# Day01.02 MySQL数据库

## 数据库概述

### 什么是数据库？

数据库：英文为Database，简称DB

数据库是按照数据结构来组织、存储和管理数据的仓库，简而言之，数据库就是存储数据的仓库。

数据库是一个专业存储和管理数据的软件系统，相比传统人工记录数据和直接使用文件保存数据，数据库具有更安全、更可靠、效率更高的优势。

\*\*扩展内容1\*\*：数据库有哪些分类？(了解)

早期：层次式数据库、网络型数据库(已过时)

现在：关系型数据库、非关系型数据库

### 什么是关系型数据库？

底层以二维表的形式保存数据的数据库，就是关系型数据库，例如：下面有一个学生表。

|  |  |  |
| --- | --- | --- |
| 编号 | 姓名 | 年龄 |
| 1 | 张三 | 20 |
| 2 | 李四 | 22 |

\*\*扩展内容2\*\*：常见的关系型数据库有哪些？(了解)

SQL Server：微软提供，适用于中型、大型项目，收费，在java中的使用占比不高，在.NET语言中使用较多。

Oracle：甲骨文公司提供，适用于大型、超大型项目，功能强大，性能优异，收费，在Java中使用占比很高。

mysql：瑞典MySQLAB公司提供，适用于小型、中型项目，免费开源，小巧轻量，性能也不差。在Java中使用占比较高。

DB2：IBM公司提供，适用于大型项目，收费，在Java中使用占比不高。

Sqlite：迷你，嵌入式设备，智能家居，手机，ipad等

\*\*扩展内容3\*\*：什么是非关系型数据库？(了解)

非关系数据库的底层结构是以Key-Value、列等结构存储数据的

常见的非关系型数据库有：redis、mongodb

### 数据库相关概念

**1、什么是数据库服务器**

服务器软件，mysql 软件，将服务器软件装在电脑上，就可以作为一台服务器对外提供服务器。（存取数据）

**2、什么是数据库**

在每一个数据库服务器中，可以有很多个仓库（数据库），通常情况下，一个网站中的所有数据会存放在一个数据库中。

京东 db\_jd数据库

百度 db\_baidu数据库

淘宝 db\_taobao数据库

...

**3、什么是表**

一个数据库中可以创建多张表，而一张表用于存放一类信息。

（Java中的类对应数据库中的表）

商品信息 tb\_product表

购物车信息 tb\_cart表

用户信息 tb\_user表

订单信息 tb\_order表

...

**4、什么表记录**

一张表中可以包含多条表记录，一个表记录用于存放某一条具体的信息



### 什么是SQL语言？

SQL：(Structured Query Language) 结构化查询语言

SQL是一种ANSI（美国国家标准化组织） 的标准计算机语言，用于访问和操作数据库。

SQL是用于操作关系型数据库的通用的语言，即通过SQL语言可以操作所有的关系型数据库。

SQL语言可以用于操作数据库（创建、删除、查看）、操作数据表（创建、删除、修改、查看）、操作表记录（新增、更新、删除、查询）等。

## 连接mysql服务器

通过命令行工具可以登录MySQL客户端，连接MySQL服务器，从而访问服务器中的数据。

**1、连接mysql服务器**：

mysql -uroot -proot

**-u：**后面的root是用户名，这里使用的是超级管理员root；

**-p：(小写的p)**后面的root是密码，这是在安装MySQL时就已经指定的密码；

**2、连接mysql服务器并指定IP和端口**:

mysql -uroot -proot **-h**127.0.0.1 **-P**3306

**-h：**后面给出的127.0.0.1是服务器主机名或ip地址，可以省略的，默认连接本机；

**-P：(大写的P)**后面的3306是连接端口，可以省略，默认连接3306端口；

**3、退出客户端命令：quit或exit或 \q**

**4、FAQ：常见问题:**

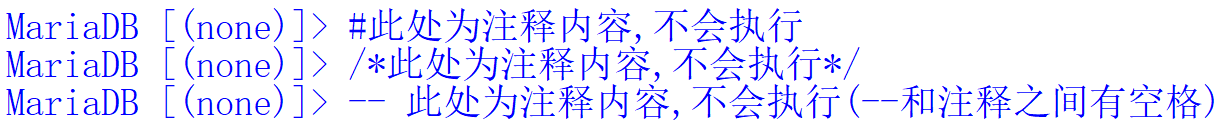


解决方法：复制mysql安装目录下的bin目录，将bin目录的路径添加到path环境变量中!!

**\*\*扩展内容4\*\*：**

(1)在cmd中连接mysql服务器之后，可以使用 #、/\*\*/、-- 等符号添加注释

例如:



(2)在cmd中连接mysql服务器之后，在书写SQL语句时，可以通过 \c 取消当前语句的执行。例如：



## 数据库及表操作

### 创建、删除、查看数据库

提示： (1)SQL 语句对大小写不敏感。推荐关键字使用大写，自定义的名称（库名，表名，列名等）使用小写。

(2)并且在自定义名称时，针对多个单词不要使用驼峰命名，而是使用下划线连接。(例如：tab\_name，而不是 tabName)

-- 01.查看mysql服务器中所有数据库

-- 02.进入某一数据库（进入数据库后，才能操作库中的表和表记录）

-- 语法：USE 库名;

-- 查看已进入的库

SELECT DATABASE();

-- 03.查看当前数据库中的所有表

-- 04.删除mydb1库

-- 语法：DROP DATABASE 库名;

-- 思考：当删除的表不存在时，如何避免错误产生?

-- 05.重新创建mydb1库，指定编码为utf8

-- 语法：CREATE DATABASE 库名 CHARSET 编码;

-- 如果不存在则创建mydb1;

-- 06.查看建库时的语句（并验证数据库库使用的编码）

-- 语法：SHOW CREATE DATABASE 库名;

### 创建、删除、查看表

-- 07.进入mydb1库，删除stu学生表(如果存在)

-- 语法：DROP TABLE 表名;

-- 08.创建stu学生表（编号[数值类型]、姓名、性别、出生年月、考试成绩[浮点型]），建表的语法：

CREATE TABLE 表名(

列名 数据类型,

列名 数据类型,

...

);

-- 09.查看stu学生表结构

-- 语法：desc 表名

## 新增、更新、删除表记录

-- 10.往学生表(stu)中插入记录(数据)

-- 语法：INSERT INTO 表名(列名1,列名2,列名3...) VALUES(值1,值2,值3...);

提示：

(1)当为所有列插入值时，可以省写列名，但值的个数和顺序必须和声明时列的个数和顺序保持一致！

(2)SQL语句中的值为字符串或日期时，值的两边要加上单引号（有的版本的数据库双引号也可以，但推荐使用单引号）。

(3)在插入数据之前，先设置编码设置编码：set names gbk;

或者用一下命令连接mysql服务器：

mysql --default-character-set=gbk -uroot -proot

-- 11.查询stu表所有学生的信息

-- 语法：SELECT 列名 | \* FROM 表名

-- 12.修改stu表中所有学生的成绩，加10分特长分

-- 修改语法: UPDATE 表名 SET 列=值,列=值,列=值...[WHERE子句];

-- 13.修改stu表中编号为1的学生成绩，将成绩改为83分。

提示：where子句用于对记录进行筛选过滤，保留符合条件的记录，将不符合条件的记录剔除。

-- 14.删除stu表中所有的记录

-- 删除记录语法: DELETE FROM 表名 [where条件]

-- 仅删除符合条件的

## 查询表记录

**-- 准备数据： 以下练习将使用db10库中的表及表记录，请先进入db10数据库!!!**

### 基础查询

SELECT 语句用于从表中选取数据。结果被存储在一个结果表中（称为结果集）。

语法：SELECT 列名称 | \* FROM 表名

提示：(1) \*（星号）为通配符，表示查询所有列。

(2)但使用 \*（星号）有时会把不必要的列也查出来了，并且效率不如直接指定列名

-- 15.查询emp表中的所有员工，显示姓名，薪资，奖金

-- 16.查询emp表中的所有部门和职位

思考：如果查询的结果中，存在大量重复的记录，如何剔除重复记录，只保留一条？ \*/

-- 在select之后、列名之前，使用DISTINCT 剔除重复的记录

### WHERE子句查询

WHERE子句查询语法：SELECT 列名称 FROM 表名称 WHERE 列 运算符 值

下面的运算符可在 WHERE 子句中使用：



-- 17.查询emp表中薪资大于3000的所有员工，显示员工姓名、薪资

-- 18.查询emp表中总薪资(薪资+奖金)大于3500的所有员工，显示员工姓名、总薪资

-- ifnull(列, 值)函数: 判断指定的列是否包含null值，如果有null值，用第二个值替换null值

-- 注意查看上面查询结果中的表头，如何将表头中的 sal+bonus 修改为 "总薪资"

-- 19.查询emp表中薪资在3000和4500之间的员工，显示员工姓名和薪资

-- 提示: between...and... 在...之间

-- 20.查询emp表中薪资为 1400、1600、1800的员工，显示员工姓名和薪资

-- 21.查询薪资不为1400、1600、1800的员工

-- 22.查询emp表中薪资大于4000和薪资小于2000的员工，显示员工姓名、薪资。

-- 23.查询emp表中薪资大于3000并且奖金小于600的员工，显示员工姓名、薪资、奖金。

-- 处理null值

-- 24.查询没有部门的员工（即部门列为null值）

-- 思考：如何查询有部门的员工（即部门列不为null值）

### 模糊查询

LIKE 操作符用于在 WHERE 子句中搜索列中的指定模式。

可以和通配符（%、\_）配合使用，其中"%"表示0或多个任意的字符。"\_"表示一个任意的字符。

语法：SELECT 列 | \* FROM 表名 WHERE 列名 LIKE 值

示例：

-- 25.查询emp表中姓名中包含"涛"字的员工，显示员工姓名。

-- 26.查询emp表中姓名中以"刘"字开头的员工，显示员工姓名。

-- 27.查询emp表中姓名以"刘"开头，并且姓名为两个字的员工，显示员工姓名。

### 多行函数查询

多行函数也叫做聚合(聚集)函数，根据某一列或所有列进行统计。

常见的多行函数有：

COUNT( 列名|\* )：统计结果集中某一列或记录行的行数。

MAX( 列 )：统计结果集中某一列值中的最大值

MIN( 列 )：统计结果集中某一列值中的最小值

SUM( 列 )：统计结果集中某一列所有值的和

AVG( 列 )：统计结果集中某一列值的平均值

提示：多行函数不能用在where子句中

-- 28.统计emp表中薪资大于3000的员工个数）

-- 29.求emp表中的最高薪资

-- 30.统计emp表中所有员工的薪资总和(不包含奖金)

-- 31.统计emp表员工的平均薪资(不包含奖金)

### 分组查询

GROUP BY 语句根据一个或多个列对结果集进行分组。

在分组的列上我们可以使用 COUNT，SUM，AVG，MAX，MIN等函数。

语法：SELECT 列 | \* FROM 表名 [WHERE子句] GROUP BY 列;

-- 32.对emp表按照部门对员工进行分组，查看分组后效果。

-- 33.对emp表按照职位进行分组，并统计每个职位的人数，显示职位和对应人数

-- 34.对emp表按照部门进行分组，求每个部门的最高薪资(不包含奖金)，显示部门名称和最高薪资

### 排序查询

使用 ORDER BY 子句将结果集根据指定的列排序后再返回

语法：SELECT 列名 FROM 表名 ORDER BY 列名 [ASC|DESC]

ASC(默认)升序，即从低到高；DESC 降序，即从高到低。

-- 35.对emp表中所有员工的薪资进行升序(从低到高)排序，显示员工姓名、薪资。

-- 36.对emp表中所有员工奖金进行降序(从高到低)排序，显示员工姓名、奖金。

### 分页查询

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

-- 37.查询emp表中的所有记录，分页显示：每页显示3条记录，返回第 1 页。

-- 38.查询emp表中的所有记录，分页显示：每页显示3条记录，返回第 2 页。

### 其他函数

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

year(date) 返回date中的年份

month(date) 返回date中的月份

day(date) 返回date中的月份

hour(date) 返回date中的小时

minute(date) 返回date中的分钟

second(date) 返回date中的秒

CONCAT(s1,s2..) 将s1,s2 等多个字符串合并为一个字符串

CONCAT\_WS(x,s1,s2..) 同CONCAT(s1,s2,..) 函数，但是每个字符串之间要加上x，x是分隔符

-- 39.查询emp表中所有在1993和1995年之间出生的员工，显示姓名、出生日期。

-- 40.查询emp表中本月过生日的所有员工

-- 41.查询emp表中员工的姓名和薪资（薪资格式为: xxx(元) ）

## mysql的数据类型

### 数值类型

MySQL中支持多种整型，其实很大程度上是相同的，只是存储值的大小范围不同而已。

**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(n)** 定长字符串，最长255个字符。n表示字符数，例如：

-- 创建user表，指定用户名为char类型，字符长度不超过10

create table user(

username **char(10)**,

...

);

所谓的定长，是当插入的值长度小于指定的长度时，剩余的空间会用空格填充。（这样会浪费空间）

**2、varchar(n)** 变长字符串，最长不超过 65535个字节，n表示字符数，一般超过255个字节，会使用text类型，例如：

-- 创建user表，指定用户名为varchar类型，长度不超过10

create table user(

username **varchar(10)**

);

所谓的不定长，是当插入的值长度小于指定的长度时，剩余的空间可以留给别的数据使用。（节省空间）

**3、大文本（长文本）类型**

最长65535个字节，一般超过255个字符列的会使用text。

-- 创建user表：

create table user(

resume **text**

);

另，text也分多种，其中bigtext存储数据的长度约为4GB。

\*\*扩展内容3\*\*：(面试题)char(n)、varchar(n)、text都可以表示字符串类型，其区别在于：

(1)char(n)在保存数据时, 如果存入的字符串长度小于指定的长度n，后面会用空格补全，因此可能会造成空间浪费，但是char类型的存储速度较varchar和text快。

因此char类型适合存储长度固定的数据，这样就不会有空间浪费，存储效率比后两者还快！

(2)varchar(n)保存数据时, 按数据的真实长度存储, 剩余的空间可以留给别的数据用，因此varchar不会浪费空间。

因此varchar适合存储长度不固定的数据，这样不会有空间的浪费。

(3)text是大文本类型，一般文本长度超过255个字符，就会使用text类型存储。

### 12日期类型

**date**：年月日

**time**：时分秒

**datetim**e：年月日 时分秒

**timestamp**：时间戳(实际存储的是一个时间毫秒值)，与datetime存储日期格式相同。两者的区别是:

(1)timestamp最大表示2038年，而datetime范围是1000~9999

(2)timestamp在插入数、修改数据时，可以自动更新成系统当前时间(后面用到时再做讲解)

## mysql的字段约束

### 主键约束

主键约束：如果为一个列添加了主键约束，那么这个列就是主键，主键的特点是唯一且不能为空。

添加主键约束，例如将id设置为主键：

create table stu(

id int primary key,

...

);

### 非空约束

非空约束：如果为一个列添加了非空约束，那么这个列的值就不能为空，但可以重复。

添加非空约束，例如为password添加非空约束：

create table user(

password varchar(50) not null,

...

);

### 唯一约束

唯一约束：如果为一个列添加了唯一约束，那么这个列的值就必须是唯一的（即不能重复），但可以为空。

添加唯一约束，例如为username添加唯一约束及非空约束：

create table user(

username varchar(50) unique not null,

...

