# Review

## HTML

### 基本HTML标签

<html>

<head>

<title>网页标题</title>

<meta charset="utf-8"/><!-- 编码 -->

</head>

<body>

<h1>标题标签</h1>

…

<h6>标题标签</h6>

<p>段落</p>

换行<br>

空格&nbsp;引号&quot;大于&lt;小于&gt;

<img src=”图片地址” alt=“图片错误显示的文字” title=“鼠标放图片上显示的文字”/>

分割线<hr>

<ol>

<li>有序列表</li>

</ol>

<ul>

<li>无序列表</li>

</ul>

<a href=”超链接网址” target=”\_blank”>超链接 \_blank新窗口打开</a>

<a href=”#”>返回顶部</a>

<a href=”#标签id属性的值”>锚点</a>

</body>

</html>

### 表格

<table>

<tr>

<th>列的头部信息</th>

</tr>

<tbody>

<tr>

<td>单元格</td>

<td rowspan=”3”>跨行</td>

<td colspan=”3”>跨列</td>

</tr>

</tbody>

</table>

### 表单

<form action=”数据提交到的地址” method=”get/post提交方式”>

<input type=”text” value=”输入框默认值”/>input标签type为空 默认是text

<input type=”password” value=”密码”/>

<input type=”checkbox” value=”多选按钮的值”/>

<input id=”单选按钮的id” type=”radio” value=”单选按钮的值”/>

<label for=”单选按钮的id”>单选按钮显示的文字，</label>

加了label标签之后选择文字也能达到选中按钮的效果

<input type=”date” value=”选择时间”/>

<input type=”color” value=”颜色”/>

<input type=”submit” value=”提交按钮”/>

<input type=”reset” value=”重置按钮”/>

<input type=”button” value=”普通按钮”/>

<select >

<option value=”select标签的值”>下拉菜单显示的内容</option>

<option value=”真正的值”/>下拉菜单显示的内容

<option value=”真正的值” selected=”selected” />也可以直接写成 selected

</select>

<textarea cols=”占的列数” rows=“占的行数”>文本域的值，两个标签之间不要加回车和空格</textarea>

</form>

每个标签都具有属性，通常情况下 ：

* Javasc控制页面使用id属性和class属性、
* 表单提交数据使用name属性、
* css样式控制使用class属性

|  |  |  |
| --- | --- | --- |
|  | **GET** | **POST** |
| 后退按钮/刷新 | 无害 | 数据会被重新提交（浏览器应该告知用户数据会被重新提交）。 |
| 书签 | 可收藏为书签 | 不可收藏为书签 |
| 缓存 | 能被缓存 | 不能缓存 |
| 编码类型 | application/x-www-form-urlencoded | application/x-www-form-urlencoded 或 multipart/form-data。为二进制数据使用多重编码。 |
| 历史 | 参数保留在浏览器历史中。 | 参数不会保存在浏览器历史中。 |
| 对数据长度的限制 | 是的。当发送数据时，GET 方法向 URL 添加数据；URL 的长度是受限制的（URL 的最大长度是 2048 个字符）。 | 无限制。 |
| 对数据类型的限制 | 只允许 ASCII 字符。 | 没有限制。也允许二进制数据。 |
| 安全性 | 与 POST 相比，GET 的安全性较差，因为所发送的数据是 URL 的一部分。  在发送密码或其他敏感信息时绝不要使用 GET ！ | POST 比 GET 更安全，因为参数不会被保存在浏览器历史或 web 服务器日志中。 |
| 可见性 | 数据在 URL 中对所有人都是可见的。 | 数据不会显示在 URL 中。 |

## CSS

### 页面引入样式表的三种方式

* 引入外部样式表<link rel="stylesheet" href="index.css">、@import 推荐
* 当前页面写样式标签<style type="text/css"></style> 偶尔使用
* 标签上直接写样式属性 <div style=””></div> 尽量不用
* 大量的样式/整个网站频繁用到的样式 放在文件里，因为加载文件需要网络成本，所以不同页面的一些零散的样式，可以直接写在style标签里，也可以集中放在一个样式文件中，文件加载一次，就会被浏览器缓存起来，整个浏览过程中都不会再次加载

### 三种选择器

* 类选择器 .类名
* Id选择器 #id
* 标签选择器 p

### 常用样式

* + 字体、颜色、大小、
  + 背景图或颜色
  + 左右浮动
  + margin、padding、border 及其简化写法
  + a:hover

### DIV+CSS布局

## JavaScript

### 页面引入js的三种方式

* + 引入外部文件：<script type="text/javascript" src="index03.js"></script>
  + 页面写标签：<script type="text/javascript"></script>
  + HTML中直接写：<a href="javascript:alert('提示信息')">你好</a>

### 常用内置函数

* + alert(“警告框”)、confirm(“确认框”)
  + parseInt()
  + parseFloat()

### 命名规范和声明

* 变量必须以字母或下划线或美元符$开始 ，不能使用关键字作为变量名
* Js是弱变量语言，所有的变量都可以使用var来声明 例如：var num = 10;
* var str = “你好”;这里的分号可以省略，双引号可以写成单引号 var str = ‘你好’

### 数据类型

* + Undefined
  + Null
  + Boolean
  + String

### 运算符号

* + 算数运算符
  + 赋值运算符
  + 比较运算符：注意 =、==、===区别
  + 逻辑运算符

### 判断语句

* + if、switch
  + 三目运算符 结果变量 = (判断条件)?如果判断条件为true 赋值:如果判断为false 赋值

### 循环语句

* + for、for(a in b)
  + while、do{}while()
  + break、continue

### 自定义函数

* + function 函数名 (参数1,参数2){}
  + var 函数名 = function(参数1,参数2){}
  + 函数返回值使用return
  + 函数之间调用 直接在方法体中写 方法名()

### 变量作用域

* + 全局变量：声明在方法外或者声明时没有写var
  + 局部变量：声明在方法内

### 事件

* + 点击/双击
  + 鼠标移动、移入、移出、悬浮
  + 获得焦点、失去焦点
  + 改变值
  + 提交

### Window对象常用属性

* + history
    - back()相当于浏览器的回退按钮
    - forward相当于浏览器前进按钮
    - go(n) n是整数前进 负数后退 0刷新
  + location
    - location.href=”页面跳转的地址”
    - location.reload();刷新当前页面
  + document
    - title：返回或设置当前文档的标题 document.title=”改变标题”
    - write()：向文档写HTML表达式或JavaScript代码
    - getElementById():返回对拥有指定id的第一个对象的引用
    - getElementsByTagName():返回带有指定名称的对象集合
    - getElementsByName():返回带有指定名称的对象集合
    - getElementsByClassName();返回指定class名称的对象集合

### Window对象常用方法

* + alert()、confirm()
  + open()打开新窗口
  + close()关闭窗口
  + setTimeout(‘fn()’,1000) 1000毫秒之后执行一次fn方法
  + setInterval(‘fn()’,1000) 每1000毫秒执行一次fn方法
  + clearTimeout(对象);取消setTimeout
  + clearInterval(对象) 取消setInterval

