总结

第一章java web 工作原理

1.1、web应用程序有web服务器，web客服端浏览器，HTTP协议以及静态HTML文件。

Web服务器的作用是接受客服端请求，然后向客服端返回些结果；

浏览器的作用是允许用户请求服务器上的某个资源，并且向用户显示请求的结果；

HTML是用于告诉浏览器怎么样向用户显示内容；

HTTP是web上客服端和服务器之间通信所用的协议。

1.1.2 HTTP协议将来自于客服端的请求信息封装成HTTP请求；

封装的信息当中包括请求行、请求头、消息体、分隔请求头、消息体的一个空行。

请求行是一个ASCII文本行，由三个标记组成：请求的HTTP方法、请求的URL、HTTP版本；中间用空格分开例如：

GET /lovobook/index.html HTTP/1.0

在HTTP1.1版本中请求方法有八种分别是下面：

GET：用于向服务器检索资源 在HTTP请求头

POST：用于向服务器发送资源，并要求指定的URI处理 在消息体

HEAD：于GET方法相同，服务器只返回状态行和头标，并不返回请求文档。

PUT：请求服务器保持请求数据作为指定的URI新内容；

DELETE：请求服务器删除URI中命名的资源；

OPTIONS：请求关于服务器支持的请求方法信息；

TRACE：请求web服务器反馈HTTP请求和其头标；

CONNECT：已文档化但当前未实现的一个方法，预留做隧道处理；

请求头：

HTTP协议使用HTTP头来传递请求的元信息。HTTP头是一个用冒号分隔的名称/值对，冒号前面是HTTP头的名称，后面是HTTP头的值。

1.1.3

HTTP响应包括：状态行、响应头、消息体、分割消息头、响应头。

状态行里面出现：

1XX:表示信息，请求收到，继续处理。

2XX:表示成功

3XX:表示重定向

4XX:表示客服端错误

5XX:表示服务器错误

1.2

Web服务器的缺陷是只能向用户提供静态网页内容。

1.3

服务器端网页编程就是web服务器创建动态服务器端内容的过程。

1.3.1

服务器端网页编程出现得最早的技术就是CGI，它的缺点就是每次请求一个CGI资源，将在服务器上创建一个新的进程，并且通过标准输入和环境变量将信息传递给该进程。

还有微软公司的ASP和ASP.NET技术

Java 运用的是servlet/JSP技术

Servlet主要用于在服务器端产生动态内容。它在服务器上主要的工作是向客户端读取和发送显示数据和隐式数据，以及在读取和发送中间生成结果。

PHP技术，它的特点是能快速创建小网站，但不能创建大型网站。

1.3.2

web容器是一种服务程序，在服务器一个端口就有一个提供相应服务的程序，而这个程序就是处理从客户端发出的请求，如JAVA中的Tomcat容器，ASP的IIS或PWS都是这样的容器。一个服务器可以多个容器。