);

### 外键约束

外键其实就是用于通知数据库两张表数据之间对应关系的这样一个列。

这样数据库就会帮我们维护两张表中数据之间的关系。

(1) 创建表的同时添加外键

create table emp(

    id int,

    name varchar(50),

    dept\_id int,

    foreign key(dept\_id) references dept(id)

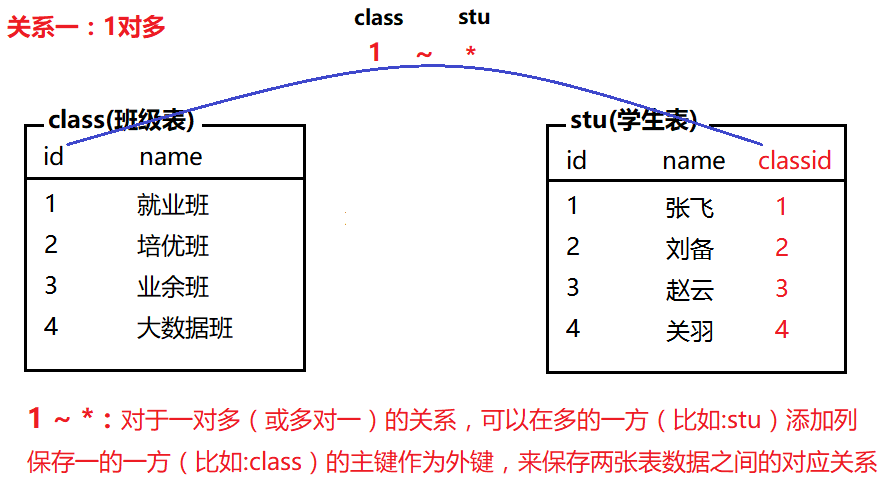
);

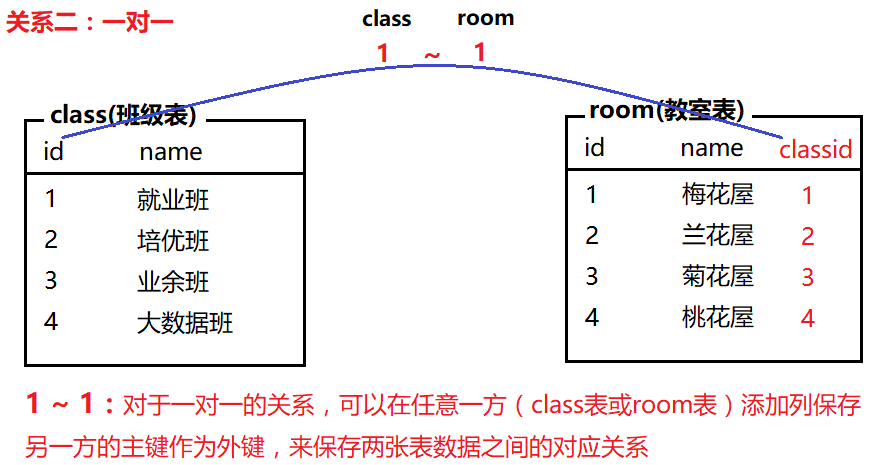
(2) 如果在创建表时没有指定外键，那么后期该如何指定外键？以及如何删除外键？

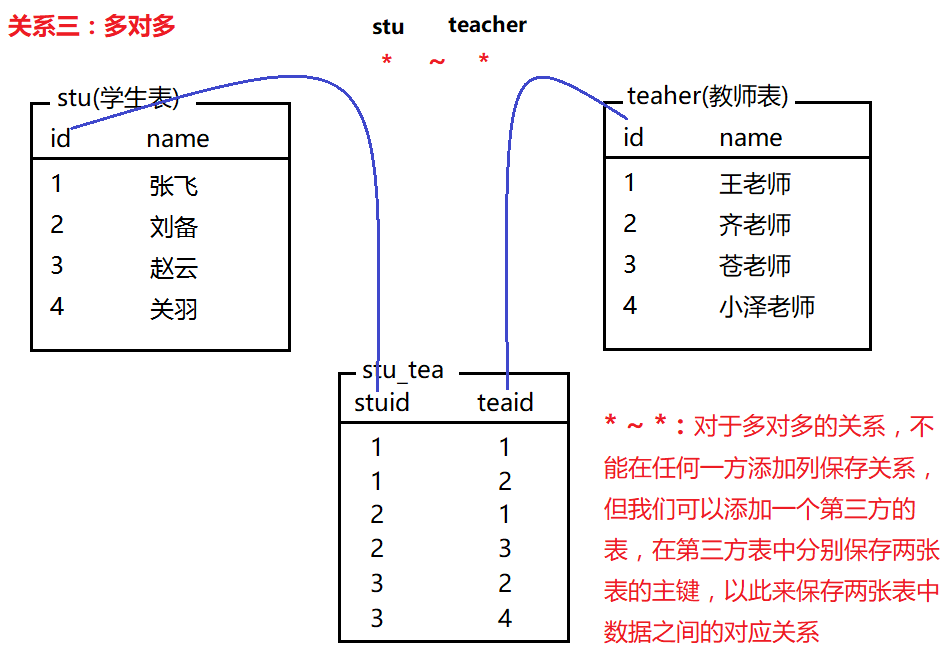
## 表关系

**常见的表关系分为以下三种：**

一对多(多对一)、一对一、多对多







## 多表查询

**-- 准备数据： 以下练习将使用db30库中的表及表记录，请先进入db30数据库!!!**

### 连接查询

-- 42.查询部门和部门对应的员工信息

### 左外连接查询

-- 43.查询所有部门和部门下的员工，如果部门下没有员工，员工显示为null

### 右外连接查询

-- 44.查询部门和所有员工，如果员工没有所属部门，部门显示为null

### 子查询

**-- 准备数据：以下练习将使用db40库中的表及表记录，请先进入db40数据库!!!**

-- 45.列出薪资比'王海涛'薪资高的所有员工，显示姓名、薪资

-- 46.列出与'刘沛霞'从事相同职位的所有员工，显示姓名、职位。

-- 47.列出薪资比'大数据部'部门(已知部门编号为30)所有员工薪资都高的员工信息，显示员工姓名、薪资和部门名称。

### 多表查询练习

-- 48.列出在'培优部'任职的员工，假定不知道'培优部'的部门编号， 显示部门名称，员工名称。

-- 关联查询两张表

-- 求出在培优部的员工

-- 49.(自查询)列出所有员工及其直接上级，显示员工姓名、上级编号，上级姓名

-- 50.列出最低薪资大于1500的各种职位，显示职位和该职位最低薪资

-- 51.列出在每个部门就职的员工数量、平均工资。显示部门编号、员工数量，平均薪资。

-- 52.查出至少有一个员工的部门，显示部门编号、部门名称、部门位置、部门人数。

-- 53.列出受雇日期早于直接上级的所有员工的编号、姓名、部门名称。

-- 54.列出每个部门薪资最高的员工信息，显示部门编号、员工姓名、薪资

## 数据库备份与恢复

### 备份数据库

在cmd窗口中(未登录的状态下)，可以通过如下命令对指定的数据库进行备份：

mysqldump -u用户名 -p 数据库的名字 > 备份文件的位置

例如: 对db40库中的数据(表，表记录)进行备份，备份到 d:/db40.sql文件中

mysqldump -uroot -p db40 > d:/db40.sql

键入密码，如果没有提示，即表示备份成功！

也可以一次性备份所有库，例如：

对mysql服务器中所有的数据库进行备份，备份到 d:/all.sql文件中

mysqldump -uroot -p --all-database > d:/all.sql

键入密码，如果没有提示，即表示备份成功！

### 恢复数据库

在cmd窗口中(未登录的状态下)，可以通过如下命令对指定的数据库进行恢复：

mysql -u用户名 -p 数据库的名字 < 备份文件的位置

例如: 将d:/db40.sql文件中的数据恢复到db60库中

-- 在cmd窗口中（已登录的状态下）， 先创建db60库，进入db60库：

create database db60 charset utf8;

-- 在cmd窗口中（未登录的状态下）

mysql -uroot -p db60 < d:/db40.sql

或 例如: 将d:/db40.sql文件中的数据恢复到db80库中

-- 在cmd窗口中（已登录的状态下），先创建db80库，进入db80库：

create database db80 charset utf8;

use db80;

-- 再通过source执行指定位置下的sql文件：

source d:/db40.sql

## 扩展内容

现创建学生表：

use test; -- 进入test库

drop table if exists stu; -- 删除学生表(如果存在)

create table stu( -- 创建学生表

id int, -- 学生id

name varchar(20), -- 学生姓名

gender char(1), -- 学生性别

birthday date -- 出生年月

);

### 修改表—新增列

语法：ALTER TABLE tabname ADD col\_name datatype [DEFAULT expr][,ADD col\_name datatype...];

1、往stu表中添加score列，double类型

alter table stu add score double;

### 修改表—修改列

语法：ALTER TABLE tabname MODIFY (col\_name datatype [DEFAULT expr][,MODIFY col\_name datatype]...);

1、修改id列，将id设置为主键

alter table stu modify id int primary key;

2、修改id列，将id主键设置为自动增长

alter table stu modify id int auto\_increment;

### 修改表—删除列

语法：ALTER TABLE tabname DROP [COLUMN] col\_name;

1、删除stu表中的score列

alter table stu drop score;

### 添加或删除主键及自增

思考：a) 在建表时，如何为id指定主键约束和自增?

b) 建好的表，如何通过修改添加主键约束和自增?

c) 如何删除表中的主键约束和自增?

1、创建stu学生表，不添加主键自增, 查看表结果

use mydb1; -- 切换到mydb1库

drop table if exists stu; -- 删除stu学生表(如果存在)

create table stu( -- 重建stu学生表，没有主键自增

id int,

name varchar(20),

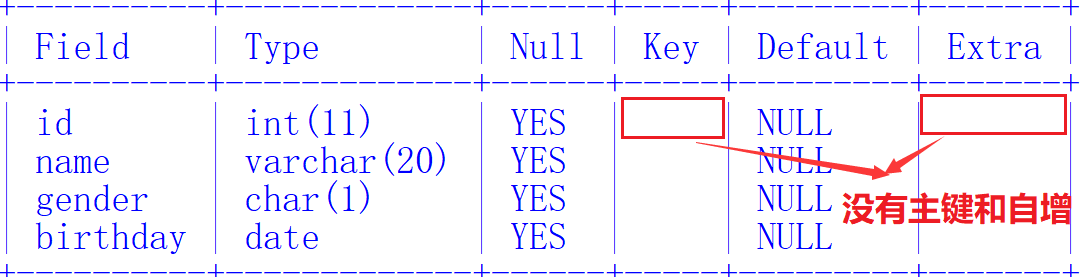
gender char(1),

birthday date

);

desc stu; -- 查看表结构

表结构如下: 没有主键约束和自增。



2、如果表没有创建，或者要删除重建，在创建时可以指定主键或主键自增

drop table if exists stu; -- 删除stu表

create table stu( -- 重新创建stu表时，指定主键自增

id int **primary key auto\_increment**,

name varchar(20),

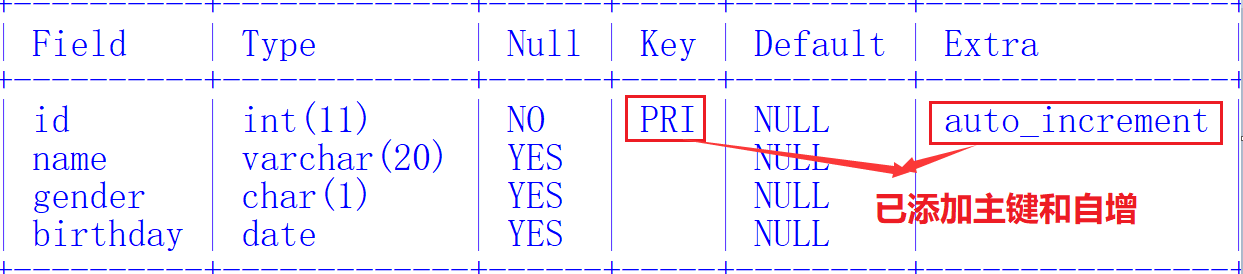
gender char(1),

birthday date

);

desc stu; -- 查看表结构

表结构如下: 已经添加了主键约束和自增。



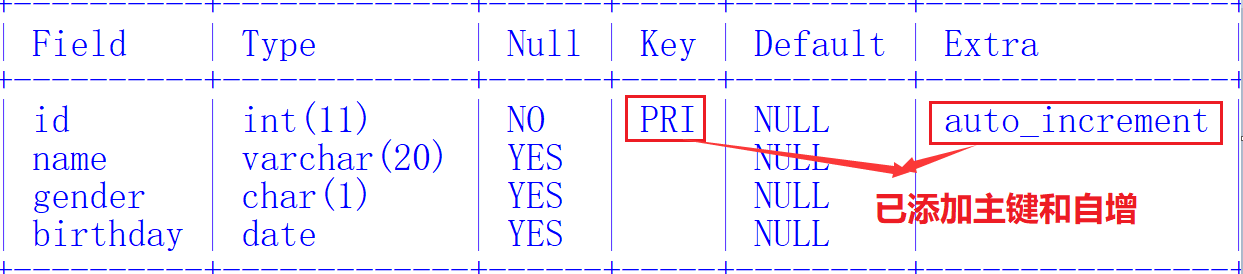
3、如果不想删除重建表，也可以通过修改表添加主键或主键自增

再次执行第1步，创建stu学生表，不添加主键自增，查看表结果

-- 例如: 将stu学生表中的id设置为主键和自动增长

alter table stu modify id int **primary key auto\_increment**;

desc stu; -- 查看表结构



如果只添加主键约束，不设置自增

alter table stu modify id int **primary key**;

如果已经添加主键约束，仅仅设置自增，但需注意:

(1)如果没有设置主键，不可添加自增

(2)只有当主键是数值时，才可以添加自增

alter table stu modify id int **auto\_increment**;

4、如果想删除主键自增

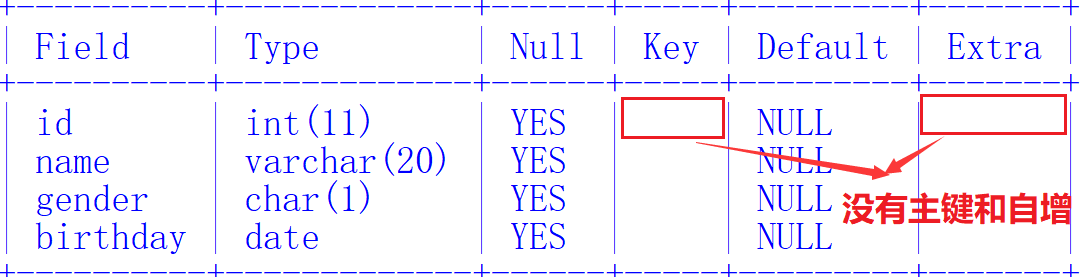
-- 删除主键自增时，要先删除自增

alter table stu modify id int;

-- 再删除主键约束

alter table stu drop primary key;

desc stu; -- 查看表结构



### 添加或删除外键约束

添加外键方式一：建表时添加外键

现有部门表如下：

-- 创建部门表

create table dept(

id int primary key auto\_increment, -- 部门编号

name varchar(20) -- 部门名称

);

要求创建员工表，并在员工表中添加外键关联部门主键

-- 创建员工表

create table emp(

id int primary key auto\_increment, -- 员工编号

name varchar(20), -- 员工姓名

dept\_id int, -- 部门编号

**foreign key(dept\_id) references dept(id)** -- 指定dept\_id为外键

);

添加外键方式二：建表后添加外键

现有部门表和员工表：

-- 创建部门表

create table dept(

id int primary key auto\_increment, -- 部门编号

name varchar(20) -- 部门名称

);

-- 创建员工表

create table emp(

id int primary key auto\_increment, -- 员工编号

name varchar(20), -- 员工姓名

dept\_id int -- 部门编号

);

如果表已存在，可以使用下面这种方式：

alter table emp add constraint fk\_dept\_id foreign key(dept\_id) references dept(id);

其中 add constraint fk\_dept\_id 表示新增列，名为**fk\_dept\_id** (名字由自己定义)，是指外键约束名称。将来可以通过该名称删除外键。

foreign key(dept\_id)中的dept\_id为外键

### 删除外键约束

1、首先通过 “show create table 表名”语法，查询含有外键表的建表语句，例如：

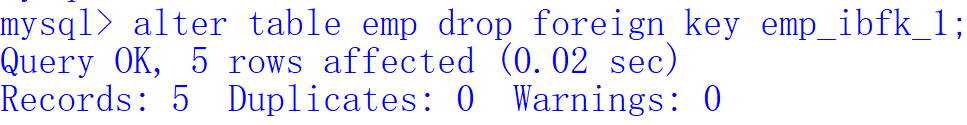
show create table emp;

显示结果如下：



其中，emp\_ibfk\_1是在创建表时，数据库为外键约束指定的一个名字，删除这个名字即可删除外键关系，例如：

alter table emp drop foreign key emp\_ibfk\_1;



外键删除成功！

### where中不能使用列别名

SQL语句的书写顺序:

select \* | 列名 -- 确定要查询的列有哪些

from 表名 -- 确定查询哪张表

where 条件 -- 通过筛选过滤，剔除不符合条件的记录

group by 分组的列 -- 指定根据哪一列进行分组

having 条件 -- 通过条件对分组后的数据进行筛选过滤

order by 排序的列 -- 指定根据哪一列进行排序

limit (countPage-1)\*rowCount, rowCount -- 指定返回第几页记录以及每页显示多少条

SQL语句的执行顺序:

from 表名 -- 确定查询哪张表

where 条件 -- 通过筛选过滤，剔除不符合条件的记录

select \* | 列名 列别名 -- 确定要查询的列有哪些，

group by 分组的列 -- 指定根据哪一列进行分组

having 条件 -- 通过条件对分组后的数据进行筛选过滤

order by 排序的列 -- 指定根据哪一列进行排序

limit (countPage-1)\*rowCount, rowCount

**\*\* 关于where中不能使用列别名但是可以使用表别名?**

是因为，表别名是声明在from中，from先于where执行，先声明再使用没有问题，但是列别名是声明在select中，where先于select执行，如果先使用列别名，再声明，这样执行会报错!!