### Document对象

* + 获得节点对象：document. getElementById()等方法获取
  + innerHTML 设置/返回 节点中间的html代码
  + innerText 设置/返回 节点中间的文字
  + value 设置/返回 节点的值
  + className 设置/返回 class属性的值
  + parentNode属性 返回父节点
  + getAttribute(“属性名”)获取节点的属性值
  + setAttribute(“属性名”,”值”)给节点的属性赋值
  + ~~createElement(“input“)创建节点~~
  + ~~追加：A节点.appendChild(B节点)~~
  + ~~克隆：节点.cloneNode(true) true复制子节点 false不复制子节点~~
  + removeChild(node)删除节点

### Table对象

* + Table对象.rows[0] 一个tr对象的数组
  + Table对象.rows[0].cells[0] 一个td对象的数组
  + Table对象.deleteRow(0) 删除一行
  + Table对象.insertRow(0) 插入一行

### 常用内置对象

* + String对象
    - indexOf(“str”) 某个字符串第一次出现的位置
    - substring(10,20) 截取字符串
    - split(“,”) 分隔符，字符串分割为数组
  + Date对象
    - new Date();返回当前时间
    - .getDate();返回一个月中的某一天，1~31之间
    - .getDay();星期几 0~6之间
    - .getHours(); 小时数 0~23
    - .getMinutes();分钟 0~59
    - .getMonth();月份 0~11
    - .getFullYear();年 4位数
    - .getTime();毫秒数
  + Array对象
    - new Array();创建空数组
    - new Array(2);创建2个undefined值的数组
    - new Array(1，2，3，4);创建数组并赋值
    - length 返回数组长度
    - join()/join(“.”) 以某个符号连接数组为一个字符串
    - sort() 数组排序
    - push() 向数组末尾添加元素 返回新长度
  + Math对象
    - Math.random();返回随机数0.9944943

### 改变页面样式

* + ~~通过style属性修改~~
  + 改变class值

## jQuery

### 引入jQuery库

* + <script src="jquery-3.2.1.min.js"></script>

### 基本语法

* + $是方法名，相当于方法jQuery();
  + $(document).ready(function(){ });
  + $(function(){ });
  + $(选择器).xxx();

### HTML

* + $(“#id”).html();获得html代码
  + $(“#id”).val();获得value属性的值
  + $(“#id”).text();获得文本

### CSS

* + $(“#id”).css(“样式名”,”值”);//设置一个样式，直接修改标签的style属性
  + $(“#id”).css({“样式1”:”值”,”样式2”:”值”});//设置多个样式，传入参数是对象

### 类

* + $(“#id”).addClass(“class名”);//在原来class属性值的基础上，追加值
  + $(“#id”).removeClass(“class名”);//删除class属性的一个值，其他值不动

### 属性

* + $(“#id”).attr(“name”);返回属性的值
  + $(“#id”).attr(“属性名”,”值”);添加一个属性的值，添加多个属性传入对象{}
  + $(“#id”).removeAttr(“属性名”);//移除某一个属性

### 尺寸

* + $("p").height();//高度 获取/设置值
  + $("p").width();//宽度 获取/设置值

### 文档(节点)

* + 内部插入
    - $("p").append("<b>Hello</b>"); 元素内部后(结尾)追加内容
    - $("p").prepend("<b>Hello</b>"); 元素内部前(开始)追加内容
  + 外部插入
    - $("p").after("<b>Hello</b>"); 元素外部之后追加
    - $("p").before("<b>Hello</b>"); 元素外部之前追加
  + 包裹
    - $("p").wrap ("<div></div>"); 元素之外包裹一个标签
  + 替换
    - $("p").replaceWith("<b>Paragraph. </b>"); 选中元素替换为内容
  + 删除
    - $("p").empty(); 删除所有子节点，保留元素本身
    - $("p").remove(); 删除元素本身及其子节点,包括绑定事件附加数据也会移除
    - $("p").detach(); 删除元素本身及其子节点,保留绑定事件等
  + 复制
    - $("b").clone(); 克隆一个副本

### jQuery对象和DOM对象转换

* + jQuery对象 = $(DOM对象)
  + DOM对象 = jQuery对象[0]/ jQuery对象.get(0);

### 选择器

* + 基本
    - $(“#id”) id选择器
    - $(“.class”) class选择器
    - $(“标签”) 标签选择器
    - $(“#id,.class,p,h1”) 并集选择器
  + 层次
    - $("form > input") 子选择器
    - $("label + input") 相邻选择器
    - $("form ~ input") 同辈选择器
  + 属性
    - $("input[name='newsletter']").attr("checked", true); 属性选择器
  + 基本过滤
    - $('li:first'); 选取第一个
    - $('li:last') ; 选取最后一个
    - $("tr:even") 偶数元素
    - $("tr:odd") 奇数元素
    - $("tr:eq(1)") 第n个元素
    - $("tr:gt(0)") 大于n的元素
    - $("tr:lt(2)") 小于n的元素
  + 可见性过滤
    - $("input:hidden") 不可见元素
    - $("tr:visible") 可见元素
  + 表单
    - $(":image") 匹配所有图像域 其他标签的选择器查看api文档
    - $("input:checked") 匹配所有选中的被选中元素(复选框、单选框等)
    - $("select option:selected") 匹配所有选中的option元素
  + 内容
    - $("td:empty") 匹配空元素
    - $("div:has(p)") 匹配又某个标签的元素

### 筛选

* + 过滤
    - $("p").eq(1) 第n个元素
    - $('li').first() 第一个
    - $('li').last() 最后一个
  + 查找
    - $("div").children(".selected") 查询子元素集合(符合某一条件)
    - $("p").find("span") 从元素中检索某个元素
    - $("p").next() 下一个同辈元素
    - $("p").prev() 上一个同辈
    - $("p").parent() 父元素
    - $("p").parents() 所有祖先元素(父元素的父元素…一直向上找)

### 对象访问

* + $("img").each(function(i){ });
    - this//方法中的this是js中的DOM对象,指自己本身
    - i//是循环到第几个元素,0开始
  + $("img").size();$("img").length; 几个元素，值相同

### 事件

* + $("p").click( function () { $(this).hide(); });//为元素绑定点击事件
  + 其他事件绑定方式同click，查找api文档
  + $("p").on("click", 方法); 绑定事件
  + $("p").off() 解除绑定事件

### 动画效果

* + $("p").show(); 显示
  + $("p").hide(); 隐藏
  + $("p").toggle(); 切换
  + 滑动
  + 淡入淡出
  + 自定义

### JSON

* + 是js中的对象
  + 可以通过对象.属性的方式获取值
  + var obj = {名称：值，名称：值}

### 插件

* + jQuery Easy UI
  + jQuery UI
  + jQuery Tree
  + Amaze UI
  + 能够快速上手使用一个jQuery插件库

