### 1.MYSQL：

什么是数据库

数据库服务器：

安装的mysql服务器安装包，将mysql服务器软件安装在电脑上，就可以作为一台数据库服务器

数据库：

数据库服务器中的一个仓库，一个服务器可以创建多个仓库

一般一个网站的所有数据会放在一个数据库中

表：

一个数据库中可以创建多张表

通常情况下，一张表，存放一类数据

表记录：

一张表记录多条记录

数值类型:

tinyint：占用1个字节，相对于java中的byte

smallint：占用2个字节，相对于java中的short

int：占用4个字节，相对于java中的int

bigint：占用8个字节，相对于java中的long

其次是浮点类型即：float和double类型

float：4字节单精度浮点类型，相对于java中的float

double：8字节双精度浮点类型，相对于java中的double

字符串类型：

1、char类型 – 表示定长字符串类型

name char(10) char类型的范围，0~255个字符

char类型在存储数据时，如果存储的数据长度小于最大长度，不足会用空格来补全，因 此char类型可能会造成空间的浪费。

但是，char类型在存储数据的效率上比varchar类型略高一些！

2、varchar类型 – 表示不定长字符串类型

name varchar(200)

varchar类型的范围：mysql5.0之前，0~255个字符

mysql5.0之后， 0 < 字符数\*字节 <= 65535

varchar类型在存储数据时，如果存储的数据长度小于最大长度，剩余的空间会留给别 的数据使用，因此varchar类型不会有空间的浪费！！

但是，varchar类型存储数据的效率上比char类型略低一些！

日期类型：

date 年月日

time 时分秒

datetime 年月日 时分秒

timestamp 时间戳，格式上和datetime相同，都是：年月日 时分秒，设置默认从1970开始才行

datetime和timestamp二者之间的差异：

相同点：都是日期类型，显示的日期格式是相同的

不同点：

(1)二者存储的内容不同：datetime存储的就是年月日时分秒，但是，timestamp存储的一个日期到1970年1月1日之间的时间毫秒值。

(2)二者的范围不同：datetime范围是 1000~9999，timestamp的范围是 1970~2038

(3)timestamp指定的列，可以设置自动更新时间为当前时间。但是datetime则不可以。

字段约束

主键约束 primary key 一般和自增一起用auto\_increment

唯一约束 unique

非空约束 not null

外键约束：foreign key dept\_id references dept(id)

注释：

—— 注释内容 注意中间有空格

#注释内容

Mysql --default-character-set=gbk -uroot -proot //打开mysql时设置编码

show variables like 'char%'; //查看MySQL数据库使用的编码

**create database 库名 charset utf8 //创建库时指定编码**

drop database if exists 库名; //判断是否存在,都能用

在插入(写数据)前设置编码:  **set names gbk;**

insert into 表名( ) values ( ); //插入

update 表名 set 列=值, .....; //修改

delete from 表名; //删除,单id还在

修改表

alter table 表名 add 列 类型; //新增列

alter table 表名 modify 列 int primary key; //修改列

alter table 表名 drop 列; //删除列

truncate table emp; //删除表中所有记录(将表摧毁再重建),防止id的自增

select distinct dept,job from emp; //剔除重复的记录

select name,sal+ifnull(bonus,0) as 总薪资 from emp //as取别名

where sal+ifnull(bonus,0)>3500; //ifnull判断是否有空,用第二个值替换

select name,sal from emp where sal between 3000 and 4500; //在...和...之间的

select name,sal from emp where sal in (1400,1600,1800); //在集合里的

not in //不在集合中的

select \* from emp where dept is null; //判断是否为空 /is not null

select name from emp where name like '刘%'; //like模糊查询,%：0到多个字符,\_：一个字符

select name,sal from emp order by sal asc; //排序asc升序

desc降序

select \* from emp group by dept; //分组

/\* 聚合函数:

(1)max(colname)、min(colname) -- 求某一列中的最大值、最小值

(2)sum(colname) -- 统计某一列的数据之和

(3)avg(colname) -- 求某一列的平均值

(4)count(\*) -- 统计查询结果的记录行数

-- 聚合函数会自动剔除null值(即null值不会参数计算或统计)

-- 当没有分组时,聚合函数是对整个查询结果进行计算或者统计(可以将整个查询结果看作是一个组)

-- 如果进行了分组,聚合函数是对每一个组的记录进行计算或者统计

-- 聚合函数不能用在where子句中!!