# Day03.04 JDBC

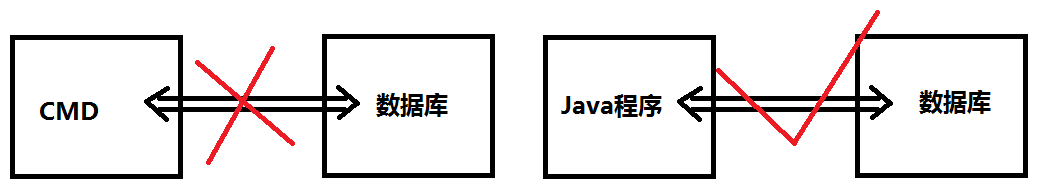
## JDBC概述

### 什么是JDBC？为什么要学习JDBC？

JDBC( Java DataBase Connectivity ) 翻译过来就是Java数据库连接，其实就是通过Java语言操作数据库的一门技术。

之前我们可以通过cmd或者navicat等工具连接并访问数据库，但是以后在企业级的开发中，我们更多的是使用JDBC（或者底层是JDBC的框架）来连接并访问数据库。

使用Java语言连接数据库，就得通过JDBC这门技术！



### 如何通过JDBC程序访问数据库?

**1.提出需求:**

创建一个 jt\_db 数据库，在库中创建一个Account表，并插入三条记录，然后利用Java程序查询出Account表中所有的记录，并打印在控制台上.

**2.开发步骤:**

(1)准备数据, 创建jt\_db库, 创建account表

drop database if exists jt\_db;

create database jt\_db charset utf8;

use jt\_db;

create table account(

id int primary key auto\_increment,

name varchar(50),

money double

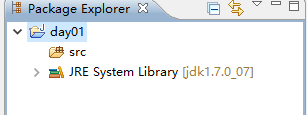
);

insert into account values(null, 'tom', 1000);

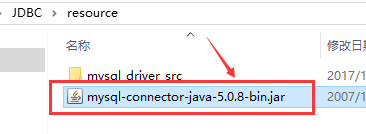
insert into account values(null, 'andy', 1000);

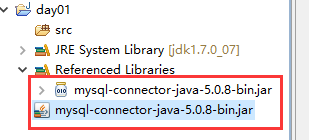
insert into account values(null, 'tony', 1000);

(2)创建JAVA工程

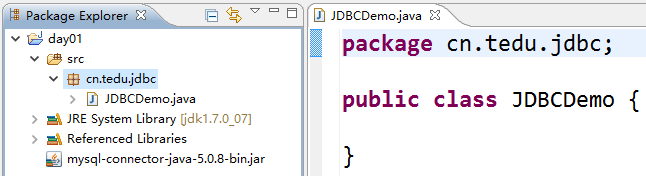


(3)导入jar包





(4)创建类并实现JDBC程序(六个步骤)



代码实现:

//1.注册数据库驱动

Class.forName("com.mysql.jdbc.Driver");

//2.获取数据库连接

Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/jt\_db?characterEncoding=utf-8", "root", "root");

//3.获取传输器

Statement stat = conn.createStatement();

//4.利用传输器发送SQL到数据库执行，并返回执行结果

ResultSet rs = stat.executeQuery("select \* from account");

//5.处理结果:将表中所有的记录输出在控制台

while (rs.next()) {

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

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

double money = rs.getDouble("money");

System.out.println(id+" : "+name+" : "+money);

}

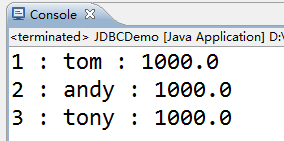
//6.释放资源

rs.close();

stat.close();

conn.close();

**3.执行结果:**



### JDBC API总结

**1、注册数据库驱动**

Class.forName("com.mysql.jdbc.Driver");

所谓的注册驱动，就是让JDBC程序加载mysql驱动程序，并管理驱动

驱动程序实现了JDBC API定义的接口以及和数据库服务器交互的功能，加载驱动是为了方便使用这些功能。

**2、获取连接之数据库URL**

Connection conn = DriverManager.getConnection(

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

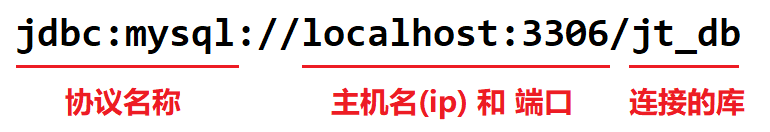
"root", "root"

);

DriverManager.getConnection() 用于获取数据连接, 返回的Connection连接对象是JDBC程序连接数据库至关重要的一个对象。

**参数2**和**参数3**分别是所连接数据库的用户名和密码。

**参数1**："jdbc:mysql://localhost:3306/jt\_db?characterEncoding=utf-8" 是连接数据库的URL，用于指定访问哪一个位置上的数据库服务器及服务器中的哪一个数据库，其写法为:



?后为拼接的参数

当连接本地数据库，并且端口为3306，可以简写为如下形式：

jdbc:mysql:///jt\_db?characterEncoding=utf-8

**3、Statement传输器对象**

Statement stat = conn.createStatement();

该方法返回用于向数据库服务器发送sql语句的Statement传输器对象，该对象上提供了发送sql的方法：

executeQuery(String sql) -- 用于向数据库发送查询类型的sql语句，返回一个ResultSet对象中

executeUpdate(String sql) -- 用于向数据库发送更新(增加、删除、修改)类型的sql语句，返回一个int值，表示影响的记录行数

4、ResultSet结果集对象

ResultSet对象用于封装sql语句查询的结果，也是一个非常重要的对象。该对象上提供了遍历数据及获取数据的方法。

(1)遍历数据行的方法

next() – 使指向数据行的索引向下移动一行

(2)获取数据的方法

getInt(int columnIndex)

getInt(String columnLable)

getString(int columnIndex)

getString(String columnLable)

getDouble(int columnIndex)

getDouble(String columnLable)

getObject(int columnIndex)

getObject(String columnLable)

...

5、释放资源

rs.close();

stat.close();

conn.close();

此处释放资源必须按照一定的顺序释放，越晚获取的越先关闭。所以先关闭 rs对象，再关闭stat对象，最后关闭conn对象。

另，为了避免上面的程序抛出异常，释放资源的代码不会执行，应该把释放资源的代码放在finally块中.

try{

...

}catch(Exception e){

...

}finally{

if (rs != null) {

try {

rs.close();

} catch (SQLException e) {

e.printStackTrace();

} finally {

rs = null;

}

}

if (stat != null) {

try {

stat.close();

} catch (SQLException e) {

e.printStackTrace();

} finally {

stat = null;

}

}

if (conn != null) {

try {

conn.close();

} catch (SQLException e) {

e.printStackTrace();

} finally {

conn = null;

}

}

}

## 增删改查

**1、新增:往account表中添加一个名称为john、money为3500的记录**

public static void add(){

Connection conn = null;

Statement stat = null;

ResultSet rs = null;

try {

//注册驱动

Class.forName("com.mysql.jdbc.Driver");

//获取连接

conn = DriverManager.getConnection(

"jdbc:mysql:///jt\_db",

"root",

"root"

);

//获取传输器

stat = conn.createStatement();

//利用传输器发送SQL到数据库执行，返回执行结果

String sql = "insert into account values "

+ "(null, 'john', 20000)";

int row = stat.executeUpdate(sql);

System.out.println("影响了"+row+"行。。。");

} catch (Exception e) {

e.printStackTrace();

}finally{

if(rs != null){

try {

rs.close();

} catch (SQLException e) {

e.printStackTrace();

}finally{

rs = null;

}

}

if(stat != null){

try {

stat.close();

} catch (SQLException e) {

e.printStackTrace();

}finally{

stat = null;

}

}

if(conn != null){

try {

conn.close();

} catch (SQLException e) {

e.printStackTrace();

}finally{

conn = null;

}

}

}

}

**2、修改:将account表中名称为john的记录，money修改为1500**

public static void update(){

Connection conn = null;

Statement stat = null;

ResultSet rs = null;

try {

//注册驱动

Class.forName("com.mysql.jdbc.Driver");

//获取连接

conn = DriverManager.getConnection(

"jdbc:mysql:///jt\_db",

"root",

"root"

);

//获取传输器

stat = conn.createStatement();

//执行SQL，返回结果

String sql = "update account set money "

+ "=2500 where name='john'";

int row = stat.executeUpdate(sql);

System.out.println("影响了"+row+"行。。。");

} catch (Exception e) {

e.printStackTrace();

} finally{

if(rs!=null){

try {

rs.close();

} catch (SQLException e) {

e.printStackTrace();

}finally{

rs = null;

}

}

if(stat!=null){

try {

stat.close();

} catch (SQLException e) {

e.printStackTrace();

}finally{

stat = null;

}

}

if(conn!=null){

try {

conn.close();

} catch (SQLException e) {

e.printStackTrace();

}finally{

conn = null;

}

}

}s

}

**3、查询:查询account表中名称为john的记录**

public static void delete(){

Connection conn = null;

Statement stat = null;

ResultSet rs = null;

try {

//通过JDBCUtils工具类获取连接

conn = JDBCUtils.getConnection();

//获取传输器

stat = conn.createStatement();

//执行SQL，返回结果

String sql = "delete from account "

+ "where name='john'";

int row = stat.executeUpdate(sql);

System.out.println("影响了"+row+"行。。。");

} catch (Exception e) {

e.printStackTrace();

} finally{

//通过调用JDBCUtils类中的close方法将资源关闭

JDBCUtils.close(conn, stat, rs);

}

}

**4、删除:删除account表中名称为john的记录**

public static void query(){

Connection conn = null;

Statement stat = null;

ResultSet rs = null;

try {

conn = JDBCUtils.getConnection();

stat = conn.createStatement();

String sql = "select \* from "

+ "account where id=1";

rs = stat.executeQuery(sql);

if(rs.next()){

System.out.println(

rs.getInt("id")+" : " +

rs.getString("name")+" : "+

rs.getDouble("money")

);

}

} catch (Exception e) {

e.printStackTrace();

} finally{

JDBCUtils.close(conn, stat, rs);

}

}

## PreparedStatement

在上面的增删改查的操作中，使用的是Statement传输器对象，而在开发中我们用的更多的传输器对象是PreparedStatement对，PreparedStatement是Statement的子接口，比Statement更加安全，并且能够提高程序执行的效率。

### 模拟用户登录案例

(1)准备数据

use jt\_db;

create table user(

id int primary key auto\_increment,

username varchar(50),

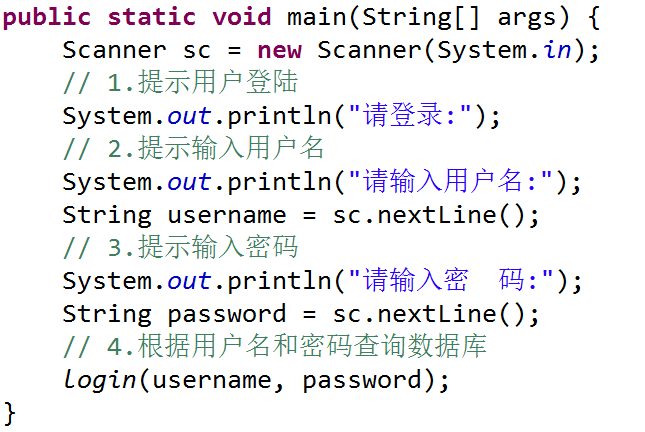
password varchar(50)

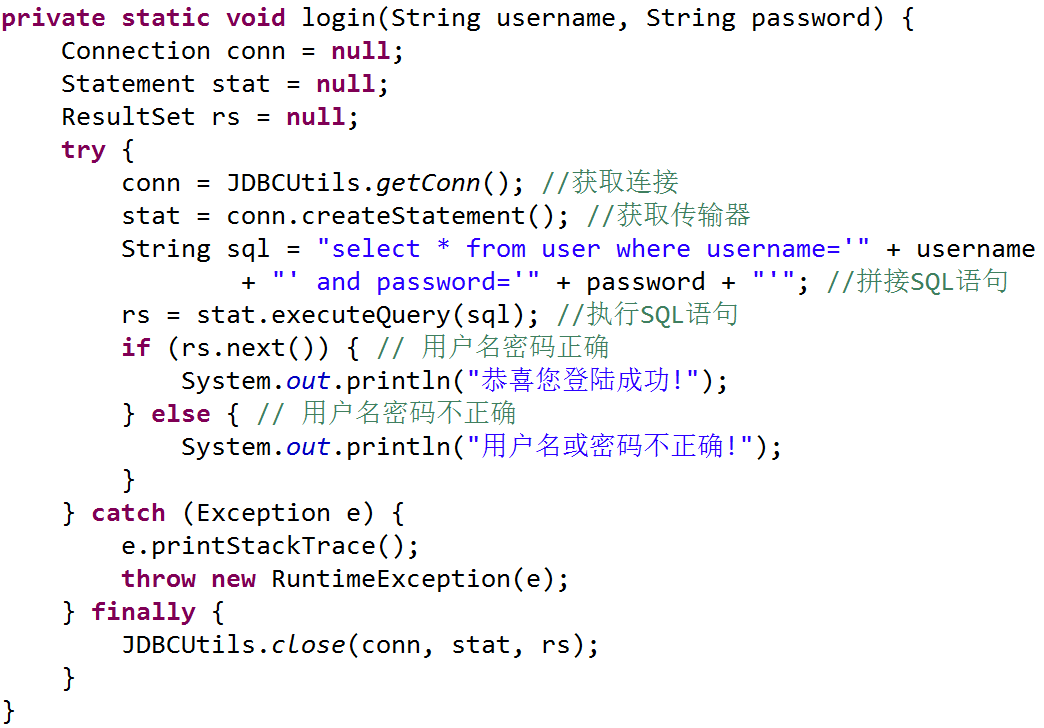
);

insert into user values(null,'张三','123');

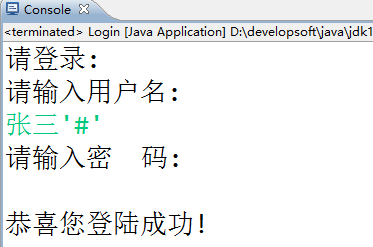
insert into user values(null,'李四','234');

(2)创建LoginUser 类，提供 main 方法 和 login 方法。

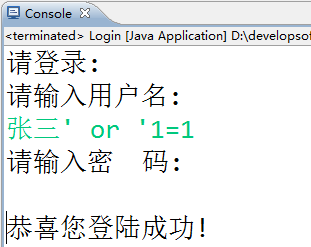




**执行时，输入：**



**或输入**



### SQL注入攻击

通过上面的案例，我们发现在执行时，不输入密码只输入用户名也可以登陆成功。这就是SQL注入攻击。

SQL注入攻击：由于后台的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，就可能会导致SQL语句的语意发生变化。从而执行一些意外的操作。

### 防止SQL注入攻击

如何防止SQL注入攻击？

(1)使用正则表达式对用户提交的参数进行校验。

(2)使用PreparedStatement对象来替代Statement对象。

下面通过第二种方式解决SQL注入攻击：添加loginByPreparedSatement方法，在方法中，使用PreparedStatement来代替Statement作为传输器对象使用！



再次执行程序，按照上面的操作登录。此时，已经成功的防止了SQL注入攻击问题了。

使用PreparedStatement对象可以防止SQL注入攻击

而且通过方法设置参数更加的方便且不易出错!