## Bootstrap3

### 引入Bootstrap库

* 引入css样式表
  + <link rel="stylesheet" href="css/bootstrap.min.css" >
* 引入js文件,放在jQuery之后引入，依赖于jQuery
  + <script src="js/bootstrap.min.js" ></script>

### 栅格系统

* 12列
* 外层div
  + .container 固定宽度
  + .container-fluid 100%宽度
* <div class=”row”></div>
  + .col-xs-12 超小屏幕
  + .col-sm-12 小屏幕
  + .col-md-12 中等屏幕
  + .col-lg-12 大屏幕
* 列偏移
  + .col-md-offset-1 同上，四个分辨率下的列偏移

### 组件/插件/样式

* 参考官方文档

### 可视化布局

* http://www.ibootstrap.cn/
* https://www.layoutit.com/

### 插件

* bootstrapValidator 表单校验插件
* bootstrap-dialog 弹出层
* bootstrap-datetimepicker 时间插件
* 记得优先上网寻找自己需要的插件

### 富文本编辑器

* UEditor 官网：http://ueditor.baidu.com/website/
  + 百度富文本编辑器，功能非常齐全
* bootstrap-wysiwyg 官网：http://www.bootcss.com/p/bootstrap-wysiwyg/
  + 基于BootStrap的轻量级富文本编辑器
* 编辑带样式/格式的类word文档时，需要用到富文本编辑器，如编辑新闻页

### 图表

* Echarts 官网：http://echarts.baidu.com/
  + 百度的可视化图表绘制插件
  + API完全是中文，非常容易上手，方法封装层次较高，遇见问题容易解决
  + 目前国内最好用最强大的插件库，有许多非常炫酷的图表
  + 支持多种图表样式，对大数据的展示有很大帮助
* Chart.js 官网：https://chartjs.bootcss.com/
  + 是一个简单、面向对象、为设计者和开发者准备的图表绘制工具库
  + 较Echarts更轻量级，适合处理简单常见的图表
* D3 官网：https://d3js.org/
  + 国外的一款图表插件，收录了许多非常好的插件
  + API完全是英文，不容易上手，需要较高的H5+CSS+JS功底才能使用
  + 另外，Echarts许多图表创意都是模仿此网站的实例。
* 其他图表插件等待你自己去发现

## Servlet

### B/S结构

* + 浏览器/服务器
  + 浏览器与服务器通过请求/响应模式进行交互
  + 浏览器通过url访问页面
  + URL:http://localhost:8080/myproject/index.html
  + 协议/ip:端口号/项目名/文件名

### WEB项目

* Web项目目录结构
  + - / 根目录
    - /WEB-INF 被保护的文件夹，外部不能直接访问，只能内部访问
    - /WEB-INF/classes 存放web项目的class文件
    - /WEB-INF/lib 存放web项目的jar文件
  + web.xml
    - 文件必须保存在/WEB-INF文件夹下
    - 存放整个web项目的配置信息

### Tomcat部署项目

* 启动/停止文件
  + /bin/startup 启动
  + /bin/shutdown 停止
  + .bat文件是Windows系统运行 .sh文件是Linux系统运行
* 提高Tomcat的最大内存
  + 如果Tomcat运行报内存溢出错误，可以考虑增加最大内存
  + 修改catalina.bat/.sh 文件
    - 首行加入set JAVA\_OPTS=%JAVA\_OPTS% -Xms128m -Xmx512m
    - 表示初始化最小可用内存是128MB,最大可用内存是512MB
    - 根据实际需要修改这两个值
* 修改端口号
  + /conf/server.xml
    - <Connector port="8080" protocol="HTTP/1.1" connectionTimeout="20000" redirectPort="8443" />
    - 增加属性
      * disableUploadTimeout="true"允许Servlet容器，正在执行使用一个较长的连接超时值
      * useBodyEncodingForURI="true"指示是否使用在contentType中指定的编码来取代URIEncoding
      * URIEncoding="utf-8"
    - 属性介绍
      * allowTrace：是否允许HTTP的TRACE方法，默认为false
      * emptySessionPath：如果设置为true，用户的所有路径都将设置为/，默认为false。
      * enableLookups：调用request、getRemoteHost()执行DNS查询，以返回远程主机的主机名，如果设置为false，则直接返回IP地址。
      * maxPostSize：指定POST方式请求的最大量，没有指定默认为2097152。
      * protocol：值必须为HTTP1.1，如果使用AJP处理器，该值必须为AJP/1.3
      * proxyName：如这个连接器正在一个代理配置中被使用，指定这个属性，在request.getServerName()时返回
      * redirectPort：如连接器不支持SSL请求，如收到SSL请求，Catalina容器将会自动重定向指定的端口号，让其进行处理。
      * scheme：设置协议的名字，在request.getScheme()时返回，SSL连接器设为”https”，默认为”http”
      * secure：在SSL连接器可将其设置为true，默认为false
      * URIEncoding：用于解码URL的字符编码，没有指定默认值为ISO-8859-1
      * useBodyEncodingForURI：主要用于Tomcat4.1.x中，指示是否使用在contentType中指定的编码来取代URIEncoding，用于解码URI查询参数，默认为false
      * xpoweredBy：为true时，Tomcat使用规范建议的报头表明支持Servlet的规范版本，默认为false
      * acceptCount：当所有的可能处理的线程都正在使用时，在队列中排队请求的最大数目。当队列已满，任何接收到的请求都会被拒绝，默认值为10
      * bufferSize：设由连接器创建输入流缓冲区的大小，以字节为单位。默认情况下，缓存区大的大小为2048字节
      * compressableMimeType：MIME的列表，默认以逗号分隔。默认值是text/html，text/xml，text/plain
      * compression：指定是否对响应的数据进行压缩。off：表示禁止压缩、on：表示允许压缩（文本将被压缩）、force：表示所有情况下都进行压缩，默认值为off
      * connectionTimeout：设置连接的超时值，以毫秒为单位。默认值为60000=60秒
      * disableUploadTimeOut：允许Servlet容器，正在执行使用一个较长的连接超时值，以使Servlet有较长的时间来完成它的执行，默认值为false
      * maxHttpHeaderSize：HTTP请求和响应头的最大量，以字节为单位，默认值为4096字节
      * maxKeepAliveRequest：服务器关闭之前，客户端发送的流水线最大数目。默认值为100
      * maxSpareThreads：允许存在空闲线程的最大数目，默认值为50
      * minSpareThreads：设当连接器第一次启协创建线程的数目，确保至少有这么多的空闲线程可用。默认值为4
      * port：服务端套接字监听的TCP端口号，默认值为8080（必须）
      * socketBuffer：设Socket输出缓冲区的大小（以字节为单位），-1表示禁止缓冲，默认值为9000字节
      * toNoDelay：为true时，可以提高性能。默认值为true
      * threadPriority：设JVM中请求处理线程优先级。默认值为NORMAL-PRIORITY
    - <context>元素
      * <Content path=”/abc” docBase=”d:/xyz” reloadable=”true” />
      * 访问路径是abc
      * 项目实际路径是d:/xyz
      * Tomcat运行时，如果WEB-INF/classes和WEB-INF/lib目录中有改变，Tomcat会自动重新加载该WEB应用程序。开发时使用，正式环境下关闭
* 项目部署文件夹
  + /webapps
  + 所有发布的项目都在此文件夹中
  + 此文件夹中的名字，即是浏览器访问时的项目名
  + ROOT文件夹的内容不需要项目名即可访问
  + 可以发布.war文件
* 运行文件目录
  + /work
  + 编译之后的jsp文件在这里
  + 调试如果遇见缓存(无论如何修改页面，最新代码也不生效)，可以删除此文件夹，运行时会自动生成最新的