\*/

select \* from emp where month(curdate())=month(birthday);

year(date)、month(date)、day(date) //只获取日期中的年\月\日

curdate() 获取当前日期 年月日

curtime() 获取当前时间 时分秒

sysdate() 获取当前日期+时间 年月日 时分秒

now() 获取当前日期+时间 年月日 时分秒

/\* order by 排序的列 asc 升序(从低到高)

order by 排序的列 desc 降序(从高到低)

默认就是升序，所以asc可以省略不写 \*/

/\* 在mysql中，通过limit进行分页查询：

limit (页码-1)\*每页显示记录数, 每页显示记录数 \*/

select \* from emp limit 0,3; 第一页

select \* from emp limit 3,3; 第二页

一般用来求前几名：如前五名 limit 5;

关联查询

select \* from dept,emp where emp.dept\_id = dept.id;

外连接查询（左left右right外连接）

select \* from dept left join emp on emp.dept\_id=dept.id;

/\* where和having都用于筛选过滤，但是：

where用于在分组之前进行筛选, having用于在分组之后进行筛选

where中不能使用列别名和聚合函数, having中可以使用别名和聚合函数

因为where子句比select先执行!!\*/

SQL语句的书写顺序：

select...

from...

where...

group by...

having...

order by...

limit...

...

SQL语句的执行顺序：

from... -- 确定要查询的是哪张表 (定义表别名)

where... -- 从整张表的数据中进行筛选过滤

select... -- 确定要显示哪些列 (定义列别名)

group by... -- 根据指定的列进行分组

having... -- 对分组之后的数据进行筛选过滤

order by... -- 根据指定的列进行排序

...

JDBC：

Java DataBase Connectivity Java数据库路连接

JDBC的六个步骤：

1. 注册数据库驱动

Class.forName(“com.mysql.jdbc.Driver”);

1. 获取数据库连接

Connection conn = DriverManager.getConnection(

“jdbc:mysql://localhost:3306/jt\_db?characterEncoding=utf-8”,

“root”,”root”);

1. 获取传输器

Statement stat = conn.createStatement();

1. 执行sql语句，返回执行结果

String sql = "select \* from account";

ResultSet rs = stat.executeQuery(sql); //执行查询操作时用

Int rows = stat.executeUpdate(sql); //执行加删改操作时用

1. 处理结果

while(rs.next()) { //使指向数据行的索引向下移动一行

int id = rs.getInt("id");

String name = rs.getString("name");

1. 释放资源

放在finally代码块中，每一个对象都有单独的一个try/catch/finally

SQL注入攻击：由于SQL语句是拼接而来的。其中的参数是由用户提交的，如果用户在提交参数时，在其中掺杂了一些SQL关键字或者特殊符号，就会跳过验证

如：加注释 aa’# 加或 a or ‘1=1

使用preparedStatement对象，是Statement的子接口，比Statement更安全

String sql = "select \* from user where username=? and password=?"; //用？占位符

preparedStatement ps = conn.prepareStatement(sql); //获取传输器，并发送sql语句骨架

ps.setString(1, user); //设置参数

ps.setString(2, pwd);

rs = ps.executeQuery(); //提交

c3p0数据库连接池

创建数据库连接池：

ComboPooledDataSource pool = new ComboPooledDataSource();

方式一：直接写

pool.setDriverClass("com.mysql.jdbc.Driver");

pool.setJdbcUrl("jdbc:mysql:///jt\_db?characterEncoding=utf-8");

pool.setUser("root");

pool.setPassword("root");

方式二：通过c3p0-config.xml文件

<?xml version="1.0" encoding="UTF-8"?>

<c3p0-config>

<default-config>

   <property name="driverClass">

      com.mysql.jdbc.Driver

    </property>

  <property name="jdbcUrl">

     jdbc:mysql:///jt\_db?useUnicode=true&amp;characterEncoding=utf-8

  </property>

  <property name="user">

     root

  </property>

  <property name="password">

     root

  </property>

</default-config>

</c3p0-config>

方式三：通过c3p0.properties文件

c3p0.driverClass=com.mysql.jdbc.Driver

c3p0.jdbcUrl=jdbc:mysql:///jt\_db

c3p0.user=root

c3p0.password=root

从连接池获取连接

conn = pool.getConnection();