Web容器对Servlet的支持包括：通信支持，生命周期管理，jsp支持和处理安全性。

**一、B/S模式的优点和缺点**

　　B/S结构的优点

　　（1）、具有分布性特点，可以随时随地进行查询、浏览等业务处理。

　　（2）、业务扩展简单方便，通过增加网页即可增加服务器功能。

　　（3）、维护简单方便，只需要改变网页，即可实现所有用户的同步更新。

　　（4）、开发简单，共享性强

　　B/S 模式的缺点

　　（1）、个性化特点明显降低，无法实现具有个性化的功能要求。

　　（2）、操作是以鼠标为最基本的操作方式，无法满足快速操作的要求。

　　（3）、页面动态刷新，响应速度明显降低。

　　（4）、功能弱化，难以实现传统模式下的特殊功能要求。

**二、C/S 模式的优点和缺点**

　　C/S 模式的优点

　　1.由于客户端实现与服务器的直接相连，没有中间环节，因此响应速度快。

　　2.操作界面漂亮、形式多样，可以充分满足客户自身的个性化要求。

　　3.C/S结构的管理信息系统具有较强的事务处理能力，能实现复杂的业务流程。

　　C/S 模式的缺点

　　1.需要专门的客户端安装程序，分布功能弱，针对点多面广且不具备网络条件的用户群体，不能够实现快速部署安装和配置。

　　2.兼容性差，对于不同的开发工具，具有较大的局限性。若采用不同工具，需要重新改写程序。

3.开发成本较高，需要具有一定专业水准的技术人员才能完成。

Web容器的功能是网络连接管理，会话管理等功能都由容器实现。

Servlet

Servlet是一种服务器端的Java应用程序，具有独立于平台和协议的特性,可以生成动态的Web页面。 它担当客户请求（Web浏览器或其他HTTP客户程序）与服务器响应（HTTP服务器上的数据库或应用程序）的中间层。 Servlet是位于Web 服务器内部的服务器端的Java应用程序，与传统的从命令行启动的Java应用程序不同，Servlet由Web服务器进行加载，该Web服务器必须包含支持Servlet的Java虚拟机

Servlet的方法分为以下几个类。

1. init()方法；

一个servlet在服务器上最多只会驻留一个实例。所以说第一次调用Servlet时，将会创建一个实例。在实例化的过程中，HttpServlet中的init方法会被调用。因此将一些初始化代码放在该函数内。

1. doGet()/doPost()/service()方法

Servlet 有两个处理方法：doGet()和doPost()。

doGet()在以get方式请求Servlet时运行。常见的get请求方式有链接、get方式表单提交，直接访问Servlet。

doPost()在以post方式请求Servlet运行。常见的post请求为post方式表单提交。事实上，客服端对Servlet发送一个请求过来，服务器端将会开启一个线程，该线程会调用service()方法，service()方法会根据收到的客服端的请求类型来决定是调用doGet()还是doPost()。但是一般情况下不用覆盖service()方法，使用doGet()与doPost()方法，同样可以达到处理目的。

1. destroy()方法

destroy()方法在Servlet实例消亡时自动调用。在web服务器运行Servlet实例时，因为一些原因，Servlet对象会消亡。但是在此Servlet消亡之前，还必须进行某些操作，比如释放数据库连接以节省资源等，这个时候就可以重写destroy()方法。

Servlet中如何获得内置对象的方法：

1. 获得out对象

Import java.io.PrintWriter;

…..

Public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

PrintWriter out = response.getWriter();

//获得out对象

}

在默认情况下out对象是无法打印中文的。因为out输出流中有中文却没有设置编码。解决这个问题可以讲doGet代码改为：

response.setContentType(“test/html;charset=gb2312”);

PrintWriter out = response.getWriter();

2获得request和response对象。这两个作为参数传给了doGet()或doPost()方法中。

3获得session对象。

Session对象对应的是HttpSession接口。

HttpSession session = request.getSession();

将session当成session对象来使用。

4获得application对象

application对象对应的是ServletContext接口

ServletContext application=this.get ServletContext();

将application当成application对象来使用。

可以使用application实现服务器内跳转。由于Servlet和JSP的同质性常用的Servlet内跳转有两种：

1. 重定向

response.sendRedirect(“URL地址”);

1. 服务器内跳转

ServletContext application=this.get ServletContext();

RequestDispatcher rd =application.getRequestDispatcher(“URL地址”);

rd.forward(request,response);

设置参数：

Web.xml文件有两种类型的参数设定。

1设置全局参数，该参数所有的Servlet都可以访问。

<context-param>

<param-name>参数名</ param-name >

< param-value >参数值</ param-value>

< /context-param >

1. 设置局部参数，该参数只有相应的Servlet才能访问

<servlet>

<servlet-name>Servlet名称</ servlet-name >

< servlet-class>Servlet类路径< /servlet-class>

<init-param>

<param-name>参数名</ param-name >

< param-value >参数值</ param-value>

</ init-param >

</servlet>

获取参数：

获取全局参数的方法是：

ServletContext application=this.get ServletContext();

application.getInitParameter(“参数名称”);

获取局部参数的方法是：

this.getInitParameter(“参数名称”);

注意：此处的this是指Servlet本身。

Servlet基础

一、什么是Servlet?