### Servlet实现

* 实现Servlet的三种方式：一个实现，两个继承
  1. 实现Servlet接口
  2. 继承GenericServlet抽象类

重写service()方法

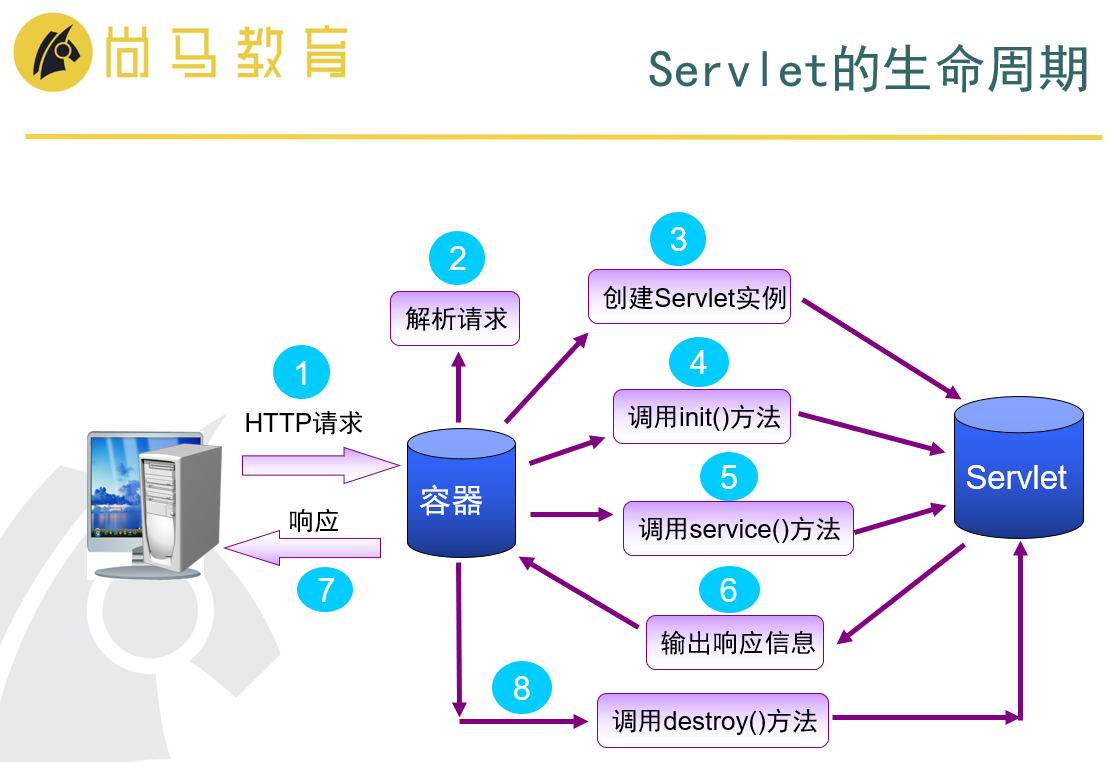
* 1. 继承HttpServlet抽象类

HttpServlet继承于GenericServlet

重写doGet()，doPost()方法

* 配置web.xml文件
* 通过<servlet>标签，告诉web容器，映射路径
* Servlet执行过程

1. 创建Servlet对象
2. 第一次访问Servlet对象时执行init()方法，该方法只执行一次service()方法
3. 之后每次访问就执行一次service()方法
4. 关闭服务器时调用destroy()方法

* Servlet的服务周期
  1. 创建
  2. 初始化
  3. 服务
  4. 销毁
* 

### Servlet常用类和方法

* Servlet三大域对象的应用
  + Request
    - 一个用户可有多个
    - 每次请求都会产生一个
    - 请求结束对象消失
    - 建议用于页面间值的传递
  + Session
    - 一个用户一个
    - 每一个浏览器都会产生一个
    - 会话过期或者关闭浏览器对象消失
    - 建议存放一些用户信息，不要把过多的信息存放在session里
  + application（ServletContext）
    - 全局只有一个
    - 是一个全局的储存信息的空间，服务器开始就存在，服务器关闭才释放。
    - 为了节省空间，提高效率，只存放少量的重要信息
* ServeltContext接口
  + ServletContext getContext(String path)
    - 获取ServletContext上下文对象
  + void setAttribute(String name,Object obj)
    - 在ServletContext对象中存值
  + Object getAttribute(String name)
    - 从ServletContext对象中取值
* ServletConfig接口
  + String getInitParameter(String name)
    - 获取<init-param>中的配置信息
    - 在<servlet>标签里设置<init-param>
    - <init-param>
    - <param-name>initParam</param-name>
    - <param-value>Hello Servlet</param-value>
    - </init-param>
  + ServletContext getServletContext()
    - 获取Servlet上下文对象
* Request
  + 请求，包含浏览器传过来的参数
  + request.setCharacterEncoding("UTF-8");输入的值设置为中文
  + request.getParameter("user\_name");获取页面参数,即html表单中name和它的值
  + request.setAttribute(String name,Object obj);存值
  + request.getAttribute(String name);取值
  + request.getRequestDispatcher("/abc/xyz.html").forward(request, response);转发
    - url地址不变
    - 可以传递request对象
    - 服务器行为
  + 获取各种路径
    - request.getContextPath(); 项目名/WebDemo
    - request.getScheme()+"://"+request.getServerName()+":"+request.getServerPort()+request.getContextPath()+"/";
      * 结果：http://localhost:8080/WebDemo/
    - request.getRemoteAddr() 客户端ip地址
    - request.getServletPath()-/student
    - request.getRealPath("/") 在tomcat中的路径
    - request.getRequestURI(); 获得请求路径/web\_demo/student
* Response
  + 响应，返回给浏览器的对象
  + response.setContentType("text/html;charset=UTF-8");输出的值设置为中文
  + response. sendRedirect ("/abc/xyz.html");重定向
    - url地址会发生改变
    - 不可以传递request对象
    - 客户端行为 相当于 客户端自动跳转了页面
* HttpSession
  + 会话对象
  + HttpSession session = request.getSession();从request对象中获取会话
  + session.invalidate();注销
  + session.setAttribute(String name,Object obj);存值
  + session.getAttribute(String name);取值
* PrintWriter
  + PrintWriter out = response.getWriter();获得输出对象，输出内容到页面
  + out.println("<html>");打印html代码到页面
  + out.close();关闭流