释放资源

conn.close 不是关闭连接，而是被c3p0连接池封装为还连接

数据库事务(Database Transaction)，是指作为单个逻辑工作单元执行的一系列操作，要么完全地执行，要么完全地不执行。

事务的四大特性(ACID)是：

(1)原子性（Atomicity）：事务中所有操作是不可再分割的原子单位。事务中所有操作要么全部执行成功，要么全部执行失败。

(2)一致性（Consistency）：事务执行后，数据库状态与其它业务规则保持一致。如转账业务，无论事务执行成功与否，参与转账的两个账号金额之和应该是不变的。

(3)隔离性（Isolation）：隔离性是指在并发操作中，不同事务之间应该隔离开来，使每个并发中的事务不会相互干扰。也就是说，在事务中查看数据更新时，数据所处的状态要么是另一事务修改它之前的状态，要么是另一事务修改它之后的状态，事务不会查看到中间状态的数据。例如：在A事务中，查看另一B事务(正在修改张三的账户金额)中张三的账户金额，要查看到B事务之前的张三的账户金额，要么查看到B事务之后张三的账户金额。

(4)持久性（Durability）：一旦事务提交成功，事务中所有的数据操作都必须被持久化到数据库中，即使提交事务后，数据库马上崩溃，在数据库重启时，也必须能保证通过某种机制恢复数据。

Mysql中的事务：

开启事务：start transaction；

setAutoCommit(Boolean autoCommit);设置是否自动提交事务，默认为自动提交

结束事务：commit（提交事务）或rollback（回滚事务）。

事务并发读问题：多个事务对相同的数据同时进行操作，这叫做事务并发。

(1)脏读（dirty read）：读到另一个事务的未提交更新数据，即读取到了脏数据；

例如：A给B转账100元但未提交事务，在B查询后，A做了回滚操作，那么B查询到了A未提交的数据，就称之为脏读。

(2)不可重复读（unrepeatable read）：对同一记录的两次读取不一致，因为另一事务对该记录做了修改（是针对修改操作）

例如：在事务1中，前后两次查询A账户的金额，在两次查询之间，另一事物2对A账户的金额做了修改，此种情况可能会导致事务1中，前后两次查询的结果不一致。这就是不可重复度

1. 幻读（虚读）（phantom read）：对同一张表的两次查询不一致，因为另一事务插入了一条记录(是针对插入或删除操作)；

事务隔离级别：

1、READ UNCOMMITTED（读未提交数据）

安全级别最低, 可能出现任何事务并发问题(比如脏读、不可以重复读、幻读等)

性能最好（不使用!!）

2、READ COMMITTED（读已提交数据）（Oracle默认）

防止脏读，没有处理不可重复读，也没有处理幻读；

性能比REPEATABLE READ好

3、REPEATABLE READ（可重复读）（MySQL默认）

防止脏读和不可重复读，不能处理幻读问题；

性能比SERIALIZABLE好

4、SERIALIZABLE（串行化）

不会出现任何并发问题，因为它是对同一数据的访问是串行的，非并发访问的；

性能最差；

MySQL的默认隔离级别为REPEATABLE READ，即可以防止脏读和不可重复读

Mysql设置事务级别

set tx\_isolation='read-uncommitted';

### 2.HTML+CSS：

HTML：超文本标记语言

结构：

<!DOCTYPE html> <!--文档声明声明当前html的版本为5.0 -->

<html>

    <head>

<meta charset="utf-8" />

       <title>网页的标题</title>

    </head>

    <body>

       Hello CGB1808...