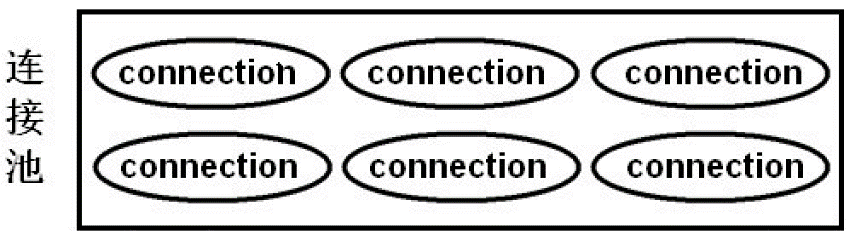
还可以从某些方面提高程序执行的效率!

## 数据库连接池

### 什么是连接池

在开发中，所谓的池就是一个容器，来存储程序的中的数据.

而数据库连接池就是用来存储数据库连接的池子，用于在整个程序中共享连接，减少连接开关的次数，实现连接的复用，从而提高程序执行的效率.



提示：连接池还有另外一个名字，即DataSource，翻译为数据源，因此连接池也叫做数据源。

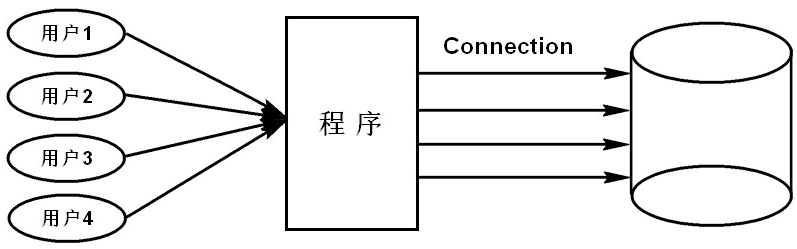
### 为什么要使用数据库连接池

对于数据库来说，频繁的开关连接会非常的耗费资源，也会导致程序执行效率的低下。

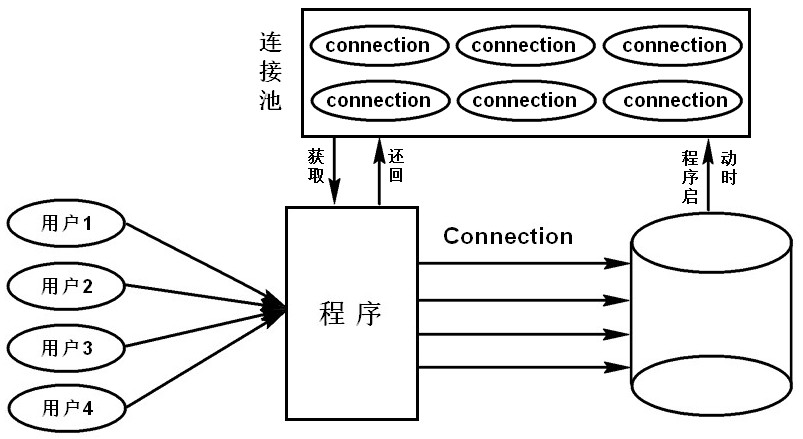
我们可以在程序中创建一个池子，在程序启动时就初始化一批连接放在连接池中，当用户需要连接时，就直接从池子中拿一个连接使用，当用完连接后，也不要将连接关闭，而是将连接还回池中，下一个用户需要连接时也是如此。

这样可以减少链接开关的次数，从而提供程序执行的效率.

1、传统方式操作数据库



2、使用连接池操作数据库



### 如何使用C3P0连接池

使用C3P0连接池开发步骤：

1、导入开发包



2、创建数据库连接池

ComboPooledDataSource cpds = new ComboPooledDataSource();

3、设置数据库连接的基本信息

(1)方式一：(不推荐)

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

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

cpds.setUser("root");

cpds.setPassword("root");

(2)方式二：(推荐)

在类目录下(开发时可以放在src或者类似的源码目录下), 添加一个c3p0-config.xml文件, 配置内容如下：

<?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?characterEncoding=utf-8

</property>

<property name="user">

root

</property>

<property name="password">

root

</property>

</default-config>

</c3p0-config>

(3)方式三：(推荐)

在类目录下(开发时可以放在src或者类似的源码目录下(config目录)), 添加一个c3p0.properties文件, 配置内容如下：

c3p0.driverClass=com.mysql.jdbc.Driver

c3p0.jdbcUrl=jdbc:mysql:///jt\_db?characterEncoding=utf-8

c3p0.user=root

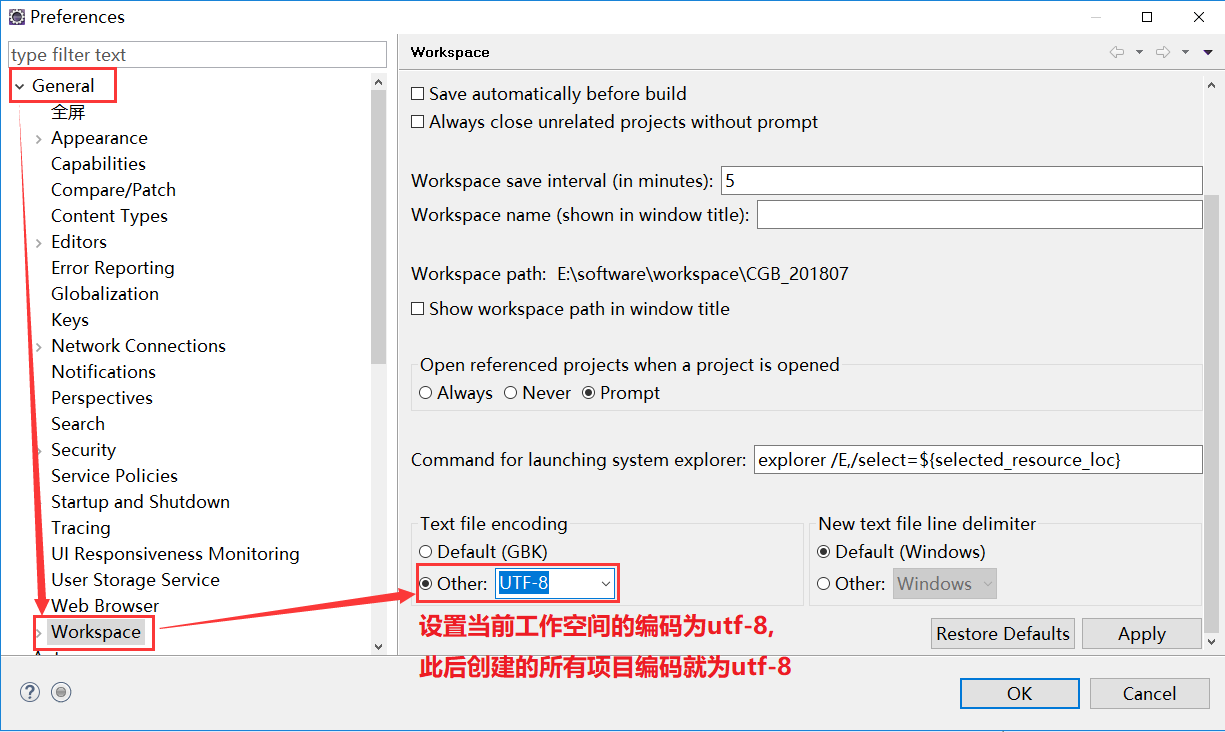
c3p0.password=root

## 扩展：切换工作空间，修改默认编码

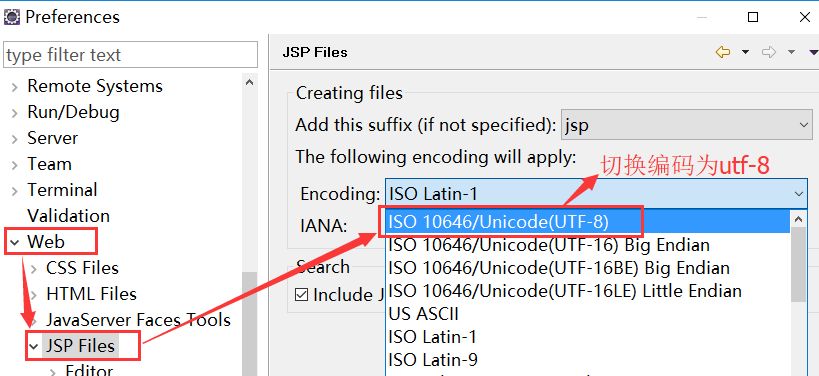
### 切换新的工作空间，设置工作空间编码为utf-8

(1)切换到新的工作空间目的: 若旧的工作空间内容过多，可能会导致eclipse不编译，甚至进入休眠状态。

(2)设置工作空间编码为utf-8，此后在当前工作空间下创建的项目编码也默认是utf-8。



### 设置JSP文件的编码为utf-8



# Day05 HTML、CSS

## HTML概述

### HTML是什么

HTML: 超文本标记语言，是开发网页的最基础的语言

(1)由W3C组织提供

(2)使用html开发的网页文件，以".html"或".htm "（.shtml）为后缀

(3)使用html开发的网页文件，由浏览器负责解析并执行（即显示在浏览器中）

关于HTML:

(1) HTML是一门标记(也叫做标签或元素)语言，不是编程语言

(2) HTML是一门标记语言，是利用标记来组织网页结构的

(3) HTML是文档的一种，例如：word pdf txt…

### HTML的结构

**1.案例：编写我的第一个HTML网页, 并用浏览器打开**

新建一个txt文档，将后缀名改为.html，代码实现如下：

<!DOCTYPE html>

<html>

<head>

<title>网页的标题</title>

</head>

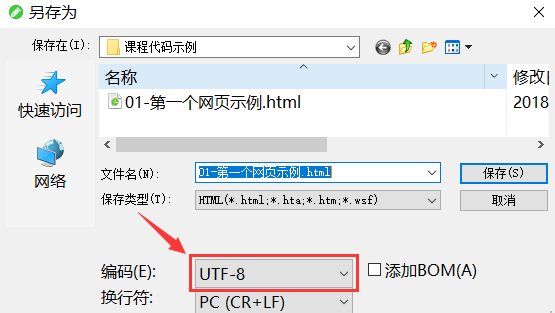
<body>

Hello CGB1808...

</body>

</html>

例如：另存html文档时保存的编码为utf-8:



只要在html文档中添加一个meta标签，同时也指定保存的编码为utf-8即可解决乱码问题!!

<!DOCTYPE html>

<html>

<head>

<title>网页的标题</title>

<meta charset="utf-8" />

</head>

<body>

Hello CGB1808...

</body>

</html>

**2.HTML结构详解**

(1)**<!DOCTYPE HTML>** 文档声明, 用来声明HTML文档所遵循的HTML规范和版本

上面是html5.0的声明, 也是目前最常用的版本

(2)**<head></head>** 头部分, 用来存放HTML文档的基本属性信息, 比如网页的标题, 文档使用的编码等, 这部分信息会被浏览器优先加载.

(3)**<body></body>** 体部分, 用来存放网页可视化数据. 即真正的网页数据

(4)**<title></title>** 声明网页的标题

(5)**<meta charset="utf-8"/>** 用来通知浏览器使用哪一个编码来打开HTML文档, 这个编码一定要和文件保存时的编码保持一致, 才不会出现中文乱码问题.

### HTML语法(了解)

**1、html标签**

HTML是一门标记语言，标记也叫做元素/标签，标签分为开始标签和结束标签。例如:

<body></body>

<table></table>

<form></form>

...

