servlet继承关系

> Servlet --- 基础servlet接口
>
> > GenericServlet 通用servlet。(抽象类)，重写service()
> >
> > > HttpServlet 是和HTTP协议相关的servlet。HttpServlet其中包含和HTTP协议相关的api操作，更善于HTTP相关开发

Servlet运行过程(详见图片)

request和response对象是何时产生的?

> 在servlet对象创建之后，request和response对象产生

HttpServletRequest和HttpServletResponse都是接口，如何创建request和response对象?

> 两个接口并不能创建对象，而由各自接口的子实现类创建的对象，这两个子实现类分别为HttpServletRequestWrapper、HttpServletResponseWrapper

问题：为什么静态资源也可以加载?

> 分析：在地址栏中输入的index.html也会作为虚拟路径使用，它和servlet运行过程一致，可以在web.xml中没有与index.html匹配的路径， 所以这个虚拟路径会交给DefaultServlet处理，这个servlet称之为缺省Servlet。

修改Servlet模板(参考课前资料)

Servlet细节

Servlet细节--标签组成

`<servlet>`:其中servlet称之为注册servlet标签

`<servlet-mapping>`:servlet-mapping称之为servlet映射标签

> 如果有多个路径要映射在同一个servlet身上，那么需要重复书写映射servlet标签。修改其中的url-parttern标签。

servlet细节--通配映射

```
<servlet-mapping>
    <servlet-name>AnyName </servlet-name>
    <url-pattern> *.do</url-pattern>
</servlet-mapping>
```

```
<servlet-mapping>
    <servlet-name>AnyName </servlet-name>
    <url-pattern> /action/*</url-pattern>
</servlet-mapping>
```

> Servlet映射路径与URL匹配程度越高，越优先使用
>
> > *.do （永远匹配级最低）

`<load-on-startup>`

> 若标签中的数字为0或者大于0，那么WEB应用程序在启动时就会装载并创建Servlet的实例对象、以及调用Servlet实例对象的init()方法。

> 正数的值越小，启动该servlet的优先级越高。

Servlet的一些细节(7)—线程安全

解决线程安全问题的方式：

> 尽量少的使用全局变量，多使用局部变量。

> 合理加锁。利用锁机制，锁住关键部分代码。因为锁会降低程序的执行效率，所以不能锁住过多的代码。

servlet细节--生命周期运行过程

> 浏览器发出请求

> 将域名解析为ip地址，访问对应ip地址的服务器，再根据端口号确定访问的是tomcat服务器

> 再根据请求头host确定访问哪一台虚拟主机。

> 再根据请求行确定请求的web应用虚拟路径。

> 再根据请求行确定请求的哪一个web资源虚拟路径

> 再根据web.xml文件与虚拟路径相比较，如果有对应的servlet映射路径，则将创建一个servlet对象。

> 在执行service方法后，会将响应结果放入response缓冲区中。在服务器中组织成一定结构。

> 由服务器发送到浏览器中，浏览器解析执行。

Request对象

> 代表HTTP请求的对象。

请求的组成：

> 一个请求行 GET /Day07-servlet/servlet/RequestDemo1 HTTP/1.1 多个请求头 一个空行 请求实体内容

继承结构

> ServletRequest --- request请求的最高级别的接口
>
> > -HttpServletRequest 在原有的接口之上实现了和HTTP协议相关的方法。这个接口更善于HTTP相关开发。

HttpServletRequest的API操作：

功能一：获取浏览器相关的信息

> getRequestURL方法 -- 返回客户端发出请求完整URL getRequestURI方法 -- 返回请求行中的资源名部分 getQueryString方法 -- 返回请求行中的参数部分 getRemoteAddr方法 -- 返回发出请求的客户机的IP地址 getMethod -- 得到客户机请求方式 !!getContextPath -- 获得当前web应用虚拟目录名称 -- 在写路径时不要将web应用的虚拟 路径的名称写死, 应该在需要写web应用的名称的地方通过getContextPath方法动态获取

功能二：获取请求头信息

> getHeader(name)方法 --- String getHeaders(String name)方法 --- Enumeration getHeaderNames方法 --- Enumeration getIntHeader(name)方法 --- int getDateHeader(name)方法 --- long(日期对应毫秒)

第九天

详见第九天笔记整理