    </body>

</html>

标签：

开始标签和结束标签：<from></from>

自闭标签：<img /> 也可以写成上面那种

空格：&nbsp; &emsp;中文空格 换行<br/>

常用标签

图像标签：

<img src="img/1.jpg" width="70%" border="5px"/>

相对路径 宽度 边框

超链接：

<a href="http://www.baidu.com" target="\_blank">百度一下，你就不知道</a>

点击后跳转的地址 \_self：默认值，在当前窗口打开超链接

\_blank：在新的窗口中打开超链接

表格标签：

<table>

    <tr>

       <td>11</td>

    </tr>

    <tr rowspan="2">

       <td colspan="2">21</td>

    </tr>

</table>

<table> -- 用于定义一个表格

<tr> -- 用于定义表格中的行

<td> -- 用于定义表格行中的单元格

<th> -- 也是单元格, 是表头中的单元格

其中的colspan是设置单元格横跨的列数

另外rowspan是设置单元格竖跨的行数

表单标签：

<form action="#" method="GET"></form>

action 必须存在的属性，用来指定表单提交的目的地地址

method 可选属性，用来指定以何种方式来提交表单，如果不指定，默认是GET提交

提示1：在HTTP协议中规定了7种提交方式，其中5种都不常用，只用GET和POST。

提示2：只有使用表单，并且明确的指定了提交方式为POST时，才是POST提交，其他方式都是GET提交。

表单项标签：

input标签

(1)作为普通文本输入框

<input type="text" name="username"/>

(2)作为密码输入框

<input type="password" name="password"/>

(3)作为单选框

<input type="radio" name="gender" value="male" checked="checked"/>男

/\*name值相同表示为一个组，防止多选，value设置选中时提交的值，

checked设置为默认选中\*/

(4)作为复选框

<input type="checkbox" name="like" value="basketball"/>篮球

(5)提交按钮

<input type="submit" value="提交"/>

(6)普通按钮

<input type="button" value="点我"/>

select、option标签 -- 实现下拉选框

<select name=*"city"*>

    <option value=*"bj"*>北京</option> /\*改变提交时的值\*/

    <option>上海</option>

    <option selected=*"selected"*>广州</option> /\*设置为默认被选中\*/

    <option>深圳</option>

</select>

textarea标签 -- 多行文本输入框

<textarea rows=*"5"* cols=*"30"*  name=*"description"*

    placeholder=*"请输入描述信息..."*></textarea>

rows用于指定行数, clos用于指定列数，placeholder用于指定输入框中的提示消息

引用css的三种方式：

1. 通过style属性引入css
2. 通过style标签引入css
3. 通过link链接引入外部的css文件

<link href=*"demo.css"*  rel=*"stylesheet"*/>

选择器：

标签名选择器：

格式：标签名/元素名{ css属性... }

class选择器：

格式：.class值{ css属性... }

id选择器：

格式：#id{ css属性... }

后代选择器：

格式: 父选择器 后代选择器{ css样式… }

属性选择器

格式: 选择器[属性条件1][属性条件2]…{ css样式 }

### JavaScript+JQuery：

JavaScript：

1、特点

(1)JS是一门直译式的语言，直接执行源代码（是一边解释一边执行，不像Java需要提前编译为class文件再运行）

(2)JS是一门弱类型的语言

2、优势：

(1)良好的交互性

(2)一定的安全性

(3)跨平台性（有浏览器的地方就可以执行JS）

数据类型：

1. 基本数据类型

数值类型(number)

JS底层只有一种--浮点型，但是在处理或显示的时候，会在整型和浮点型之间 进行自动的转换

NaN非数值 Infinity/-Infinity 正/负无穷

字符串类型(string)

基本数据类型。但也有对应的包装对象

布尔类型(boolean)

布尔类型的值有true和false

undefined未定义

null空值

1. 复杂数据类型

对象、数组、函数（都属于对象）

var p = {

name : "张三",

age : 18

} //对象

var arr = [100, false, "abc"]; //数组

var fn = function(){ alert("aaa") } //函数

JS中是通过var关键字声明所有变量。

添加/删除元素：

var newDiv = document.createElement("元素名"); 创建一个新元素

newDiv.innerHTML = "我是新来的"; 给标签加内容

Var parent = newDiv.parentNode; 获取父元素

parent.appendChild(new); 将元素添加到父元素的最后面

parent.removeChild( new ); 删除子元素

var dateStr = new Date().toLocaleString(); //获取表示当前时间的字符串

document.body -- 获取当前文档中的body元素

JavaScript获取标签的六种方法：

document.getElementById('id名');

document.getElementsByName('name名'); 集合----数组选择[0]

document.getElementsByTagName(‘标签名’); 集合----数组选择[0]

document.getElementsByClassName('class名'); 集合----数组选择[0]

document.querrySelector('可以写任何选择器.或# 只能选择一个');

document.querrySelectorAll('可以写任何选择器.或# 可以选择多个');

JQuery：

轻量的、免费开源的JS函数库

引用库：

<script src=*"js/jquery-1.8.3.js"*></script>

文档就绪事件-----解决js使用网页元素时网页还没有加载的问题

/\* (JS提供)文档就绪事件函数\*/

window.onload = **function**(){

}

/\*jQuery提供的文档就绪事件函数\*/

$(function(){

//在整个html网页被加载完后立即执行

});