1、是运行在Web服务器或应用服务器上的Java程序

2、是在Web上创建动态内容的有效而强大的解决方案

3、有容器来管理生命周期与Web服务器交互

4、由sun规范了其功能

二、Servlet的生命周期

1、在整个服务器运行过程中，一个Servlet只有一个对象

2、这个对象在第一次请求来的时候产生，所有对相同Servlet的请求都是访问同意对象

3、Servlet不是线程安全的

4、Servlet对象会在服务器关闭或重新加载的时候被销毁

5、方法调用 init()---service(doget/dopost)---destory

解释：init()：初始化

重写service()方法：真正接受请求的，了解生命周期，判断请求类型（get/post）,在调用doget()或dopost()方法

destroy() ：servlet销毁方法

注意：Servlet在第一次请求来的时候产生对象这不是绝对的，可以在web.xml里面配置，在服务器启动的时候产生对象并告诉产生的顺序

三、控制生命周期(出现的顺序)

在web.xml文件中servlet元素中添加子元素<load-on-startup>自然数（改变顺序）</load-on-startup>

四、参数的初始化

在web.xml文件中servlet元素添加子元素 </init-param> </init-param>（需要注意添加顺序），写在<load-on-startup>之前

例：

<init-param>

<param-name>encoding</param-name>

<param-value>utf-8</param-value>

</init-param>

五、Servlet的架构体系

Servlet(接口) ServletConfig(接口)

GenericServlet(抽象类)

HttpServlet

六、Request对象

1、什么叫request对象：客户端请求对象，对象里面封装了当次客户请求的所有

2、用request对象来取得请求的参数req.getParameter(String s)

3、参数的传递格式 ：url?xx=xx&yy=yy.....

4、表单提交的时候表单控件参数就是name=value

5、处理中文乱码

1)、post方式请求设置页面以及request对象的字符编码级utf-8 setCharacterEncoding("utf-8")

2)、get方式配置服务器端口元素的字符编码 URIEncoding="utf-8"

七、通过流取到动态的网页

resp.setCharacterEnconding("utf-8") //乱码

resp.setContentType("text/html;charset=utf-8") //设置响应格式里的消息体（乱码）

八、一个键获取多个值：

String[] str = req.getParameterValues("che");

for(String s = str){

System.out.println(s+"========");

}

总结：

1、Servlet的配置

2、初始化参数的配置以及获取 <init-parm>

3、Servlet的对象创建顺序的配置 <load-on-start>

4、HttpServletRequest对象的方法：getParameter(),getParameterValues(),setCharacterEnconding()

5、response对象的方法：setContentType(),setCharacterEnconding(),getWriter()

Session会话

Session表示通话，会话，对话期间，对谈时间的意思。

Session:服务器端的一块内存，可以和服务器对应，每个客户端都有自己独立的一个区域。

Session里面所有的方法都是键值对

Session里面的内容并不是永无止境的保存，它会有一定的时间限。

Session课堂总结：

session会话：就是客户端(浏览器)对服务器的一次会话范围，也就是在服务器端的一块内存区域，一个客户端对应自己的一块内存区域，可以向这块内存里面存取数据。数据的格式就是键值对。(key的类型只能是String,value的类型是Object)

①：HttpSession session = req.getSession(); //从服务器得到session对象

session.setAttribute("User","你好"); //把session对象放进去

②：HttpSession session = req.getSession(); //从服务器得到session对象

Object o = session.getAttribute("User"); //把session对象取出来

③：HttpSession session = req.getSession().removeAttribute("User"); //删除，移出session对象

④：HttpSession session = req.getSession(); //从服务器得到session对象

session.setMaxInactiveInterval(10); //设置请求的时间(秒) 多少秒后就取不到了

ServletContent

ServletContent：Servlet上下文，与session用法相同

课堂总结：

请求转发：RequestServlet req.getRequestDispatcher("s2").forward(req,resp);

请求转发(服务器端跳转)：不会结束当次请求，而是直接在服务器内部把这一次请求进行跳转。request范围一样，可以通过request范围传递数据。

重定向：resp.sendRendriect("s2");