## JSP

* 什么是JSP
* 在HTML中嵌入Java脚本代码
* 由应用服务器中的JSP引擎来编译和执行嵌入的Java脚本代码
* 然后将生成的整个页面信息返回给客户端
* Web容器处理JSP文件请求需要经过3个阶段
* 翻译阶段
* 编译阶段
* 执行阶段
* Page指令

<%@ page 属性1="属性值" 属性2="属性值1,属性值2"… 属性n="属性值n"%>

* 小脚本

<% %>

小脚本中可以直接声明变量

* 声明

<%! %>

声明中定义方法

* 注释

<%-- --%>

* Include指令

<%@ include file="left.jsp" %>

### JSP内置对象

* 什么是的内置对象
  + 在jsp开发中会频繁使用到一些对象.如果每次我们在jsp页面中需要使用这些对象都要自己亲自动手创建就会特别的繁琐
  + SUN公司因此在设计jsp时,在jsp页面加载完毕之后自动帮开发者创建好了这些对象,开发者只需要使用相应的对象调用相应的方法即可
  + 这些系统创建好的对象就叫做内置对象.
* JSP的九大内置对象有哪些
  + 内置对象名 类型
  + request HttpServletRequest
  + response HttpServletResponse
  + config ServletConfig
  + application ServletContext
  + session HttpSession
  + exception Throwable
  + page Object(this)
  + out JspWriter
  + pageContext PageContext
* JSP的四大作用域有哪些
  + ServletContext context域
  + HttpServletRequet request域
  + HttpSession session域
  + PageContext page域

### 客户端缓存

#### Cookie

Cookie译为小型文本文件或小甜饼，Web应用程序利用Cookie在客户端缓存服务器端文件。Cookie是以键值对形式存储在客户端主机硬盘中，由服务器端发送给客户端，客户端再下一次访问服务器端时，服务器端可以获取到客户端Cookie缓存文件。

Cookie是由服务器端创建的，然后由服务器端发送给客户端，客户端以键值对形式存储Cookie，并标注Cookie的来源。客户端再次访问服务器端时，存储的Cookie会保存在请求协议中，服务器端可以获取上次存储的缓存文件内容。

Cookie的用途：

* 电子商城中购物车功能（每买一样商品，保存一个Cookie）
* 用户自动登录功能（第一次登录时，将用户名和密码存储在Cookie）

Cookie的缺点：

* 多人共用一台计算机（例如导致用户名和密码不安全等问题）。
* Cookie被删除时，利用Cookie统计用户数量出现偏差。
* 一人使用多台计算机（网站会将看成多个用户等问题）
* Cookie会被附加在每次Http请求协议中，增加流量。
* Cookie使用明文（未加密）传递的，安全性低。
* Cookie的大小限制在4KB左右，无法存储复杂需求。

Cookie规范

Http协议提供了有关Cookie的规范，现今市场上出现大量浏览器，一些浏览器对该Cookie规范进行了一些“扩展”，但Cookie缓存文件不会占满硬盘空间。

* Cookie存储的大小上限为4KB。
* 一个服务器最多在客户端浏览器中可以保存20个Cookie。
* 一个浏览器最多可以保存300个Cookie。

##### Java操作

存值：

//创建/修改cookie对象，键值对形式

Cookie cookie = new Cookie("username", "admin");

//设置过期时间 单位秒，删除Cookie只需要修改过期时间为负数

cookie.setMaxAge(60\*60);

//cookie对象存入response，响应给浏览器

response.addCookie(cookie);

取值：

//从request对象中取值

Cookie[] cookies = request.getCookies();

//循环输出值

for(Cookie cookie : cookies){

key:cookie.getName()

值：cookie.getValue()

}

##### JavaScript操作

JavaScript 可以使用 document.cookie 属性来创建 、读取、及删除 cookie。

//js创建/修改cookie

//expires设置过期时间2018年10月10日。删除只需设置时间为以前时间

document.cookie = "password=admin;expires="+new Date(2018,9,10);

//获取cookie的所有值

//document.cookie 将以字符串的方式返回所有的 cookie，

//类型格式： cookie1=value; cookie2=value; cookie3=value;

var cookies = document.cookie;

//循环输出

$.each(cookies.split(";"),function(i,c){

console.log(c.split("=")[0]);

console.log(c.split("=")[1]);

})