Jquery选择器：

基本选择器：同css的基本一样

层级选择器：

$("#one span") -- 匹配one内部的所有span元素

$("#one").next("span")/$("#one+span") --匹配one的下一个相邻的span兄弟元素

$("#one").nextAll("span") /$("#one~span")-- 匹配one的后面所有的span元素。

$("#one").prev("span")--匹配one的元素前一个相邻的span兄弟元素

$("#one").prevAll("span")--匹配one的前的所有span元素

$("#one").siblings("span")--匹配one的前/后面所有的span元素

基本过滤选择器：类似数组筛选

$("div:eq(0)")

$("div").eq(0)

$("div:first") -- 以上都用于匹配所有div中的第一个div元素

$("div:eq(-1)")

$("div:last")

var len = $("div").length;

$("div:eq("+(len-1)+")") -- 以上都用于匹配所有div中的最后一个div元素

直接写在元素上的属性称之为元素属性, 例如:

type/name/id/class/checked/value...

用于设置css样式的属性叫做css属性, 例如:

width/height/background/font-size/color...

创建元素：$("<div></div>")

添加子元素：$parent.append( $child )

删除元素：$ele.remove()

替换元素：$ele.replaceWith( $other )

html()函数 (类似于js中的innerHTML属性)

$("#div1").html();-- 获取id为div1元素的所有内容

text()函数 (类似于js中的innerText属性)

$("#div1").text() -- 获取id为div1元素内部的文本

val()函数 (类似于js中的value属性)

$("#user").val() -- 获取用户名输入框中的值

prop()函数：用于获取或设置元素的属性值

$("#user").prop("name")

css()函数： 获取或设置css属性值

$("#div1").css("width")

show() 函数-- 设置元素由隐藏变为显示 等价于 css("display","block")

$("div").show() -- 设置所有的div元素为显示

hide()函数 -- 设置元素由显示变为隐藏 等价于 css("display","none")

$("div").hide() -- 设置所有的div元素为隐藏

toggle() -- 切换元素的显示状态, 如果元素是显示的, 则切换为隐藏, 否则切换为显示

slidToggle() --切换元素的显示状态, 如果元素是显示的, 则切换为隐藏, 否则切换为显示, 切换为显示为下拉状态, 隐藏为收缩状态.

### Tomcat：

端口占用问题：

cmd中netstat -ano 地址:端口—》找到进程编号PID

taskkill /f /pid 进程编号 ---结束进程

虚拟主机：是tomcat服务器中配置的一个站点

1.先在server.xml中添加Host标签

image080

2.在c:/windows/system32/drivers/etc/hosts文件中添加主机和ip地址映射

127.0.0.1      [www.baidu123.com](http://www.baidu123.com/) #可以自己使用

192.168.31.2  [www.baidu123.com](http://www.baidu123.com) #可以别人使用

3.清除浏览器的缓存(ctrl+shift+delete), 重启服务器, 重启浏览器, 访问 www.baidu123.com/....

web应用：

news(Web应用/目录)

|-- 其他目录：放在除WEB-INF以外的目录下的资源文件，通过浏览器都可以访问。

|-- WEB-INF目录: 放在WEB-INF中的资源文件是受保护的，通过浏览器不能直接去访问。

|-- classes: 用于存放编译后的class文件

|-- lib: 用于存放web应用所依赖的jar包

|-- web.xml: 是web应用的核心配置文件

Tomcat>>虚拟主机>>web应用>>web资源

### HTTP协议详解：

http请求：

1、请求行---------GET /news/hello.html HTTP/1.1

GET: 表示请求的方式, HTTP协议中一共规定了7种提交方式, 其中5种都不常用, 只用GET和POST提交。

/news/hello.html：请求资源路径

HTTP/1.1:请求所遵循的协议和版本。

2、若干请求头、请求报头

Key-Value结构,方便服务器解析

例如:Host: localhost -- 通知服务器浏览器访问的是哪一个主机

Cookie: 将服务器之前发送给浏览器的cookie信息再次带回服务器中。

3、请求实体内容

如果请求方式是GET提交，请求实体内容是空的。

如果请求方式是POST提交，并且请求中携带了数据，此时请求实体才会有内容。

http响应：

1、状态行HTTP/1.1 200 OK

(1)HTTP/1.1 -- 响应所遵循的协议和版本

(2)200: 状态码(三位的数字)，作用是表示服务器对请求处理的结果。

200：表示请求处理成功。

304/307: 表示通知浏览器使用缓存!