重定向(客户端跳转)：结束当次请求，让客户端发起新的请求去访问。request范围不是一般在这一次请求结束的时候，为了导致后台的混乱时使用。

三大范围：

1、request：当次请求范围，只能在一次请求的过程当中存取数据，一般用来数据的简单传递。请求开始时产生，请求结束时消失(一次请求中)。

2、session:会话，用来保存客户端的状态，只是服务器端的一块内存区域。每个客户端都有自己的一块内存区域，相互之间不冲突。(第一次请求时产生，可以控制超时时间)。

3、ServletContent:servlet上下文，整个服务器端最大的一块内存区域，所有的Servlet共享，保存全局性的数据。(开始时就有，结束时消失)

过滤器:Filter

filter概念：拦截客户端到服务器端的请求(过滤)，可以帮助我们在执行具体 业务之前进行请求的过滤。常用来做权限，分类，以及一些通用项目设置。

书写方式：要让自己的类成为一个过滤器(拦截器)去实现filter接口，要让filter起作用就必须在web.xml里面配置。

生命周期：在服务器启动的时候创建对象，整个运用过程中只有一个对象服务，服务器重新加载或者关闭的时候销毁。

初始化参数：跟servlet的初始化一样

注意：1、页面上资源访问，比如图片，音乐，视频等等都是在向服务器发送请求，所以注意拦截器拦截这请求的时候会让资源不能访问。

2、拦截器只能够拦截客户端发送到服务器端的请求，不能够拦截服务器内部的跳转。

cookie：服务器向客户端发送的一个文本信息，并且保存在客户端。文本信息的格式是key-value，最大不能超过4k.

可以通过request对象里面把当前请求的客户端的所有cookie拿到，可以通过response对象发送cookie到客户端。

cookie的生命周期通过设置存在的时间长度来决定。

session就是通过cookie实现的，每个session对象都有一个id值(唯一的)，这个id会发送到浏览器的cookie里面，每次浏览器请求服务器，服务器都会把这个id从cookie取出来，然后再去匹配session对象的id,这样来确定到底当前的浏览器该使用哪一个session对象。

session比cookie更安全。通常情况下都会选择使用session.cookie不安全，一般用来保存一些不是特别关键的数据，可以减少服务器的压力。因为session是服务器端的内存。cookie是客户端保存。

一旦客户禁用了cookie，这个时候session就会失效，我们可以通过URL重写来让session继续生效。URL重写就是改变你原来的访问地址和URL，让URL上面跟上session ID作为参数来传递

监听器

一、监听器:在servlet里面用来监听三大范围对象的事件，提供了8个借口来针对3个对象的事件监听。

二、事件触发主要有两大类型：

1、对象自身的生命周期(产生，销毁)

2、对象范围内数据的变化(添加，移除，替换)

三、三大对象对这两大类型都有自己的接口：

ServletContextListener ServletRequestListener

ServletContextAttributeListener ServletRequestAttributeListener

HttpSessionListener

HttpSessionAttributeListener

四、session对象还有另外两个自己的监听：

1、和某个特定的类型的对象绑定的监听器接口(只监听该类型的对象被添加或者销毁的事件)。

HttpSessionBindingListener

2、监听session被钝化或者激活的一个监听接口(了解) HttpSessionActivationListener

注意：所有的监听器都需要在web.xml里面注册了才能生效，只有HttpSessionBindingListener监听器不需要注册。

第八章：JSP基础

一、JSP就是服务器端的动态网页技术，就是能够在HTML网页上面书写java代码。

二、JSP的工作原理：

jsp在被访问的时候会被翻译成一个servlet(处理jsp翻译的servlet类)类，然后编译成class文件，然后产生servlet对象来接收请求。(jsp本质就是一个servlet，在服务器里面都有jsp引擎)

1. JSP的生命周期：

它只比servlet多一点“翻译”(生命周期和servlet一样)

1. JSP的基本语法：

①、<% %>它们之间能够书写Java代码，这里的代码就相当于在翻译成德servlet的service方法内部(局部变量)

②、<%! %>它们之间能够书写Java代码，这里的代码就相当于在方法的外面类的里面(成员变量)

③、<%= %>等号后面可以跟一个变量或者一个具体的值相当于、、、

五、JSP内置对象：jsp已经准备好的可以直接使用的对象(9个)

①、request 请求对象

②、response 响应对象

③、pageContext 当前页面的上下文，里面包含了当前页面的所有数据信息

④、page 当前页面的对象指针

⑤、config servlet参数(servletConfig)