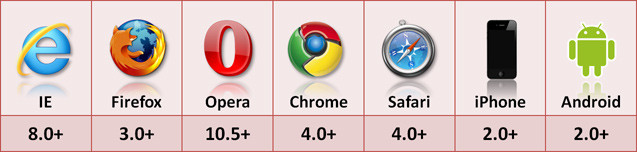
#### Web Storage

对于Web Storage来说，实际上是Cookies存储的进化版。

由于Cookie的数据由每个服务器的请求来传递，这使得cookie的速度很慢而且效率不高

HTML5 提供了两种在客户端存储数据的新方法：

* localStorage - 没有时间限制的数据存储
* sessionStorage - 针对一个 session 的数据存储，数据在浏览器关闭后自动删除



localStorage.username = "admin";//设置值

console.log(localStorage.username)//取值

localStorage.setItem("user", "user\_name");//设置值

console.log(localStorage.getItem("user"));//取值

//以上两种方式存值/取值都可以

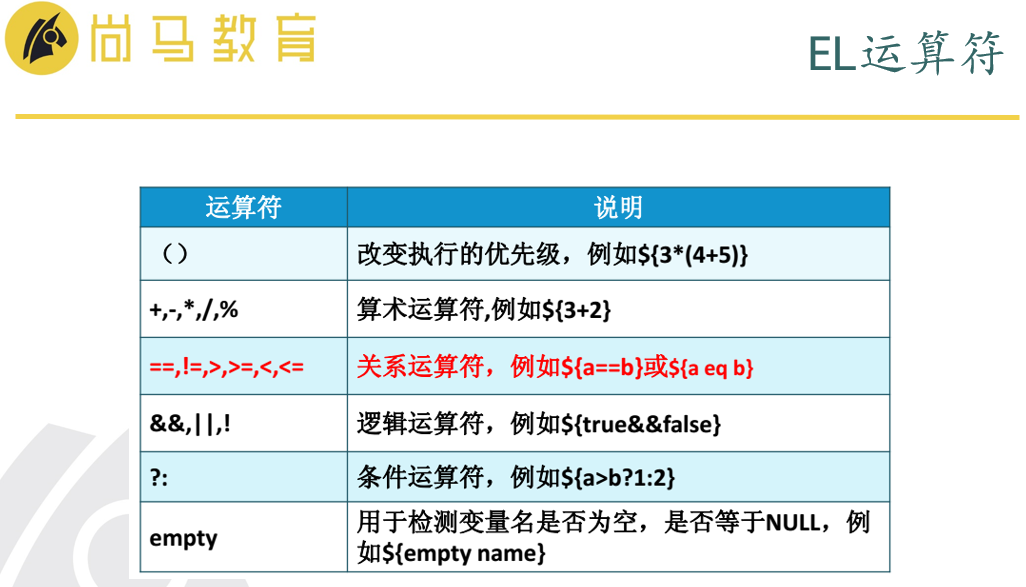
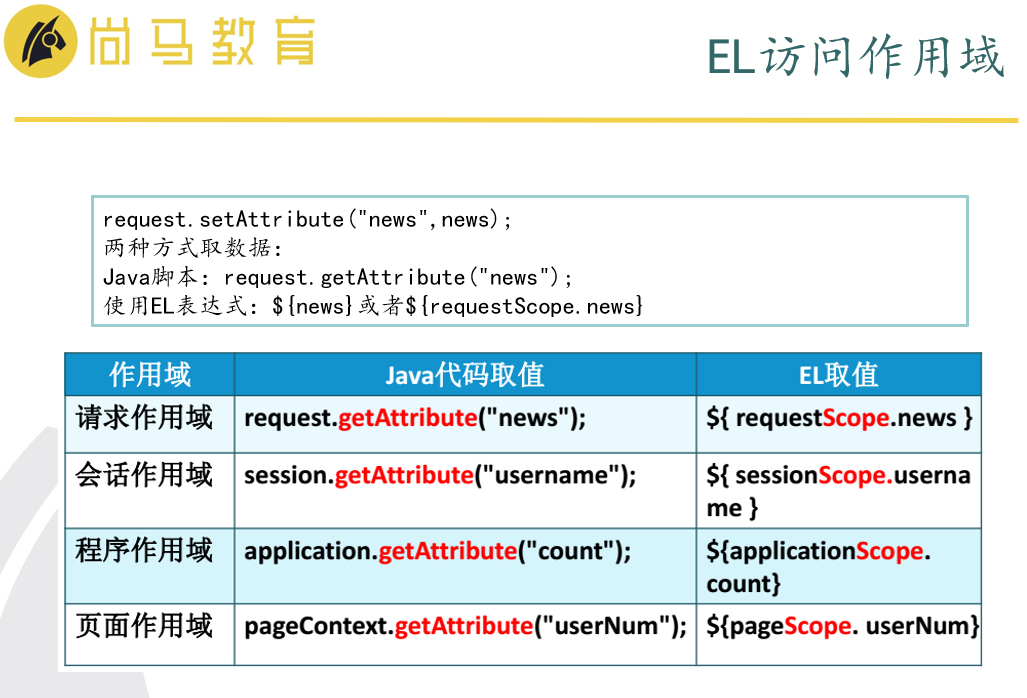
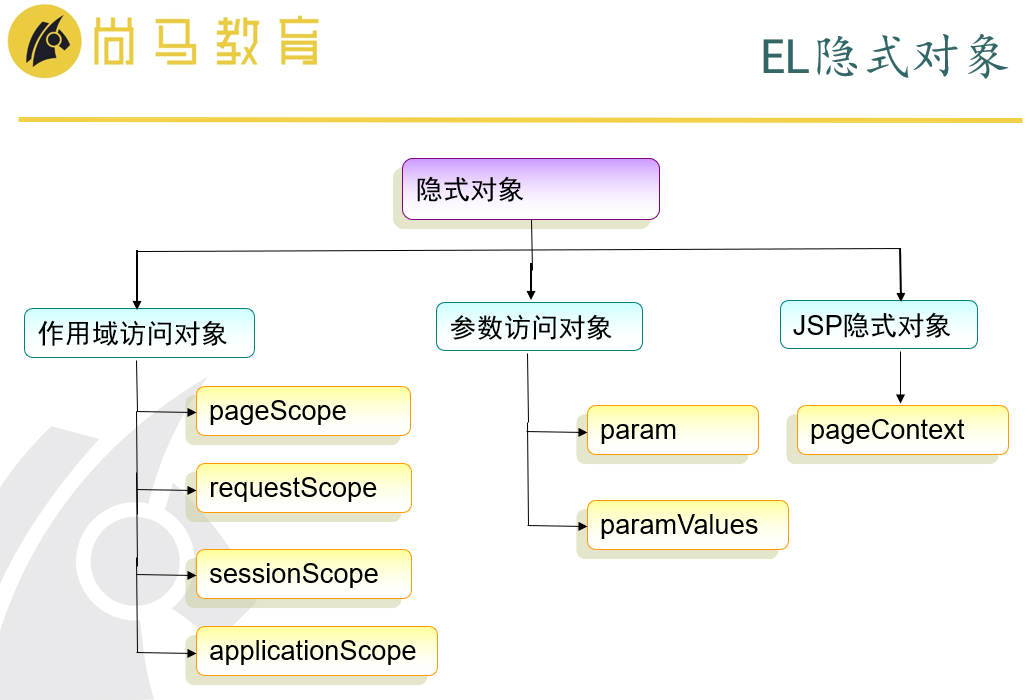
localStorage.removeItem("user");//移除指定的key

localStorage.clear();//清空所有的数据

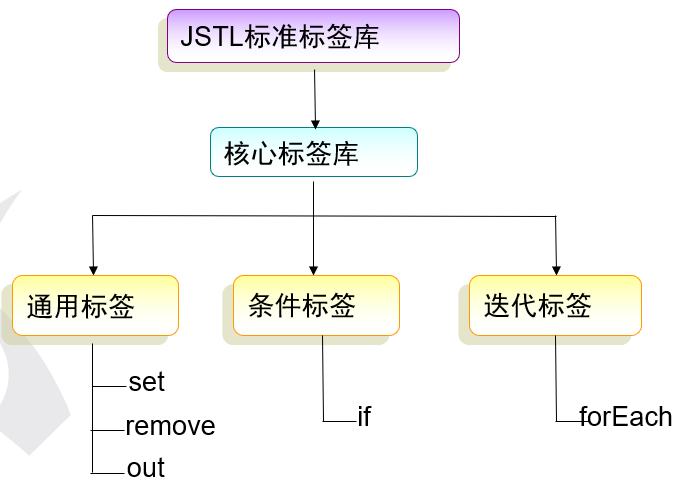
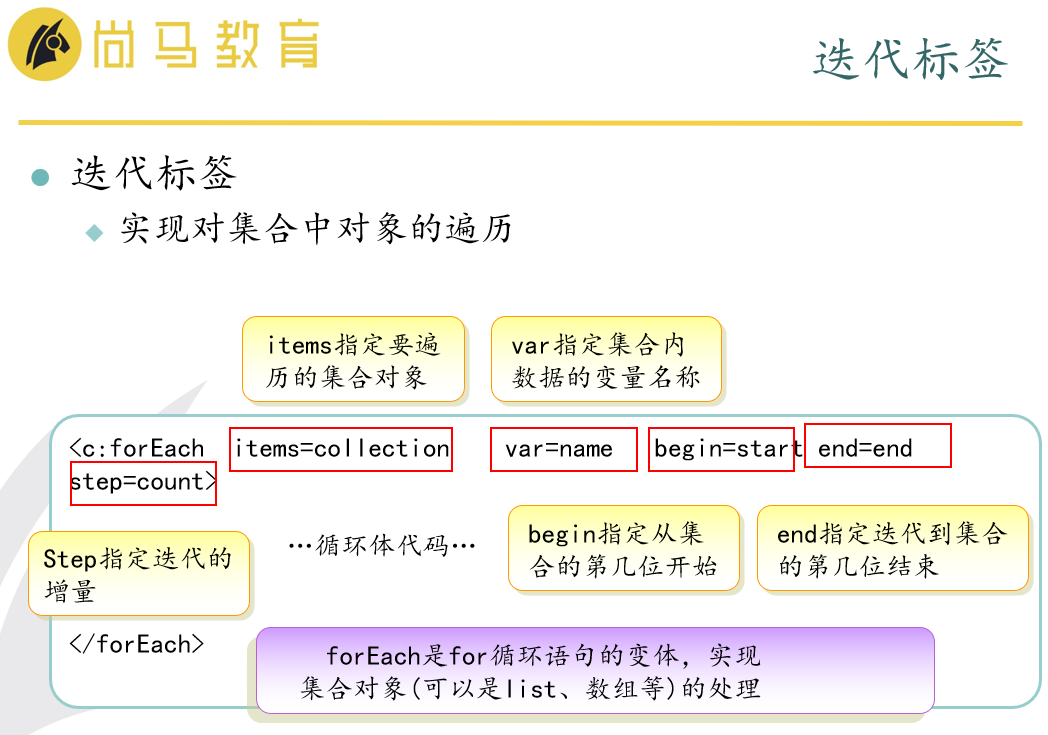
sessionStorage用法同localStorage