如果标签内没有内容要修饰，可以合并成一个自闭标签。例如:

<meta/> <br/> <hr/> <input/> <img/>等等

**2、html属性**

标签都可以具有属性，属性可以有多个，多个属性之间用空格隔开。例如：

<font size="7" color="red" face="华文琥珀">文本内容…</font>

提示：属性的值用单引号或双引号引起来，或者不用引号。通常使用双引号引起来。

3、html注释

注释格式：<!-- 注释内容 -->，注意html注释不能交叉嵌套，例如：

<!-- 下面声明了一个标题 -->

<h1>一级标题</h1>

4、html空格和换行

如何在网页中做一个空格或者做一个换行？

由于在网页中多个连续的空白字符会被当成一个空格来显示，所以

如果要做一个**空格**，可以用转义字符 **&nbsp;**来代替;

如果要做一个**换行**，可以用**<br/>**标签来代替;

提示：中文状态下的空格(全角空格)请使用:&emsp;

## HTML标签

### 图像标签

img标签用于在网页中插入一幅图片

标签介绍：

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

属性介绍：

src属性：用来指定图片的url地址（即图片的所在路径）

width属性：指定图片的宽度，单位可以为px（像素）或者%（百分比）

height属性：指定图片的高度，单位可以为px（像素）或者%（百分比）

自己动手试一试~！

### 超链接

在HTML网页文档中，通过a标签可以创建一个超链接标签

1、用于创建指向另外一个文档的超链接（点击后可以跳转到另外一个文档），例如：

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

属性介绍：

href属性：用来指定点击超链接后将要跳转到的url地址

target属性：用来指定以何种方式打开超链，其常用取值为:

\_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

**1.案例：在网页中插入一个表格**

html代码示例:

<table>

<tr>

<td>11</td>

<td>12</td>

<td>13</td>

</tr>

<tr>

<td>21</td>

<td>22</td>

<td>23</td>

</tr>

<tr>

<td>31</td>

<td>32</td>

<td>33</td>

</tr>

</table>

在浏览器中显示效果为：



在head标签内添加如下内容:

<style>

table,td{

border:1px solid #000;/\* 设置边框 \*/

border-collapse:collapse;/\* 设置边框合并 \*/

}

table{

width:70%; /\* 设置表格宽度 \*/

margin:0px auto;/\* 设置表格居中 \*/

background:pink;/\* 设置表格背景颜色 \*/

}

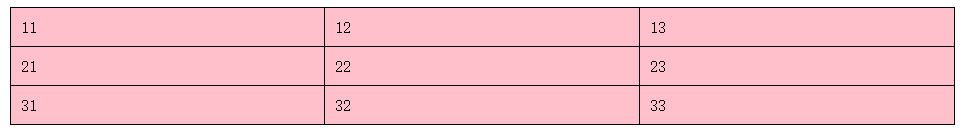
td{

padding:10px;/\* 设置单元格边框和内容之间的距离 \*/

}

</style>

再次刷新浏览器显示效果为：



1、表格标签介绍

table – 用来定义一个表格

tr – 用来定义表格中的行

td – 用来定义表格中的单元格

th – 用来定义表头

**2、练习：使用表格标签在网页中生成一个表格，如下图：**

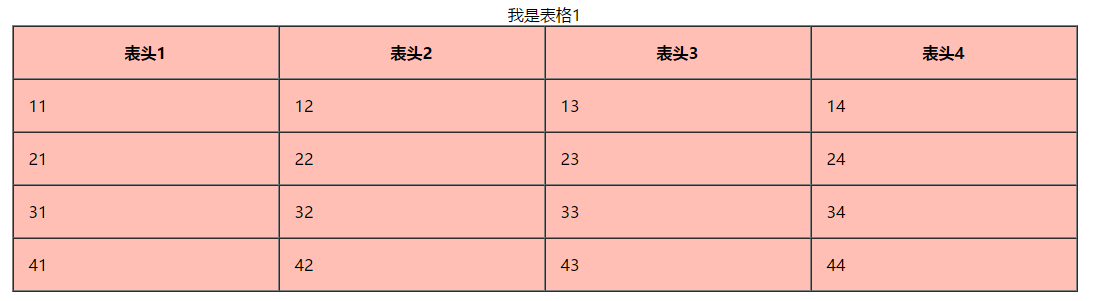
要求如下:

(1) 表格内容如下图, 并设置表格边框

(2) 设置单元格之间没有缝隙, 并设置单元格边框和内容之间的距离为5px

(3) 设置表格的背景颜色为pink, 并设置表格的宽度为70%

(4) 设置表格在网页中居中显示, 并为表格添加表头以及标题



### 表单标签

**表单的作用：**是向服务器发送/提交数据。

1、浏览器向服务器发送数据的两种方式

思考：如果浏览器在和服务器进行通信的过程中，需要向服务器提交一些数据，比如在登陆时，需要向服务器提交用户名和密码，或者在百度搜索时，需要提交搜索的关键词汇，这些数据浏览器是如何发送给服务器的？

**1、通过超链接向服务器发送数据**

其实就是在超链接后面通过？拼接参数将数据带给服务器

参数和参数值之间用等号分隔，参数可以有多个，多个参数之间用&分隔，并且参数的名字可以重复。



**2、通过表单向服务器发送数据**

其实里通过表单及表单项标签，用户可以通过表单项输入数据，通过提交表单向服务器发送数据。例如下图：



标签介绍：

<form action="http://www.baidu.com" method="GET"></form>

属性介绍：

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

表单中可以有多个输入项，输入项必须有name属性才可以被提交，如果输入项没有name属性，则表单在提交时会忽略它

1、input元素

**(1) 普通的文本输入框**

用户名: <input **type="text"** name="username" value="设置默认值"/>

例如：



**(2) 密码输入框**

密码: <input **type="password"** name="password" value="设置默认值"/>

例如：



**(3) 单选按钮(单选框)**

性别: <input **type="radio"** name="gender" value="man"/>男

<input **type="radio"** name="gender" value="woman" />女

属性介绍：

其中checked属性可以设置选项默认被选中

value用来指定选项被提交时的值, 如果不设置value, 选项被提交时, 值为on.

例如：



**(4) 复选框**

爱好: <input **type="checkbox"** name="like" value="lanqiu"/>篮球

<input **type="checkbox"** name="like" value="zuqiu"/>足球

属性介绍：

其中checked属性可以设置选项默认被选中

value属性用来指定选项被提交时的值, 如果不设置value, 选项被提交时, 值为on.

例如：



**(5) 提交按钮**

<input **type="submit"** value="按钮上显示的文本"> 用来提交表单用的

**(6) 普通的按钮**

<input **type="button"** value="按钮上显示的文本">

普通按钮本身没有功能, 但是可以通过js来为按钮添加功能和行为.

2、textarea多行文本输入区域

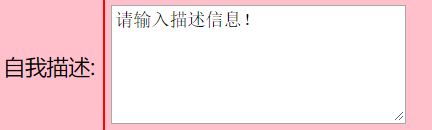
个人描述: <textarea name="description" cols="宽度" rows="高度">请输入个人描述..</textarea>

属性介绍：

cols属性：设置输入框宽度

rows属性：设置输入框高度

示例：



3、下拉选框：

城市:

<select name="city">

<option value="bj">北京</option>

<option value="sh">上海</option>

<option value="gz" selected="selected">广州</option>

</select>

**属性介绍：**

其中selected属性用来设置该选项默认被选中

value属性用来设置选项被提交时的值

例如：



### 表单细节问题

1、提交表单时，表单中的数据为什么没有被提交？

在表单中，每一个要提交数据的表单项标签都要具有name属性，例如：

<!-- 用户名 -->

<input type="text" name="username"/>

<!-- 密码 -->

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

如果表单项标签没有name属性，在表单提交时，该项默认会被忽略。

2、如何让多个单选框只能有一个被选中？

设置这多个单选框的name属性值相同，如果name相同，这多个单选框就默认是一个组，一个组就只能选中一个选项。

<!-- 性别 -->

<input type="radio" name="gender"/>男

<input type="radio" name="gender"/>女

3、为什么单选框、复选框选择后提交时的值都是on?

如果单选框或复选框没有设置value属性，选择任何一个选项最终提交的值都是on，这样服务器无法区分用户到底选择是什么选项。

因此我们需要通过value属性为单选框和复选框设置提交时的值，例如：

<!-- 性别 -->

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

<input type="radio" name="gender" value="female"/>女

<!-- 爱好 -->

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

<input type="checkbox" name="like" value="football"/>足球

<input type="checkbox" name="like" value="velloyball"/>排球

4、如何设置单选框或复选框默认选中某一项?

只需要在单选框或复选框上面添加一个属性: checked="checked"，例如：

单选框: <input type="radio" name="gender" value="female"

checked="checked"/>女

复选框: <input type="checkbox" name="like" value="football"

checked="checked"/>足球

5、如何设置下拉选框默认选中某一项?

如果不设置，下拉选框默认选中的第一个选项，可以通过 selected="selected"属性设置某一项默认被选中，例如：

<select name="city"><!-- 下拉选框 -->

<option>北京</option><!-- 选项 -->

<option>上海</option>

<option selected="selected">广州</option>

<option>深圳</option>

</select>

6、下拉选框中option选项上的value属性的作用是什么？

选中某一个option，在表单提交时，如果option上没有value属性，就提交option标签的内容；如果option上有value属性，就提交value属性的值。

<select name="city"><!-- 下拉选框 -->

<option value="bj">北京</option><!-- 选项 -->

<option value="sh">上海</option>

<option>广州</option>

<option>深圳</option>

</select>

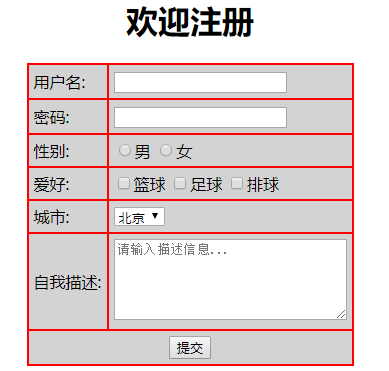
## 注册表单案例

使用表格标签、表单及表单项标签、图像标签等实现网站注册表单。

部分要求如下:

(1)设置表格居中、边框颜色为2px red，背景颜色为lightgrey；

(2)设置单元格边框和内容之间的距离为5px；

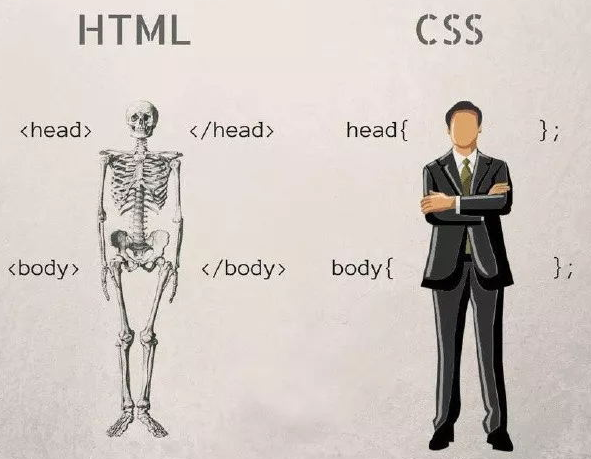


## CSS概述

### 什么是CSS？（了解）

CSS：(Cascading Style Sheets)层叠样式表，用来修饰、美化HTML网页的一门语言。

使用CSS来设置样式，可以将设置样式的CSS代码和展示数据的HTML代码进行分离。并且还可以实现代码的复用，增强网页的展示能力。



**html的作用:** 通过html标签组织网页的结构

**css的作用:** 渲染网页、美化网页

### 在HTML中引入CSS

方式1：通过style属性引入css

大多数html标签都可以具有style属性，可以通过标签上的style属性为当前标签设置样式，例如：

<!--

1.通过标签上的style属性给div设置样式

边框:2px solid red

字体大小:26px

背景颜色为:pink

-->

<div style="border:*2px solid red*;font-size:*26px*;background:*pink*;">

这是一个div...

</div>

这种方式设置样式不推荐大量使用，会造成页面结构的混乱，不利于后期的扩展和维护。

方式2：通过style标签引入css

在head标签内部提供一个style标签，在style标签内部可以选中元素进行修饰，例如：

<!-- 2.通过style标签给span设置样式如下：

边框: 2px solid green

字体大小: 30px

字体加粗

-->

<style type=*"text/css"*>

/\* \*\*\*\*\*\* CSS样式 \*\*\*\*\*\* \*/

**span**{

border:*2px solid green*;

font-size:*30px*;

font-weight:*bolder*;

}

</style>

这种方式是将所有的css代码集中在一个style标签内部统一管理。初步实现了html代码和css代码的分离。实现了代码的复用（文件内部复用）。

推荐使用这种方式。

方式3：通过link链接引入外部的css文件

在head标签下添加一个link标签，link标签可以引入外部的css文件

<!-- 引入demo.css文件 -->

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

demo.css文件内容

@charset *"UTF-8"*; /\* 指定当前css文件的编码为utf-8 \*/

**p**{

border: *2px* *solid* *blue*;

color: *red*;

font-family: *"华文琥珀"*;

text-indent: *50px*;

}

## CSS选择器

所谓的选择器就是能够在html中帮助我们选中元素进行修饰的一门技术。

### 标签名选择器

根据标签(元素)的名称来选择指定名称的元素进行样式的修饰.

格式：**标签名{ css属性… }**

示例：

/\* ----- 1.标签名选择器 -----

将所有span标签的背景颜色设置为#DDA0DD, 设置字体大小为24px，

字体加粗；\*/

**span**{

background:*#DDA0DD*;

font-size:*24px*;

font-weight:*bolder*;

}

### class选择器

通过标签上通用的属性class，可以为标签指定所属的类(组)，所有class值相同的元素则为一组(类)，可以通过class属性的值选择这一组的标签，为这一组的标签统一设置样式。

格式：**.类名{ css属性… }**

示例：

/\* ----- 2.class类选择器 -----

(1)将所有的span（但是不包括div和p标签下的span）的背景颜色

设置为#b1eb0a, 字体颜色为#0b190b；

(2)将div下的span和内容为"span111"的span，背景颜色设置为

#f3db09、字体颜色设置blue； \*/

*.c1*{ /\* 表示选中所有class值为c1的元素 \*/

background: *#b1eb0a*;

color: *#0b190b*;

}

*.c2*{ /\* 表示选中所有class值为c2的元素 \*/

background:*#f3db09*;

color: *blue*;

}

另外，一个元素可以设置多个class值，表示当前元素同时属于多个分组。例如：

<span class="c1 c2">

span111

</span>

多class的样式会同时作用在当前这个元素上。

如果多个样式之间出现冲突（c1和c2同时给当前元素设置背景），后设置的样式会覆盖前面设置的样式。如上，由于c2在后面所以c2会覆盖c1。

### id选择器

通过标签上通用的属性id，可以为标签设置唯一的标识(ID的值在整个HTML中应该是独一无二的)，可以通过id值选中指定的元素。

格式：**#id值{ css属性… }**

/\* ----- 3.id选择器 -----

用id选择器将第一个p标签设置字体大小为28px，字体颜色

为red, 首行文本缩进20px。 \*/

*#p1*{ /\* 选中id为p1的元素 \*/

font-size:*28px*;

color: *red*;

text-indent: *20px*;

}

### 后代选择器

选中父元素内部的指定后代元素

格式： 父选择器 后代元素{ css属性… }

示例：

/\* ----- 4.后代选择器 -----

为p元素内部的所有span元素，设置字体大小为20px,字体颜色

为绿色,背景颜色为cyan; \*/

**p** **span**{/\* 选中所有p元素内部的所有span元素 \*/

font-size: *20px*;

color:*green*;

background: *cyan*;

}

### 属性选择器

根据属性条件选中元素进行修饰

格式: 选择器[属性条件…] { css属性... }

示例:

/\* ----- 5.属性选择器 -----

为所有的文本输入框，设置背景颜色为pink、字体大小

30px，首行缩进30px； \*/

**input**[type='text'][name='user']{

/\* 匹配元素名为input的并且type的值为text的元素 \*/

background:*pink*;

font-size:*30px*;

text-indent:*30px*;

}

其中属性条件可以有多个, 多个属性条件放在多个方括号内部即可，例如，匹配所有的用户名输入框：

input[type='text'][name='username']{...}

## 常用属性总结

### 文本属性

**1.text-align** 设置元素中文本水平对齐方式，其常用取值为:

left: 默认值。左对齐

right: 右对齐

center: 居中对齐

justify: 两端对齐

**2.text-decoration**：设置文本下划线，其常用取值为:

underline: 有下划线

none: 没有下划线

**3.letter-spacing**: 设置字符间隔/间距，其常用取值为:

normal

像素值

**4.text-shadow**: 设置字体阴影，其取值为:

像素值 像素值 像素值 颜色值

第一个值为阴影水平位移, 第二个值为阴影垂直位移, 第三个值为阴影扩散值, 第四个值为阴影颜色

### 字体属性

font-size：设置字体大小

font-weight：设置字体粗细 bold、bolder、normal

font-family：设置字体，比如微软雅黑、黑体、楷体等

color：设置字体颜色

### 背景属性

background-color：设置背景颜色

background-image：设置背景图片，url('图片的路径');

background-repeat：设置或检索对象的背景图像是否及如何铺排

no-repeat(不重复)

repeat-x(横向重复)

repeat-y(纵向重复)

repeat(横向、纵向都重复)

repeat-y：纵向重复

background-position：设置或检索对象的背景图像位置

值1(背景图片距离元素左边框的位移)

值2(背景图片距离元素上边框的位移)

### 边框（border）

### 其他属性

width：设置元素的宽度

height：设置元素的高度

border: 宽度 样式 颜色;

border用于设置元素的边框，可以同时设置边框的宽度、样式、颜色等

例如，设置div元素的边框为2像素、实线、红色：border:2px solid red;

## 扩展内容

* + 1. **display属性**

display用来设置元素的类型，常用取值：

block：块级元素的默认值

默认情况下独占一行

可以设置宽高

inline：行内元素的默认值

默认情况下多个行内元素可以处在同一行

一般不能设置宽高

inline-block：行内块元素

多个元素既可以显示在同一行, 也可以设置宽和高

none：表示隐藏元素

* + 1. **元素类型(了解)**

(1)块级元素（block）

默认情况下，块级元素独占一行

可以设置宽和高，就是设置宽和高

如果不设置宽和高，其中宽是默认填满父元素，而高是由内容决定(由内容支撑)

(2)行内元素（inline）

默认情况下，多个行内元素处在同一行

不能设置宽和高

(3)行内块元素（inline-block）

既具备行内元素的特征(可以同行显示)，还具备块级元素的特征(可以设置宽和高)

# Day06 JavaScript

## JavaScript简介

### 什么是JavaScript(了解)

全称JavaScript，简称JS，由Netscape公司提供的。

是嵌入在html网页中的一门脚本语言，专门用于实现网页的交互。

JS嵌入在网页中，由浏览器负责解析并执行。为网页添加各式各样的动态效果或为表单添加校验，为用户提供更好的浏览体验。

JS语言和其他语言一样，有自身的基本数据类型、表达式、运算符、语句等程序的基本组成部分。

JS目前已经被广泛的用于web应用开发。

### JS特点和优势(了解)

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）

### 在HTML书写JS的方式(掌握)

1、在script标签内部可以书写JS代码：

**示例：**

<head>

...