⑥、out 输出流(PrintWriter)

⑦、session 会话

⑧、application 服务器上下文(servletContext)

⑨、excption 异常处理对象(throws,excption)

六、三大指令

page（页面指令）

taglib（引入标记库）

include（静态包含）

七、动作

include（动态包含）

forward（请求转发）

注意：静态包含和动态包含的区别

1、静态包含是在编译期执行包含这个步骤，把包含的页面的所有的代码直接拿到本页面的代码里面来，所有两个页面最终只会生成一个Java类，也就是说所有的属性，对象都共享。

2、动态包含会在执行期间动态的把包含页面的信息拿到本页面来，它们还有独立的两个页面会生成两个类，它们各自使用各自的参数，对象。向本页面传递的参数在包含页面也能够取到。但是它们的request对象不是同一个。

标 记 库

EL表达式：sun公司定义一套方便处理Java表达式的语法，通常

用来在JSP进行范围取值。

语法：${ }

注意：1、在对对象取值的时候一定要有get方法，而且后面的

那个名称不是属性名，而是get方法后面的名称(首

字母小写)

2、取值的顺序是：pageContext、request、session、

applicaton

3、可以对集合、数组、Map进行取值，能够进行逻辑

表达式，关系表达式

JSTL：sun公司定义一套JSP标签库，用来处理业务逻辑，核心标签

是 c

注意：怎么引入标签

会使用逻辑判断语法，以及迭代语法，设置语法，了解一下格式

化与函数

判断为空，用==(在EL表达式里)

MVC架构模式

MVC：

M：Model 模型层(service dao)

V：View 视图层(HTML,JSP)

C：Controller 控制层(servlet)

MVC的主流思想：就是让视图和业务所相分离，降低偶合度，使系统更具扩展性，可维护性。

JavaWeb: jsp+servlet+javabean（普通的Java类而已，有无参构造方法，必须提供get，set方法，一般用来封装数据）

数据建模

数据建模的步骤：

(CDM)1、概念数据建模(把客观存在的事物形成自身的概念上的东西，就是找对象，找关系)

2、逻辑数据建模(分析具体对象有哪些属性，数据类型是什么)

(PDM)3、物理数据建模(和具体的数据库有关，并且在真正的数据库系统里面实现)

第一章：JavaScript核心基础语法

1. JavaScript的概念：

一种嵌入在HTML内部的脚本语言，是一种解释性的语言，它主要用来丰富客户端内容，用来提高用户体验。

1. JavaScript的具体功能：点击事件、伪URL
2. JavaScript能够在HTML里面书写在什么地方

1、head里面的script标签内

2、body里面的script标签内

Script标签的几个属性：type(type="text/javaScript") 、defer(defer="defer" 延迟加载，基本浏览器都不支持，IE支持)、

src(src=" jsa.js") 、charset(charset="utf-8" 规定引入的文件的字符编码级)、language(langusge="JavaScript" 已过时的)

3、事件执行程序里面

4、伪URL，比如超链接的href里面 (<a href="JavaScript=alert('伪URL')">伪URL</a>)

5、外部文件

1. 体系架构

1、ECMAScript 核心(语法)

2、DOM(文档对象模型)

3、BOM(浏览器对象模型)

1. 基础语法

1、数据类型：

基本数据类型：number(数值型 在运算时把不能运算的转换为运算的)、boolean、String、undefined(未初始化)、null(空指针)

(转换的时候首字母大写)

引用数据类型

2、变量定义(弱类型) var=

3、Switch

4、全等 === 相等==

5、for in

1. JavaScript的应用：

1、增加网页交互性

表单校验、页面修饰也特效、制作导航系统、执行基本的算术运算、动态文档生成、操作结构化文档

2、解释性语言

3、出于安全性考虑，增加了JavaScript的限制

1. 函数的初步

1、没有返回类型

2、没有参数列表，可以传入0到多个任意类型的参数

arguments来取得参数

3、可以返回任何数据类型，也可以不返回

4、没有重载，重名的函数会被覆盖掉

1. 变量的作用域

1、没有块级的作用域，也即是没有结构体的概念

2、通过执行环境(宿主环境)来规定作用域

3、同一个环境下面的都可以相互调用

九、引用数据类型

1、原生对象：跟环境没有关系，是JS语言自身拥有的对象，这些对象需要你自己创建(Array、date、Object、Function、RegExp、所有的包装器)