404：表示客户端请求的资源不存在（问题出在客户端）

500：表示服务器端程序在处理请求的过程中抛出了异常。

(3)OK: 描述短语, 作用和200一致,表示请求处理的结果.

2、若干响应头

Content-Type: text/html -- 通知浏览器响应的数据类型

text/html: 表示响应的是html格式的网页

image/jpeg:表示响应的是jpg格式的图片

Content-Length: 响应数据的长度。

Set-Cookie: 和cookie技术相关的头

Refresh: 5;url=http://www.baidu.com/index.html

3、响应实体内容

响应实体中就是浏览器所请求文件的内容

例如: 浏览器向服务器请求一个xxx.html文件，服务器会将xxx.html文件的内容通过响应实体发送给浏览器。

### Servlet：

对服务器接收到的请求进行处理!!

Servlet继承关系：

Servlet接口 -- 定义了一个Servlet应该具备哪些功能 5个方法

|-implements

|-- GenericServlet抽象类，实现了Servlet接口，但没实现核心的处理请求方法 service，

|-extends

|-- HttpServlet抽象类，继承了GenericServlet，实现了service方法，判 断根据不同的请求方式，调用不同的doXxx方法。

自己的servlet继承HttpServlet，重写父类中的doGet方法和doPost方法。

request请求和response响应：

request.setCharacterEncoding("utf-8");--解决获取post提交时的参数的中文乱码

request.getParameter(“username”);-- 获取单个参数值

request.getParameterValues(“likes”);-- 获取所有参数值，返回数组

response.setContentType("text/html;charset=utf-8");--处理响应正文乱码

Request请求转发：

浏览器—>服务器—>JSP—>浏览器

一次请求、一次响应

request.getRequestDispatcher("转发到资源的路径").forward(req, res);

只能在Web应用内部的两个资源之间进行跳转

传递数据使用

域对象：

域对象的四个范围

PageContext request session application

一个jsp 一次请求 一次会话 一个web

request.setAttribute方法存数据

request.getAttribute方法获取数据

request域对象所具备的三大特征:

生命周期： 一次请求开始时，创建request对象，一次请求结束时（响应完成了），销毁request对象

作用范围： 一次请求范围内

主要功能： 和转发配合使用，带数据到目的地

Response重定向：

response.sendRedirect("重定向到资源的地址");

(1)重定向前后是两次请求、两次响应。

(2)重定向前后地址栏地址会发生变化

(3)重定向既可以在同一个Web应用内部的两个资源之间进行跳转，也可以在不同的Web 应用之间或者不同的服务器之间进行跳转！

### JSP、EL、JSTL：

动态web资源技术

JSP执行过程

浏览器发请求访问服务器中的index.jsp --> 根据访问路径寻找index.jsp --> 服务器把 index.jsp翻译为一个Servlet程序(xx.java) --> 编译为 xx.class --> 执行 xx.class程序 --> 最终向浏览器输出一个html网页。  
JSP表达式

格式: <%= 常量/变量/表达式 %>

JSP脚本片段

格式: <% 若干Java语句 %>

JSP指令

格式: <%@ 指令名称 若干属性声明... %>

JSP标签技术：减少java代码

El表达式：${ 表达式/常量/变量 }

用于从四大作用域中获取数据

也可以将常量的值/表达式的计算结果输出。

寻找时的顺序是按照域从小到大的去寻找: pageContext < request < session < application

找到就返回value值，找不到不会返回null，而是什么都不输出

${ name1 }，数组${ ns[0] }，map集合${ map1.name }，javaBean对象${ user.name }

JSTL标签库：

导入包：

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

<c:set></c:set> -- 用于往域中添加属性或者修改域中已有的属性值

<c:set var=*"name1"*  value=*"张三"*  scope=*"request"*/>

Var：存入的名称 Value：存入的值 Scope：存入的域

<c:if></c:if> -- 构造if…else…语句

<c:if test="${ score>=60 && score <80 }">中等!</c:if>

Test：判断的条件

<c:forEach></c:forEach>-- 对集合或数组等元素循环遍历

<c:forEach items=*"*${ list }*"* var=*"u"* varStatus=*"status"*>

    ${ status.count }, ${ u.name }, ${ u.age }, ${ u.addr }<br/>

</c:forEach>

(1)items: 指定需要遍历的集合或数组

(2)var: 指定用于接收遍历过程中的元素

(3)begin: 指定循环从哪儿开始

(4)end: 指定循环到哪儿结束

(5)step: 指定循环时的步长, 默认值是1

(6)varStatus: 用于表示循环遍历状态信息的对象, 常用属性有:

first属性: 表示当前遍历是否是第一次, 若是, 则返回true;

last属性: 表示当前遍历是否是最后一次, 若是, 则返回true;