<!-- 第一种引入JS的方式 -->

<script>

alter("引入JS的第一种方式...");

</script>

</head>

script标签可以放在head或body或页面的其他位置中，通常会放在head或body中

2、通过script标签引入外部的JS文件

将所有JS代码放在一个单独的JS文件中（后缀名为.js），再通过script标签引入JS文件即可，例如：

<head>

...

<!-- 第二种引入JS的方式 -->

<script src=*"demo.js"*></script>

</head>

注意事项: 如果script标签是用于引入JS文件的，就不要在该标签内部再书写JS代码，否则写了也不会执行！！例如：

<script src="demo.js">

    alert(333);//此处的代码执行不到

</script>

## JavaScript语法

### 注释格式

JS的注释符合和Java的注释符合相同，如下：

**//** 单行注释内容

**/\*** 多行注释内容 **\*/**

### 数据类型

1、基本数据类型

JS的基本数据类型一共有五种，分别为数值类型(number)、字符串类型(string)、布尔类型(boolean)、undefined、null。

**(1)数值类型**

JS中的数值底层**都是浮点型**，但在需要时会自动的和整型之间进行转换，例如：

2.4+3.6 结果是6，而不是6.0

NaN 非数值 Infinity正无穷 -Infinity负无穷

(2)字符串类型

JS中的字符串是**基本数据类型**，但也有对应的包装对象，和Java中不同的是，JS中的字符串可以用单引号或双引号引起来。例如：

var str1 = "Hello JS"; //sting类型

var str2 = 'Hello JS';

Var str3 = new String(“a”) //Objecet对象

//都可以当成对象使用(JS会自动进行数据类型转换)

(3)布尔类型

布尔类型常用条件测试中，值为true和false。

(4)undefined和null（了解）

undefined类型的值只有一个, 就是undefined。当声明了变量没有为变量赋值时，变量的值则为undefined。

var x;

alert(x);// x的值为undefined

null类型的值也只有一个, 就是null。表示空值。可以作为函数的返回值，表示函数返回的是一个空的对象

undefined和null不能调用属性或方法的，否则就会抛异常

2、复杂数据类型

JS的复杂数据类型主要指对象

需要注意的是，在JS中，数组、函数也属于对象

### 变量声明

1、在JS中通过var声明变量，例如：

var a = 10; //声明变量，赋值为数值10

var b = true; //声明变量，赋值为布尔值true

var c = "Hello JS"; //声明变量，赋值为字符串"Hello JS"

2、在JS中声明的变量不区分类型，可以指向任意的数据类型，例如：

var x = 10; //声明变量，赋值为数值10

x = true; //将x赋值为布尔值true

x = "Hello JS"; 将x赋值为字符串"Hello JS"

JS严格区分大小写

如果一行代码后面没有分号结束，也可以正常解析

### JS运算符

JS和Java中的运算符大致相同，例如：

算术运算符: +，-，\*，/，%，++，--

赋值运算符: =，+=，-=，\*=，/=，%=

比较运算符: ==，!=，>，>=，<，<=

位运算符: & ， |

逻辑运算符: && ，||

前置逻辑运算符: ! (not)

三元运算符: 表达式 ? 表达式 : 表达式

。。。

### JS语句

JS中的语句和Java中的大致相同

1、if分支结构

if分支结构用于基于不同的条件来执行不同的动作。语法结构如下：

if (条件 1){

当条件 1 为 true 时执行的代码

}else if (条件 2){

当条件 2 为 true 时执行的代码

}else{

当条件 1 和 条件 2 都不为 true 时执行的代码

}

2、switch语句

使用 switch 语句来选择要执行的多个代码块之一。语法结构如下：

switch(n){

case 1:

执行代码块 1

break;

case 2:

执行代码块 2

break;

default:

与 case 1 和 case 2 不同时执行的代码

}

执行原理：首先设置表达式 n（通常是一个变量）。随后表达式的值会与结构中的每个 case 的值做比较。如果存在匹配，则与该 case 关联的代码块会被执行。请使用 break 来阻止代码自动地向下一个 case 运行。

3、for循环语句 -- 循环代码块一定的次数

for 循环的语法结构如下：

for (语句 1; 语句 2; 语句 3){

被执行的代码块

}

4、while循环 -- 在指定条件为真时循环执行代码块

JS中while循环也分为while和do/while循环，下面为while循环语法结构:

while (条件){

需要执行的代码

}

while 循环会在指定条件为真时循环执行代码块。

案例1：if分支案例

例子: 接收用户输入的成绩，判断成绩所属的等级

80~100(包括80，也包括100) 优秀

60~80(包括60，但不包括80) 中等

0~60(包括0，但不包括60) 不及格

其他值 输入有误

代码实现如下：

**var** score = prompt("请输入您的成绩: ");

score = parseFloat( score ); //将传入的内容转成浮点型的数值

**if**( score >= 80 && score <= 100 ){

alert("优秀!");

}**else** **if**( score >= 60 && score < 80 ){

alert("良好!");

}**else** **if**( score >= 0 && score < 60 ){

alert("不及格!");

}**else**{

alert("您输入的成绩不合法!");

}

案例2：switch语句案例

实现一个简易的计算器：可以接收用户输入的两个数值和一个操作符(+、-、\*、/)，根据操作符号的不同，对两个数值执行不同的运算。

代码实现如下：

**var** n1 = prompt("请输入数值1：") - 0 ;

**var** opt = prompt("请输入运算符："); // + - \* /

**var** n2 = prompt("请输入数值2：") - 0 ;

**switch**( opt ){

**case** "+":

alert("两个数的和为: "+(n1+n2) ); **break**;

**case** "-":

alert("两个数的差为: "+(n1-n2) ); **break**;

**case** "\*":

alert("两个数的乘积为: "+(n1\*n2) ); **break**;

**case** "/":

alert("两个数的商为: "+(n1/n2) ); **break**;

**default**:

alert("您输入的运算符不合法! ");

}

案例3：while循环语句案例

遍历1~100之间的所有整数，求1~100之间所有整数的和，并输出到控制台;

代码实现如下：

**var** sum = 0;

**var** i = 1;

**while**( i > 0 && i <= 100 ){

sum += i;

i++;

}

案例4：for循环语句案例

遍历下面数组中的元素，将元素输出到控制台。

**var** arr = [123, "abc", **false**, **new** Object() ];

代码实现如下：

**var** arr = [123, "abc", **false**, **new** Object() ];

**for**( **var** i=0; i<arr.length; i++){

console.log( arr[i] );

}

### JS数组

Array 对象用于在单个的变量中存储多个值。

(1)通过Array构造函数创建数组

例如，声明一个空数组：

var arr1 = new Array();

例如，声明一个指定初始值的数组：

var arr2 = new Array(88,"Hello" ,true , 100);

(2)通过数组直接量创建数组

例如，声明一个空数组：

var arr1 = [];

例如，声明一个指定初始值的数组：

var arr2 = [88,"Hello" ,true , 100];

提示：数组的length属性可以返回数组的长度

### JS函数

函数就是一个具有功能的代码块, 可以反复调用

函数就是包裹在花括号中的代码块，前面使用了关键词 function。

声明函数的语法:

function funName([参数1,参数2,…]){

需要执行的代码

}

调用函数：funName([参数1,参数2,…])

示例：声明一个函数, 接收两个数值并计算两个数值的乘积，并返回结果：

function getSum(a,b){

return a\*b;

}

fn(3\*7);//返回21

### 综合练习

(自己完成)声明一个函数fn，在函数中实现：遍历指定的两个数值(例如1和100)之间的整数，将是3的倍数的数值存入到一个数组中，并将数组返回。

1、声明fn函数

function fn(x,y){

var arr = [];

for(var i=x,j=0;i<y;i++){

if(i%3==0){

arr[j] = i;

j++;

}

}

return arr;

}

2、调用fn函数, 获取1~100之间是3的倍数的数值组成的数组

var arr = fn(1,100);

3、遍历数组中的元素, 输出在网页上(提示: document.write("输出的内容") )

for(var i=0;i<arr.length;i++){

document.write(arr[i]+" ");

}

## DOM操作

### 案例：电灯开关

点击电灯可以实现开/关灯，代码实现如下：

<script>

/\*\* 练习：点击按钮，开灯或关灯 \*/

**var** flag = **false**; //false是默认状态,表示关灯

**function** changeImg(){

//1.获取id为img1的元素,返回表示当前元素的js对象

**var** img1 = document.getElementById("img1");

**if**( flag ){ //true表示当前为开灯

img1.src = "imgs/off.gif";

flag = **false**;

}**else**{ //false表示当前为关灯

//2.通过img的src属性修改图片

img1.src = "imgs/on.gif";

flag = **true**;

}

}

</script>

</head>

<body>

<input type=*"button"* value=*"开/关灯"*

onclick="changeImg()" /> <br/><br/>

<img id=*"img1"* src=*"imgs/off.gif"*/>

</body>

### 案例：增删改元素

点击网页中的按钮对html元素进行操作

<script type="text/javascript">

/\* 1、添加元素：创建一个div添加到body中 \*/

**function** addNode(){

//1.创建一个div元素,返回表示div元素的js对象

**var** newDiv = document.createElement("div");

//>>为div设置内容(innerHTML)

newDiv.innerHTML = "我是一个新来的div~";

//2.获取文档中的body元素

//var body = document.getElementsByTagName("body")[0];

**var** body = document.body;

//3.通过父元素添加子元素

body.appendChild( newDiv );

}

/\* 2、删除元素：删除id为div\_2的元素 \*/

**function** deleteNode(){

//1.获取将要被删除的元素(div\_2)

**var** div2 = document.getElementById("div\_2");

//2.获取被删元素的父元素(body)

**var** parent = div2.parentNode;

//3.通过父元素删除子元素

parent.removeChild( div2 );

}

/\* 3、将div\_3的内容更新为当前时间

new Date().toLocaleString(); \*/

**function** updateNode(){

//1.获取id为div\_3的元素

**var** div3 = document.getElementById("div\_3");

//2.获取表示当前时间的字符串

**var** dateStr = **new** Date().toLocaleString();

//3.将字符串设置给div的内容

div3.innerHTML = dateStr;

}

</script>

</head>

<body>

<input type="button" onclick="addNode()"

value="创建一个div添加到body中"/>

<input type="button" onclick="deleteNode()"

value="删除id为div\_2的元素"/>

<input type="button" onclick="updateNode()"

value="将div\_3的内容更新为当前时间"/><hr/>

<div id="div\_1">

div\_1

</div>

<div id="div\_2">

div\_2

</div>

<div id="div\_3">

div\_3

</div>

<div id="div\_4">

div\_4

</div>

</body>

### 案例：网页换肤

<script type="text/javascript">

/\*\* 练习1：执行下面的函数，切换字体大小 \*/

**function** resize( selector ){ //接收的是选择器的名字

//1.获取id为newstext的元素

**var** div = document.getElementById("newstext");

//2.为div设置class值

div.className = selector;

}

/\*\* 练习2：执行下面的函数，为页面切换不同的皮肤 \*/

**var** arr = ["css/red.css", "css/green.css", "css/blue.css", "css/pink.css"];

**var** index = 0;

**function** changeStyle(){

//1.获取id为link的元素(link标签)

**var** link = document.getElementById("link");

link.href = arr[index];

index++;

**if**( index == arr.length ){

index = 0;

}

}

</script>

</head>

<body>

<h2>达内时代科技集团简介</h2>

<div id=*"change-font"* >

<!-- javascript:void(0): 用于阻止超链接跳转 -->

<a href=*"javascript:void(0)"*

onclick="resize('super-min')">超小字体</a>

<a href=*"javascript:void(0)"*

onclick="resize('min')">小字体</a>

<a href=*"javascript:void(0)"*

onclick="resize('middle')">中字体</a>

<a href=*"javascript:void(0)"*

onclick="resize('max')">大字体</a>

<a href=*"javascript:void(0)"*

onclick="resize('super-max')">超大字体</a>

<a href=*"javascript:void(0)"*

onclick="changeStyle()">换肤</a>

</div>

<hr/>

<div id=*"newstext"* class=*"middle"*>

...

</div>

</body>

### 总结：JS获取元素

document.**getElementById**( id值 )

-- 通过元素的id值，获取一个元素。返回的是表示该元素的js对象。

document.**getElementsByTagName**( 元素名 )

-- 通过元素名称获取当前文档中的所有指定名称的元素，返回的是一个数组，其中包含了所有指定名称的元素。

document.**body**

-- 获取当前文档中的body元素

ele.**parentNode**

-- 获取当前元素的父元素。obj表示当前元素

### 总结：JS增删改元素

document.**createElement**( 元素名称 )

-- 根据元素名称创建指定名称的元素，返回的是表示新创建元素的js对象

parent.**appendChild**( child )

-- 通过父元素添加子元素，其中parent表示父元素，child表示子元素

parent.**removeChild**( child )

-- 通过父元素删除子元素，其中parent表示父元素，child表示子元素

ele.**innerHTML**

-- 获取当前元素的html内容(从开始标签到结束标签之间的所有内容)

-- 或者, 还可以设置当前元素的html内容(如果元素内部有内容，将会覆盖原有内容)

# Day07 jQuery

## Query简介

### 什么是jQuery(了解)

jQuery是一门轻量的、免费开源的JS函数库，可以极大的简化JS代码。

jQuery的核心思想：“写得更少,但做得更多”

轻量的：代码或项目对一门技术的依赖程度，依赖程度越低，这个技术越轻；反之，依赖程度越高，这个技术越重，推荐使用轻量级的技术。

### jQuery的优势(了解)

(1)可以极大的简化JS代码

(2)可以像css选择器一样获取元素

$("#div1") – 获取id为div1的元素

$(".c1") – 获取所有class值为c1的元素

$("#div1 span") – 获取id值为div1元素内部的所有span元素

(3)可以通过修改css样式控制页面的效果

//为所有div设置背景(jquery方式)

$("div").css("background","red");

//为所有div设置背景(js方式)

var arr = Document.getElementsByTagName("div");

for(var i=0;i<arr.length;i++){

arr[i].style.background = “red”;

}

(4)可以兼容常用浏览器 (谷歌/火狐/苹果/欧朋)

### jQuery引入

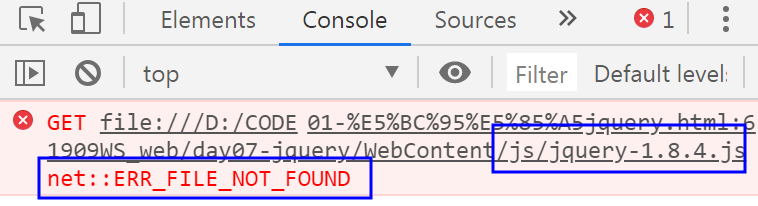
jQuery函数库其实就是一个JS文件(其中封装了很多js函数)

如果要使用jQuery,将jQuery函数库文件引入即可!