2、内置对象：也是原生对象，但是已经准备好可以使用的对象(Math、Global全局对象) 不需要声明和初始化

3、宿主对象：跟执行环境有关的对象(DOM、BOM)

4、自定义对象

Eval方法：直接把字符串转化为js代码运行，如果是数组要加小括号

第二章：JavaScript面向对象编程

一、

1、工厂模式(先创建对象，然后往对象中添加属性和方法)：不能区分类型，创建的对象都是Object类型

2、构造函数模式(constractor)：已经有类型区别，并且通过new关键字在创建对象，方法会被创建多次

3、原型模式：不能够写参数进行初始化，属性也被共享(利用了对象的prototype属性)

4、混合模式(构造函数与原型模式混合)：避免方法的重复创建，达到属性的分离

5、动态原型模式：看起来更具封装感

6、字面量(JSON)：键值对，以逗号分割，冒号左边的可以用引号引起来 具体的一个对象，没有类型

1. 对象的继承

1、原型继承：属性也存在于原型对象身上，不能够传递参数给父函数(属性赋值) 不支持多重继承

2、借用构造函数

3、混合模式(通过原型继承来继承方法，通过构造函数哎继承属性)

第三章：JavaScript事件模型

1. 什么是事件：

即是HTML页面上组件在特定情况下发生的某些事情，或者用户在网页上触发的某些事情。

1. 事件处理模型：

1、捕获阶段：从外往里

2、冒泡阶段：从里往外

现在的事件处理程序都是先进行捕获阶段，然后在进

冒泡阶段，但是在捕获阶段不会触发事件处理。

1. JS事件处理步骤：

1、确定事件类型，找到监听器

2、书写事件处理程序

1. 事件处理的方式

1、DOM 0级别的处理方式(支持IE，火狐。。。)

2、DOM 2级别的处理方式(不支持IE，支持火狐) 可以添加多个监听事件

(添加)三个参数：①、事件类型字符串(不要on)②、事件处理的函数③、是否在冒泡或者捕获阶段来触发(false:冒泡触发)

删除某个事件(2个参数)：①：事件类型②：对应事件处理函数

3、IE的动态事件添加(第一个参数要带on)

1. 事件创建的方式

1、可以直接在元素上书写事件的监听属性，属性的值书写处理程序代码

2、通过元素对象事件属性赋值处理函数(DOM 0级)

3、通过添加监听器的方法来添加事件，以及通过监听来删除事件(DOM 2级)

1. 事件对象

Event:代表当前发生事件的信息封装对象。

对于IE来说，事件对象是window的属性(window.event)；

对于标准来说(火狐)，会把事件对象作为参数传递到处理函数里面

1. 默认行为

元素本身在某些事件触发的时候有默认的处理程序，比如超链接

1. 取消默认行为(是在事件对象身上去取消)

1、在自身所对应的事件处理程序里面，最后返回false (preventDeault)

2、火狐调用方法，IE设置属性

1. 取消冒泡行为

1、火狐 event.stopPropagation();

2、IE event.cancelBubble = true;

1. 事件类型

1、UI事件

2、鼠标事件

3、键盘事件



5771001803090012095 579036822859633082

5771001803090012386 576137399735760696

5771001803090013594 578077579902515512

5771001803090012387 577164982601818051

5771001803090012138 572131192158918326

5771001803090012359 579036822361076053

5771001803090012356 576135286143791742

5771001803090012355 575087869704693279

17088100343355274 101229944325833379

17088100343355275 101866732938832008

17088100343356107 101581152501500522

17088100343356108 101000180059871732

17088100343354295 101074194142687017

17088100343356184 101878660869628802

17088100343356185 101775831174086674

17088100343356109 101086014373572846

17088100343356110 101152207216014916

17088100343355237 101027041605702709

17088100343355238 101229364861425414

17088100343356169 101862204402635718

17088100343354928 101760654089788804