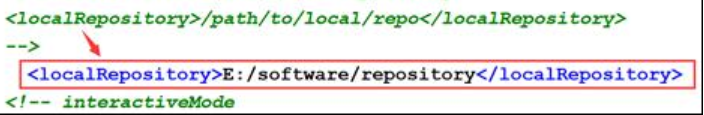
count属性: 记录当前遍历是第几次

### Maven：

项目管理工具：对项目中所有依赖的jar包进行规范化管理

官方下载地址：<http://maven.apache.org/download.cgi>

/conf/settings.xml修改maven仓库



配置镜像服务器

**配置阿里云镜像仓库：**

<mirror>

    <id>nexus-aliyun</id>

    <name>Nexus aliyun</name>

    <mirrorOf>central</mirrorOf>

    <url>http://maven.aliyun.com/nexus/content/groups/public/</url>

</mirror>

配置JDK版本

<profile>

    <id>development</id>

    <activation>

       <jdk>1.8</jdk>

       <activeByDefault>true</activeByDefault>

    </activation>

    <properties>

       <maven.compiler.source>1.8</maven.compiler.source>

       <maven.compiler.target>1.8</maven.compiler.target>

       <maven.compiler.compilerVersion>1.8</maven.compiler.compilerVersion>

    </properties>

</profile>

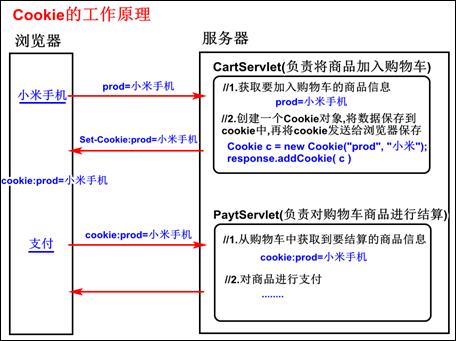
中央仓库添加依赖

[http://mvnrepository.com](http://mvnrepository.com/)

### 会话---Cookie、Session：

Cookie：将会话产生的数据保存在客户端

工作原理和API：



创建Cookie对象

Cookie = new Cookie(String name, String value);

将Cookie添加到response响应中

response.addCookie( Cookie c );

获取请求中的所有cookie对象组成的数组

Cookie[] cs = request.getCookies();

设置cookie的存活时间：如果不设置默认是会话级，随浏览器关闭而销毁

cookie.setMaxAge(60\*60\*24\*30); //存活30天，以文件保存在浏览器文件 夹中

删除cookie

通过向浏览器发送一个同名的cookie覆盖掉之前的，并设置生存时间为零

Cookie c = new Cookie("cart", "");

c.setMaxAge( 0 );

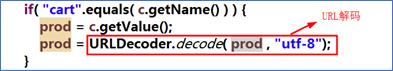
response.addCookie( c );

Cookie中保存中文数据的问题：

因为HTTP协议中规定了浏览器和服务器通信时，不能传输中文数据

Tomcat8.5之前没有自动编码和解码

image266



Cookie常用方法

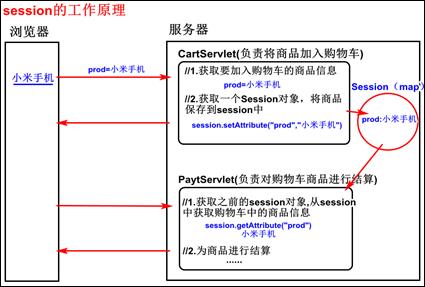
cookie.getName() -- 获取cookie的名字

cookie.getValue() -- 获取cookie中保存的值(数据)

cookie.setValue() -- 设置cookie中保存的值(数据)

Session：将会话产生的数据保存在服务端

是一个域对象



session.setAttribute(String name, Object value)

-- 往session中添加一个域属性, 属性名只能是字符串, 属性值可以是任意类型 的Object。

session.getAttribute(String name)

-- 通过属性名获取域中的属性值。返回值是一个Object。

获取一个session对象

HttpSession session = request.getSession();

request.getSession(true)