<!-- 引入jQuery库文件 -->

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

在引入时,如果路径书写不正确,浏览器的控制台会报如下错误:



### 文档就绪事件

<script type="text/javascript">

/\* 获取id为d1的元素的内容 \*/

//1.获取id为d1的元素

**var** h1 = document.getElementById("d1");

//2.获取h1的内容(innerHTML)

alert( h1.innerHTML );

</script>

<body>

<h1 id="d1">引入jquery示例...</h1>

<body>

问题描述：上面的代码执行时，会抱一个错误：



原因描述：在执行获取id为d1的元素时, h1元素还没有被浏览器加载到，所以获取不到h1元素。

**解决方式一：**可以将js代码放到h1元素的下面，让浏览器先加载h1元素，再执行获取h1的代码。就可以正常执行。

**解决方式二：**

将获取h1元素的代码放在文档就绪事件函数中，文档就绪事件函数会在浏览器加载完整个html文档后立即执行！

/\* (js版)文档就绪事件函数: 当浏览器加载完整个html网页后,

会立即执行该函数 \*/

window.onload = **function**(){

//1.获取id为d1的元素

**var** h1 = document.getElementById("d1");

//2.获取h1的内容(innerHTML)

alert( h1.innerHTML );

}

/\* (jquery版)文档就绪事件函数: 当浏览器加载完整个html网页后,

会立即执行该函数 \*/

$( **function**(){

//1.获取id为d1的元素

**var** h1 = document.getElementById("d1");

//2.获取h1的内容(innerHTML)

alert( h1.innerHTML );

} );

**简写形式为:**

$(function(){

//在整个文档被加载完后立即执行...

});

**完整写法为:**

$(document).ready(function(){

//在整个文档被加载完后立即执行...

});

文档就绪事件函数的应用场景：当获取元素时，为了避免因为元素还没有被加载，导致元素获取不到时，可以将获取元素的代码放在文档就绪事件函数中。

## jQuery选择器(重点掌握)

### 基本选择器

**1、元素名(标签名)选择器**

$("div") – 获取所有div元素

$("span") – 获取所有的span元素

**2、类选择器**

$(".c1") – 获取所有class值为c1的元素

$("div.c1") – 获取所有class值为c1的div元素

**3、ID选择器**

$("#one") – 获取id为one的元素

### 层级选择器

**1、后代选择器**

$("#one span") – 匹配id为one的元素内部的所有span元素

**2、相邻兄弟选择器**

$("#one+span") – 匹配id为one的元素后面紧邻的span兄弟元素

等价于:

$("#one").next("span") – 匹配id为one的元素后面紧邻的span兄弟元素

$("#one").prev("span") – 匹配id为one的元素前面紧邻的span兄弟元素

**3、$("#two~div")**

$("#one~span") – 匹配id为one的元素后面所有的span兄弟元素

$("#one").nextAll("span") – 匹配id为one的元素后面所有的span兄弟元素

$("#one").prevAll("span") – 匹配id为one的元素前面所有的span兄弟元素

$("#one").siblings("span") – 匹配id为one的元素前、后所有的span兄弟元素

### 基本过滤选择器

**1、:first | :last**

**2、:eq(n)**

$("div:first") – 匹配所有div中的第一个div元素

$("div:last") – 匹配所有div中的最后一个div元素

$("div:eq(0)") – 匹配所有div中的第一个div元素

$("div").eq(0) – 匹配所有div中的第一个div元素

$("div:eq(-1)") – 匹配所有div中的最后一个div元素

$("div").eq(-1) – 匹配所有div中的最后一个div元素

## 综合案例

### 全选或全不选

/\*\* 练习1：点击全选设置 商品所在的行全选或者全不选 \*/

**function** checkAll(){

//获取全选复选框的选中状态(选中或取消)

//<input checked="checked"/>

**var** status = $("#all").prop("checked");

console.log( status );

//将全选框的状态设置给所有的复选框

$("input[type='checkbox']").prop("checked", status);

}

/\* 为所有的普通复选框绑定点击事件，每次点击普通复选框都要

\* 检查所有的普通复选框是否全都选中了，如果是，就选中全选

\* 复选框，否则，就取消选中全选复选框。 \*/

$(**function**(){

$("input[name='item']").click(**function**(){

//获取所有普通复选框,并遍历其中的每一个复选框

**var** isCheckAll = **true**; //假设所有普通复选框都选中了

$("input[name='item']").each(**function**(){

//判断所有普通复选框是否全都选中了

**var** status = $(**this**).prop("checked");

**if**( !status ){

isCheckAll = **false**;//至少有一个没选中

}

});

**if**(isCheckAll){

//循环结束后, isCheckAll=true, 所有普通复选框都选中了

$("#all").prop("checked", **true**);

}**else**{

//isCheckAll=false, 所有普通复选框没有全部选中

$("#all").prop("checked", **false**);

}

//$("#all").prop("checked", isCheckAll);

});

});

### 创建表格元素

**练习1：创建单行单列的表格**

**function** createTable1(){

**var** $tab = $("<table></table>");

**var** $tr = $("<tr></tr>");

**var** $td = $("<td></td>");

$td.html("我是td!")

//将td添加到tr内部

$tr.append( $td );

//将tr添加到table内部

$tab.append( $tr );

//将table添加到网页中

$("body").append( $tab );

//$("body").append("<table><tr><td>单元格</td></tr></table>")

}

**练习2.1：创建单行6列的表格**

**function** createTable2(){

**var** $tab = $("<table></table>");

**var** $tr = $("<tr></tr>");

**for**(**var** i=0; i<6;i++){

**var** $td = $("<td></td>");

$td.html("我是td!")

//将td添加到tr内部

$tr.append( $td );

}

//将tr添加到table内部

$tab.append( $tr );

//将table添加到网页中

$("body").append( $tab );

}

**练习2.2：创建5行6列的表格**

**function** createTable2(){

**var** $tab = $("<table></table>");

**for**(**var** i=0;i<5;i++){

**var** $tr = $("<tr></tr>");

**for**(**var** j=0; j<6;j++){

**var** $td = $("<td></td>");

$td.html("我是td!")

//将td添加到tr内部

$tr.append( $td );

}

//将tr添加到table内部

$tab.append( $tr );

}

//将table添加到网页中

$("body").append( $tab );

}

**练习3：创建指定行和列的表格**

**function** createTable3(){

//var rows = $("#rows").val();

//var cosl = $("#cosl").val();

**var** rows = document.getElementById("rows").value;

**var** cols = document.getElementById("cols").value;

**var** $tab = $("<table></table>");

**for**(**var** i=0;i<rows;i++){

**var** $tr = $("<tr></tr>");

**for**(**var** j=0; j<cols;j++){

**var** $td = $("<td></td>");

$td.html("我是td!")

//将td添加到tr内部

$tr.append( $td );

}

//将tr添加到table内部

$tab.append( $tr );

}

//将table添加到网页中

$("body").append( $tab );

}

### 仿QQ好友列表

/\*\* 通过jQuery实现仿QQ好友列表 \*/

**function** openDiv( thisobj ){

//当前被点击的span元素(分组名称)

//$(thisobj).next("div").css("display", "block");

//$(thisobj).next("div").show();

/\*

//获取当前分组的状态(展开? 关闭?)

var status = $(thisobj).next("div").css("display");

if(status == "block"){ // 展开

$(thisobj).next("div").css("display", "none");

}else{

$(thisobj).next("div").css("display", "block");

} \*/

//获取其他三个分组内的div元素, 设置隐藏(关闭其他三个分组)

$(thisobj).parents("tr").siblings("tr").find("div").hide();

//获取被点击分组下的div元素, 切换元素的显示状态

$(thisobj).next("div").slideToggle(100);

}

## 总结

### html元素操作

**1、创建元素**

$("<div></div>") -- 创建一个div元素，返回的是一个表示div元素的jQuery对象

$("<div>xxxx</div>") -- 创建一个包含内容的div元素，返回的是一个表示div元素的jQuery对象

**2、添加子元素**

$parent.append( $child ) -- 父元素调用方法添加子元素

$("body").append( "<div>我是新来的...</div>" );

-- 往body元素内部追加一个div子元素

**3、删除元素**

$("div").remove() -- 删除所有的div元素

**4、替换元素**

$("div").replaceWith("<p>我是来替换的…</p>")

### html元素内容和值的操作

**1、html()函数 (类似于js中的innerHTML属性)**

-- 用于获取或设置元素的内容，比如为div、span、p、h1等元素设置内容

$("div").html() -- 获取所有div中的第一个div的内容

$("div").html("xxxx") -- 为所有div设置内容

**2、text()函数 (类似于js中的innerText属性)**

-- 用于获取或设置元素的文本内容

$("div").text() -- 获取所有div中的所有文本内容

$("div").text("xxxx") -- 为所有div设置文本内容

<div>

这是一个div元素

<span>这是一个span元素</span>

</div>

**3、val()函数** (类似于js中的value属性)

-- 获取或设置表单项元素的value值(input/select/option/textarea)

$("input").val() -- 获取所有input元素中的第一个input元素的value值

$("input").val(值) -- 为所有的input元素设置value值

### 元素属性和css属性操作

<input type="text" name="username" id="user"/>

**1、prop()函数** -- 用于获取或设置元素的属性值

在jQuery1.6版本之后才有这个函数，1.6之前版本的jQuery可以使用attr()函数

$("input[type='checkbox']").**prop**("checked")

-- 获取input复选框的选中状态, 返回true表示复选框为选中状态, 返回false表示复选框为取消选中状态

$("input[type='checkbox']").**prop**("checked", true)

-- 设置所匹配的复选框元素为选中状态

**2、css()函数** -- 用于获取或设置元素的css属性值

$("#div1").css("width") -- 获取id为div1元素的宽度

$("#div1").css("width","200px") -- 设置id为div1元素的宽度为200px

$("#div1").css({

"width" : "200px",

"height" : "150px",

"border" : "5px solid red",

"background" : "pink"

})

--设置id为div1元素的宽度为200px、高度为150px、边框以及背景颜色等样式

### 其他函数

**1、show()/hide()**

**show() -- 设置元素由隐藏变为显示**

**hide() -- 设置元素由显示变为隐藏**

$("div").show() -- 设置所有的div元素为显示

等价于:

$("div").css("display", "block");

$("div").hide() -- 设置所有的div元素为隐藏

等价于:

$("div").css("display", "none")

**2、toggle()/slideToggle()**

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.

## jQuery事件绑定

以点击事件为例，为元素绑定点击事件的方式为:

**方式1(js版)：**

<script>

**function** **fn**(){

alert("input按钮被点击了...");

}

</script>

<body>

<input **onclick="fn()"** type=*"button"* value=*"点我~!"* />

</body>

**方式2(js版)：**

<script>

window.onload = **function**(){

//获取id为btn的元素

**var** btn = document.getElementById("**btn**");

**btn.onclick = function(){**

alert("input按钮被点击了...");

**}**

}

</script>

</head>

<body>

<input id=*"****btn****"* type=*"button"* value=*"点我~!"* />

</body>

**方式3(jQuery版)：**

<script>

$(function(){

//当点击btn按钮时,触发点击事件执行其中的函数

$("#btn").click( **function**(){

alert("input按钮被点击了...");

});

});

</script>

<body>

<input id=*"****btn****"* type=*"button"* value=*"点我~!"* />

</body>

# Day08 tomcat、HTTP

## 服务器概述

### 什么是服务器?

服务器：分为 服务器硬件 和 服务器软件。在硬件服务器（一台计算机）安装了服务器软件，就可以对外界提供服务，比如：让其他计算机上的浏览器来进行访问。

(1)服务器硬件: 是一台在互联网上具有独立IP地址的计算机，比如我们面前的计算机也可以作为服务器使用。

(2)服务器软件: 就是一个计算机程序(软件), 比如在mysql服务器软件、tomcat服务器软件。

### 什么是Web服务器?

(1)Web服务器是可以向“发出请求的浏览器提供文档”的程序。 比如在访问百度时，其实就是在访问百度的服务器。

(2)Web服务器是指驻留于因特网上某种类型计算机的程序。当浏览器（客户端）连接到服务器，请求服务器上的文件时，服务器将处理该请求并将文件发送到该浏览器上，附带的信息会告诉浏览器如何查看该文件（即文件类型）。



tomcat就是一个web服务器，是由apache组织提供的一款服务器软件，特点是小巧灵活，免费开源，简单易用。

## tomcat下载、安装、启动、配置

### 下载tomcat服务器

下载地址：http://tomcat.apache.org/

tomcat有很多版本，有解压版 和 安装版，还分windows （还分为32位和64位版）和 linux版，根据自己的需求，选择对应的版本下载。

tomcat服务器运行需要jdk的支持，版本对应为：

tomcat5 需要jdk4以上支持

tomcat6 需要jdk5以上支持

tomcat7 需要jdk6以上支持

tomcat8 需要jdk7以上支持

### 安装、启动tomcat服务器

**1、安装tomcat服务器**

绿色版解压之后就可以使用(原则：安装的路径中不要包含中文和空格)



解压后还需要配置JAVA\_HOME环境变量，指向jdk的根目录，指定tomcat启动时使用哪个jdk。

**2、启动tomcat服务器**

启动tomcat服务器时，tomcat服务器程序会去寻找一个JAVA\_HOME环境变量，该变量配置JDK所在位置。（tomcat服务器是Java语言开发的，需要JDK支持才能运行）

**如何配置JAVA\_HOME环境变量：**

变量名: JAVA\_HOME

变量值: C:\Program Files\Java\jdk1.8.0\_45

-----------------------------------------------------

配置Path变量的两种方式：

方式一：

Path=**C:\Program Files\Java\jdk1.8.0\_45\bin;**xxx;xxx;xxx;

方式二：

JAVA\_HOME=C:\Program Files\Java\jdk1.8.0\_45

Path=%JAVA\_HOME%\bin;xxx;xxx;xxx;

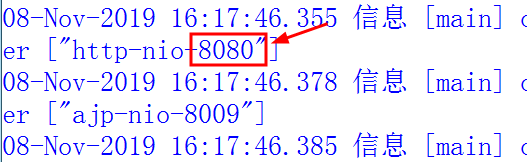
-----------------------------------------------------

**启动、关闭tomcat服务器：**

通过 [tomcat\_home]/bin/startup.bat 文件可以启动tomcat服务器

通过 [tomcat\_home]/bin/shutdown.bat 文件可以关闭tomcat服务器

在tomcat启动之后，默认会监听8080端口（没有修改的前提下），如图：



可以通过在浏览器中输入以下地址，访问tomcat服务器的主页：

http://localhost:8080

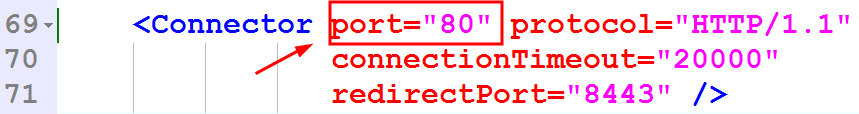
如果访问成功，则说明已经成功的安装了tomcat，并且tomcat启动也成功了。

### 修改tomcat默认端口

tomcat服务器在启动后默认监听8080端口，而8080端口在访问时是不能省略的。

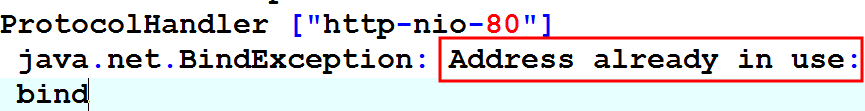
如果想省略端口，只能将8080改为80端口才可以省略。修改访问为：

找到[tomcat\_home]/conf/server.xml文件的69行的Connector标签,将上面的port属性的值改为80,保存文件,重启服务器即可!