## EL和JSTL

### El

* 什么是EL
* EL即Expression Language（表达式语言）
* EL语法
  + 以"${"作为开始，以"}"作为结束
  + ${EL exprission}
  + ${bean.name}
* EL运算符
  + 
  + 
* EL访问作用域
  + 
* EL隐式对象
  + 
* 使用EL访问作用域
  + 请求作用域：${requestScope.news}
  + 会话作用域：${sessionScope.username}
  + 程序作用域：${applicationScope.count}
  + 页面作用域：${pageScope.userNum}

### JSTL

* 什么是JSTL
  + JSTL(JavaServerPages Standard Tag Library)
  + JSP标准标签库
  + 实现JSP页面中的逻辑控制
  + JSTL通常会与EL表达式合作实现JSP页面的编码
* 环境搭建
  + Jsp页面头部加入指令
  + <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
* JSTL标准标签库介绍
  + 
* 使用方式
  + <c:if test="${not login}">
  + <c:set var="isLogin" value="${not empty sessionScope.userId}"/>
  + 
* JSTL标签分类
  + 
* <fmt:formatDate>格式化标签
  + 显示格式化的日期和时间显示

导如格式化标签库

<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %>

语法

<fmt:formatDate value="时间对象" pattern="yyyy-MM-dd HH:mm:ss"/>

* fn函数标签库

导入函数标签库

<%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn" %>

${fn:indexOf(name,'a')} 索引位置

${fn:replace(name,'k','t')}替换字符串

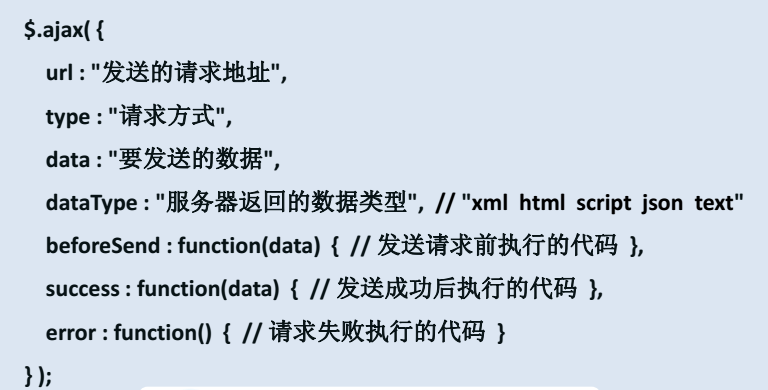
${fn:split(name,'-')}按照指定字符切分

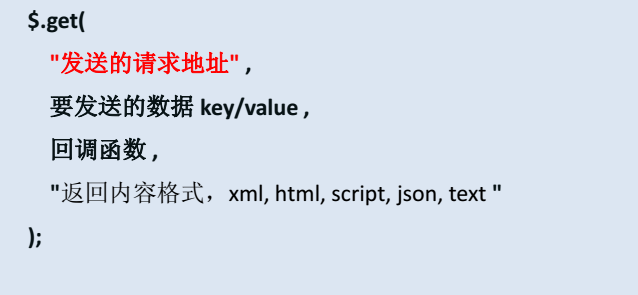
${fn:substring(name,1,5)}截取字符串

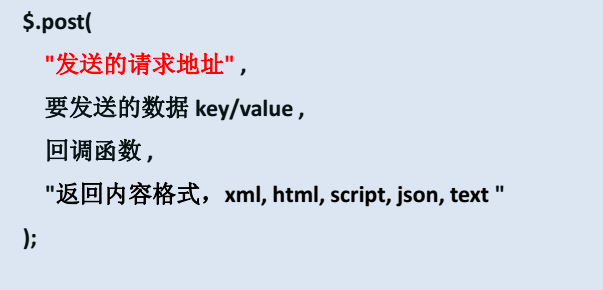
${fn:toLowerCase(name)}转化为小写

${fn:toUpperCase(name)}转化为大写

## Ajax







$.getJSON(url,{},function(data){

$.each(data,function(i){

})

});

## 分页和文件上传

### 分页

根据当前页数计算结束条数

返回总数据条数总页数等信息

### 文件上传

文件上传需要使用到apache-commons相关的jar包

commons-fileupload-1.3.3.jar

commons-io-2.5.jar

commons-lang-2.4.jar(可选)