-- 如果服务器内部有当前浏览器对应的session，就直接返回session对象；如果服务器内部没有当前浏览器对应的session，就创建一个session并返回。

request.getSession(false)

-- 如果服务器内部有当前浏览器对应的session，就直接返回session对象；如果服务器内部没有当前浏览器对应的session，就直接返回null

超时销毁：30分钟未访问销毁

自杀：调用Session.invalidate()

意外身亡：服务器非正常关闭而销毁

浏览器关闭后session获取不到解决：

因为session的标识id是通过一个名称为JSESSIONID的cookie发送给浏览器识 别的，浏览器关闭cookie销毁所以session就丢了，

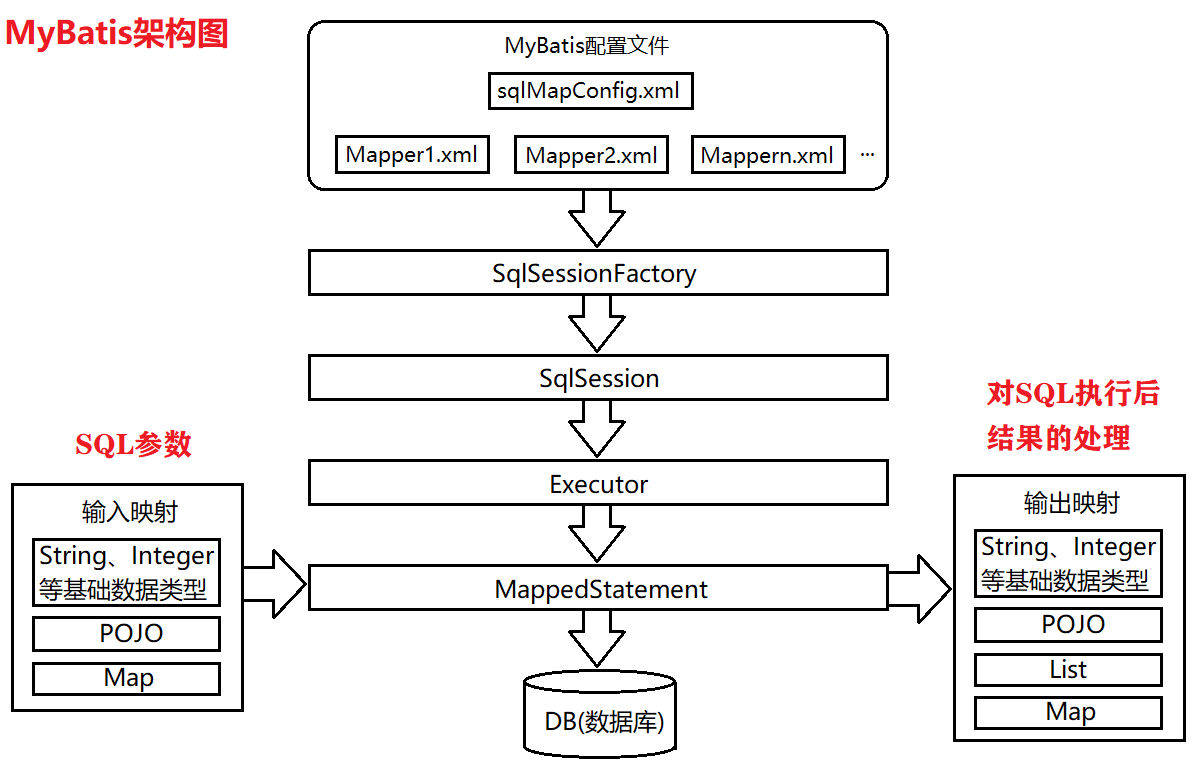
创建一个JSESSIONID的cookie设置存活时间

Cookie c = new Cookie("JSESSIONID", session.getId());

c.setMaxAge(60\*60\*24\*30);

response.addCookie(c);

### MyBatis：



//1.读取mybatis核心配置文件(mybatis-config.xml)

InputStream in = Resources.getResourceAsStream("mybatis-config.xml");

//2.通过配置获取一个SqlSessionFactory对象

SqlSessionFactory fac = new SqlSessionFactoryBuilder().build(in);

//3.通过工厂获取一个SqlSession对象, 类似于JDBC中的Connection

SqlSession session = fac.openSession();

//4.执行sql语句(XxxMapper.id)，返回处理后的结果

List<Emp> list = session.selectList("EmpMapper.findAll");