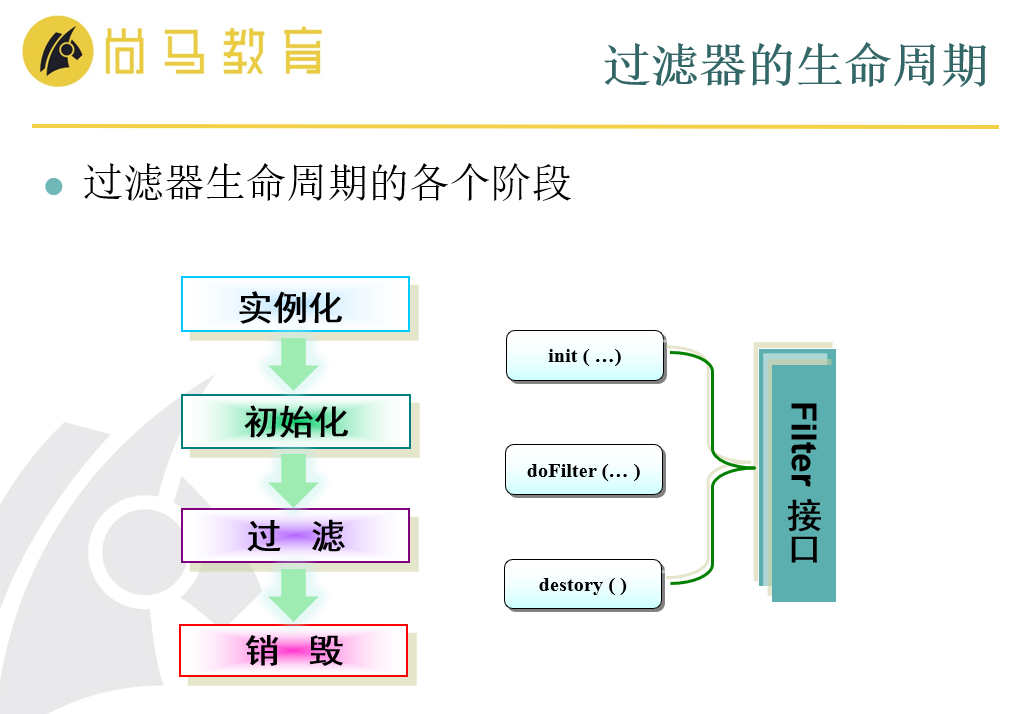






## Filter和Listener

### Filter



实现javax.servlet.Filter接口

### Listener

ServletContext监听

Session监听

Requset监听

## 数据库连接池

学会配置Druid，其他几种数据库连接池了解即可

### Druid

Jar包下载地址：<http://repo1.maven.org/maven2/com/alibaba/druid/1.1.10/druid-1.1.10.jar>

源码下载地址：<http://repo1.maven.org/maven2/com/alibaba/druid/1.1.10/druid-1.1.10-sources.jar>

帮助文档：https://github.com/alibaba/druid/wiki/首页

DruidDataSource druidDataSource = new DruidDataSource();

druidDataSource.setDriverClassName("com.mysql.jdbc.Driver");//驱动名

druidDataSource.setUrl("jdbc:mysql://192.168.2.2:3306/sqltest");//数据库地址

druidDataSource.setUsername("root");//用户名

druidDataSource.setPassword("root");//密码

//以下为非必填项，不设置也有默认值

druidDataSource.setMinIdle(1);//最小连接池数量 可设置最大值8

druidDataSource.setInitialSize(1);//初始化时建立物理连接的个数

druidDataSource.setMaxActive(10);//最大连接池数量

druidDataSource.setMaxWait(10000);//获得连接最大等待时间 单位毫秒

druidDataSource.setMinEvictableIdleTimeMillis(300000);// 配置一个连接在池中最小生存的时间，单位是毫秒

Connection conn = null; PreparedStatement ps = null; ResultSet rs = null;

try {

conn = druidDataSource.getConnection();//从数据库连接池获得连接

String sql = "select \* from t\_student";

ps = conn.prepareStatement(sql);

rs = ps.executeQuery();

while(rs.next()){

StudentModel student = new StudentModel(rs);

}

} catch (SQLException e) {

e.printStackTrace();

}finally {

try {

if (rs != null) {// 防止空指针异常rs.close();}

if (ps != null) {ps.close();}

if (conn != null) {

conn.close();//关闭的是连接，关闭后连接返回到池中

}

} catch (SQLException e) {

e.printStackTrace();

}

}

druidDataSource.close();//关闭数据库连接池，关闭后不能再使用

### c3p0

需要导入的jar包有：

* c3p0-0.9.5.2.jar

下载地址：http://central.maven.org/maven2/com/mchange/c3p0/0.9.5.2/c3p0-0.9.5.2.jar

* mchange-commons-java-0.2.11.jar

下载地址：http://central.maven.org/maven2/com/mchange/mchange-commons-java/0.2.11/mchange-commons-java-0.2.11.jar

ComboPooledDataSource cpds = new ComboPooledDataSource ();

try {

cpds.setDriverClass("com.mysql.jdbc.Driver");//加载驱动

} catch (PropertyVetoException e1) {

e1.printStackTrace();

}

cpds.setJdbcUrl("jdbc:mysql://192.168.2.2:3306/sqltest");

cpds.setUser("root");

cpds.setPassword("root");

cpds.setMinPoolSize(5); //最小连接数目

cpds.setAcquireIncrement(5); //连接不够时增加数目

cpds.setMaxPoolSize(20); //最大连接数目

Connection conn = cpds.getConnection();//从数据库连接池获得连接

//剩下的步骤与JDBC相同

### dbcp2

导入相关jar包

* commons-dbcpxxx.jar

下载地址：http://central.maven.org/maven2/org/apache/commons/commons-dbcp2/2.1.1/commons-dbcp2-2.1.1.jar

* commons-poolxxx.jar

下载地址：http://central.maven.org/maven2/org/apache/commons/commons-pool2/2.4.2/commons-pool2-2.4.2.jar

* commons-loggingxxx.jar

下载地址：http://central.maven.org/maven2/commons-logging/commons-logging/1.2/commons-logging-1.2.jar

BasicDataSource dataSource = new BasicDataSource();

dataSource.setDriverClassName("com.mysql.jdbc.Driver");

dataSource.setUrl("jdbc:mysql://192.168.2.2:3306/sqltest");

dataSource.setUsername("root");

dataSource.setPassword("root");

Connection conn = dataSource.getConnection();//从数据库连接池获得连接

### MD5加密

即Message-Digest Algorithm 5

MD5算法具有以下特点：

1、压缩性：任意长度的数据，算出的MD5值长度都是固定的。

2、容易计算：从原数据计算出MD5值很容易。

3、抗修改性：对原数据进行任何改动，哪怕只修改1个字节，所得到的MD5值都有很大区别。

4、强抗碰撞：已知原数据和其MD5值，想找到一个具有相同MD5值的数据（即伪造数据）是非常困难的。

MD5的算法不需要学习，会使用工具类即可。

public static String MD5(String key) {

char hexDigits[] = { '0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'A', 'B', 'C', 'D', 'E', 'F' };

try {

byte[] btInput = key.getBytes();

// 获得MD5摘要算法的 MessageDigest 对象

MessageDigest mdInst = MessageDigest.getInstance("MD5");

// 使用指定的字节更新摘要

mdInst.update(btInput);

// 获得密文

byte[] md = mdInst.digest();

// 把密文转换成十六进制的字符串形式

int j = md.length;

char str[] = new char[j \* 2];

int k = 0;

for (int i = 0; i < j; i++) {

byte byte0 = md[i];

str[k++] = hexDigits[byte0 >>> 4 & 0xf];

str[k++] = hexDigits[byte0 & 0xf];

}

return new String(str);

} catch (Exception e) {

return null;

}

}

## MVC

使用MVC的编程思想来编写程序