**\*\* FAQ端口占用问题\*\*：**

在启动tomcat服务器时，可能会遇到端口占用问题，如下图：



原因分析:

**情况一：**可能是之前的tomcat服务器没有完全关闭，仍然在占用80端口，导致服务器再次启动时，启动失败。

解决方式：运行shutdown.bat文件，将tomcat按照正常流程再关闭一次即可。如果再次启动服务器成功，说明问题已解决，否则看情况二。

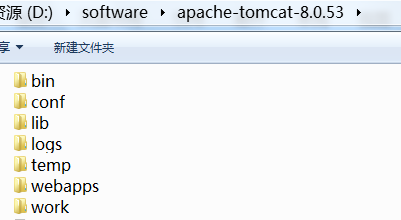
**情况二：**可能是其他程序占用了80端口，导致服务器启动失败。

解决方式：打开一个cmd窗口，通过 netstat -ano 命令查看当前系统中活动的进程，找到80端口对应的进程编号（PID），根据进程编号将进程结束即可！

taskkill /f /pid 进程编号

## tomcat目录结构

### tomcat目录结构介绍



tomcat服务器安装根目录下有很多子目录，这些目录的作用是：

(1)bin：存放了tomcat服务器中的可执行的批处理文件

(2)conf：存放了tomcat相关的配置文件(其中的server.xml是tomcat服务器核心配置文件)

(3)lib：是tomcat服务器运行时所依赖的jar包的存放目录(servlet的jar包、JSP的jar包)

(4)logs：是tomcat服务器运行时产生的日志文件的存放目录

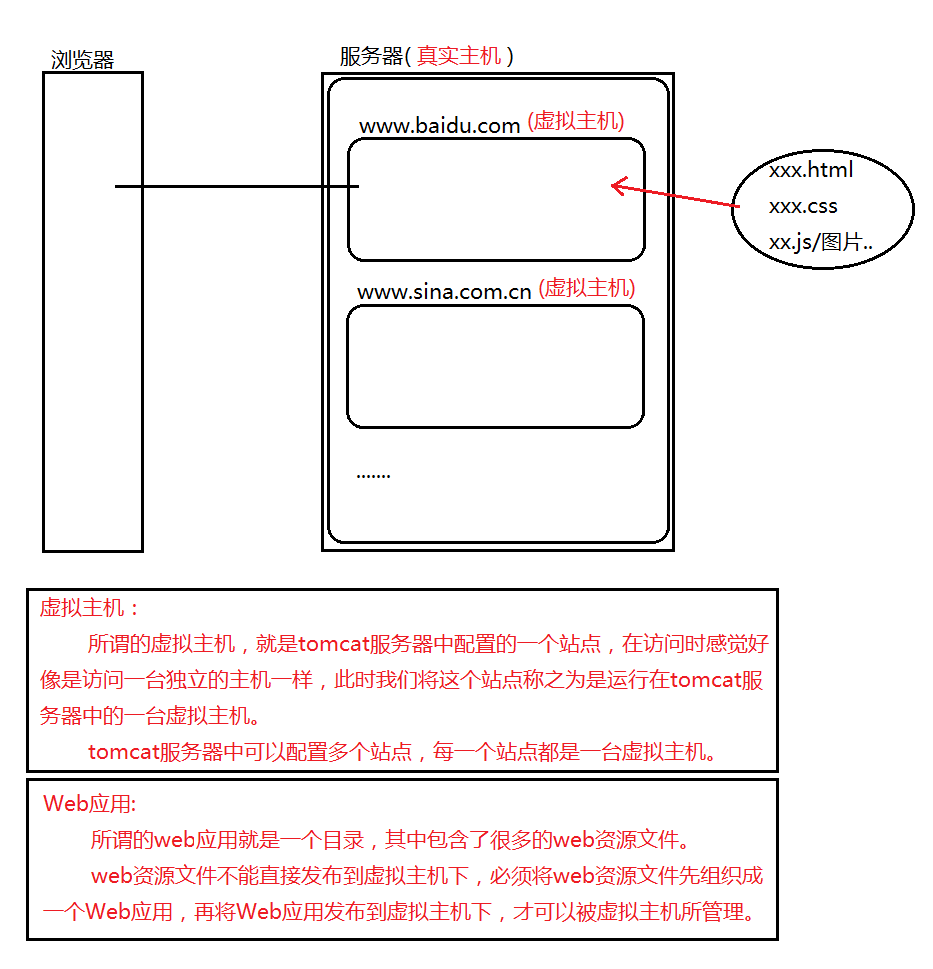
(5)temp：临时文件的存放目录，tomcat服务器会自己管理

(6)work：tomcat服务器在运行时产生的工作文件(JSP翻译后的Servlet文件、Session序列化后的文件等)

(7)webapps目录：是localhost虚拟主机默认管理的目录，可以将web应用发布（存放）到该目录下，就可以通过localhost主机访问该web应用中的资源了。

可以简单的理解为：这个webapps目录，就是一个资源的发布目录，将资源文件和目录发布到webapps目录下，就可以通过浏览器访问这些文件。

## 虚拟主机和Web应用

****

## web应用

* + 1. **web应用的目录结构**

Web应用本身就是一个目录，该目录下可以包含其他的目录和文件

news(Web应用/目录)

|

|-- 其他目录: 放在其他目录中的资源文件, 可以被浏览器直接访问。

|-- WEB-INF目录：这是一个受保护的目录，放在这个目录下的资源文件，不可以被浏览器直接访问（不是不能，是不能直接访问）。

|-- classes目录：存放编译后的class文件

|-- lib目录：存放web应用所依赖的jar包

|-- web.xml文件：web应用的核心配置文件。

* + 1. **部署web应用到虚拟主机中**

将Web应用的目录拷贝到虚拟主机所管理的目录下即可。

例如：将news应用的目录，拷贝到webapps目录下，就相当于将news发布到了localhost主机中

这种发布方式不用重启服务器就可以直接访问。

## 扩展内容（了解）

### 配置缺省的(默认的)WEB应用

在将news配置为缺省的Web应用之前，访问news下的hello.html的路径为:

<http://localhost/news/hello.html>

如果将news配置为缺省的Web应用，再去访问news下的hello.html的路径为:

<http://localhost/hello.html>

因为在访问缺省Web应用下的资源时，在访问路径中不用写缺省的Web应用名字。

如何将Web应用配置为缺省的web应用??

将Web应用的名字改为一个大写的ROOT即可!!

### 配置WEB应用的主页

在没有将hello.html配置为主页之前，访问hello.html的路径为:

<http://localhost/hello.html>

如果将hello.html配置为主页，访问hello.html的路径为:

<http://localhost>

在访问一个主页时，主页的路径是可以省略的

**如何将hello.html配置当前应用的主页???**

在web.xml文件中的跟标签内部添加如下内容:

<welcome-file-list>

<welcome-file>hello.html</welcome-file>

</welcome-file-list>

* + 1. **打war包**

war、jar都是java中的一种压缩包。类似于windows的zip、rar包

可以将Web应用中的所有内容打成一个war包，这样做的好处是：

(1)将web应用中的所有目录和文件打成一个war包,便于传输

(2)将web应用打成一个war包,体积会减少,可以节省带宽

(3)war包发布到服务器后, 会自动解压发布.

**如何打一个war包:**

进入web应用的目录，将web应用下的所有目录和文件全部选中，添加到一个xxx.zip压缩包中（注意不能是rar包，只能是zip），再将文件的后缀名改为war即可！！

**注意事项：**

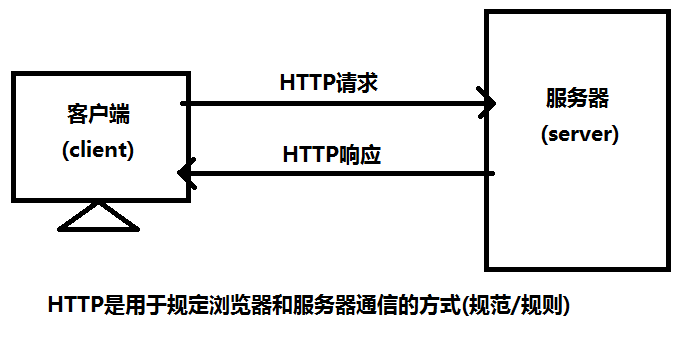
(1)打成的war包的名字不能是中文的，且不能和已发布的应用名重名！

(2)war包中不能包含中文的目录或文件。

如果出现以上问题，war包将不会自动解压！

## HTTP协议概述

### 什么是HTTP协议?



HTTP是用于规定浏览器和服务器通信的方式。

主要规定了浏览器向服务器发送的请求信息的格式，

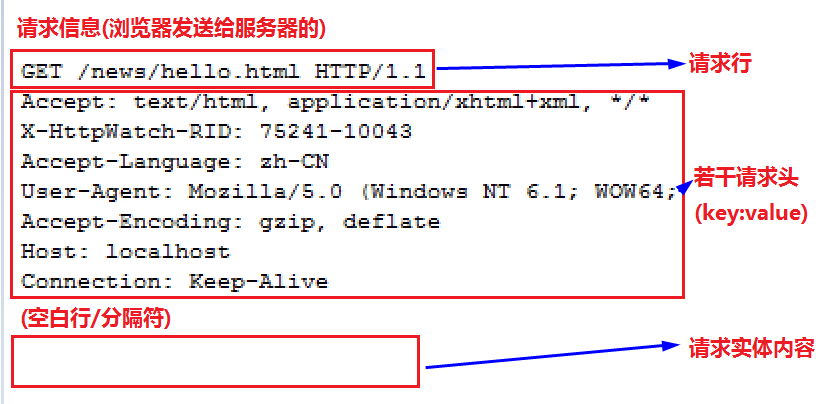
以及规定了服务器向浏览器做出的响应信息的格式。

**\*\* HTTP工作时遵循的基本原则(了解):**

(1)一次请求对应一次响应

(2)请求只能由浏览器发起,服务器只能被动的等待请求,根据请求作出回应.

* 1. **HTTP协议详解**
     1. **HTTP请求**



**1、请求行**

**GET /news/hello.html HTTP/1.1**

**GET:** 提交方式，在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。

**/news/hello.html:** 请求的资源路径，用于通知服务器浏览器访问的是服务器中哪个Web应用以及应用下的哪一个资源文件。

**HTTP/1.1：**请求所遵循的协议和版本

**2、若干请求头**

请求头都是Key-Value结构,例如:

Host : localhost -- 表示浏览器请求的是服务器中的哪一台虚拟主机。

Cookie : xxx=xxx; -- cookie技术中的一个头(后面讲)

xxx : xxx

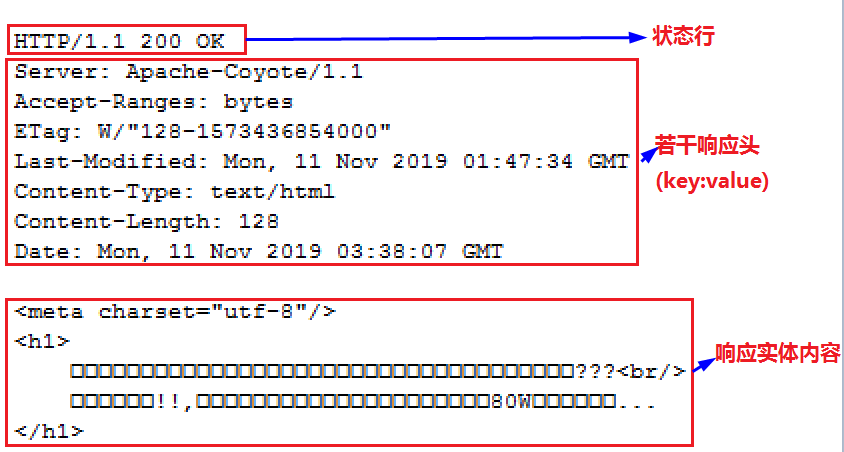
...

**3、请求实体内容**

如果请求方式是GET提交，请求实体不会有内容。

如果请求方式是POST提交，并且请求中携带了数据，请求实体才会有内容

### HTTP响应



**1、状态行**

**HTTP/1.1 200 OK**

HTTP/1.1：表示响应信息所遵循的协议和版本

200：是一个三位是数字，叫做状态码，表示请求处理的结果。常见的状态码有:

200：表示服务器处理请求的结果是成功的！

304/307：表示服务器通知浏览器使用缓存

400：表示请求参数类型不匹配

404：表示客户端请求的资源不存在

500：表示服务器端在处理请求的过程中出现了错误或异常

OK：描述短语，也表示请求处理的结果

200 OK

404 NOT FOUND

500 Internal Server Error

**2、若干响应头**

响应头的格式也是Key-Value结构

**Content-Type:** text/html -- 通知浏览器,服务器响应的数据类型是什么

text/html -- 表示服务器响应是一个html格式的网页,如果响应是一个网页,还可以在值的后面加上一个编码,例如: text/html;charset=utf-8

text/css -- 表示服务器响应是一个css文件

image/gif -- 表示服务器响应的是一个gif格式的图片

...

**Content-Length:** 服务器响应的数据长度,单位是字节

**Set-Cookie:** Cookie技术中的一个头(后面讲解)

...

**3、响应实体内容**

如果浏览器向服务器请求的是一个文件,服务器会将这个文件的内容作为响应实体发送给浏览器.

比如浏览器请求是 hello.html文件,服务器就会将hello.html文件的内容作为响应实体发送给浏览器

localhost/news/hello.html,

**HTTP总结:**

HTTP: 用于规定浏览器和服务器之间通信的方式

请求信息中包含哪几部分内容: 请求行、请求报头、请求实体

响应信息中包含哪几部分内容: 状态行、响应报头、响应实体

请求行中包含什么内容：请求方式、请求资源路径、协议和版本

状态行中包含什么内容：协议和版本、状态码、描述短语

### 内容补充

**问题1：请求方式什么时候是GET提交？什么时候是POST提交？**

只有当使用表单（form）并且在表单上通过method属性指定提交方式为POST时，才是POST提交，其他方式都是GET提交！！(AJAX除外)

**判断以下请求方式是GET还是POST?**

(1)<form action="#"></form> -- GET提交

(2)<form action="#" method="GET"></form> -- GET提交

(3)<form action="#" method="POST"></form> -- POST提交

(4)点击超链接访问服务器，例如： -- GET提交

<a href="http://www.baidu.com">百度一下</a>

(5)直接在浏览器的地址栏中书写URL地址访问服务器 -- GET提交

**问题2：GET提交和POST提交有什么区别？**

(1)GET提交会将请求参数拼接在URL地址的后面，显示在地址栏中，相对不安全

(2)GET提交将参数显示在地址栏中，数据量不能太大，不能超过1K或者是4K

(3)POST提交不会将参数拼接在地址栏后面, 而是通过请求实体将参数发送给服务器, 相对来说更加安全!

(4)POST提交通过请求实体传数据给服务器, 数据量理论上没有限制!

当向服务器传输大量的数据或者传输的数据较为隐私,才会使用POST提交。

更多情况下，还是使用GET提交！！

# Day09 Servlet

## Servlet概述

### 什么是Servlet?

Servlet是SUN公司提供的一门动态Web资源开发技术

Servlet: Server Applet: 服务器端的Java程序

Servlet本质上就是一段Java程序，和普通的Java程序不同是，Servlet程序无法独立运行，必须将Servlet程序放在服务器中，由服务器调用才可以执行！！

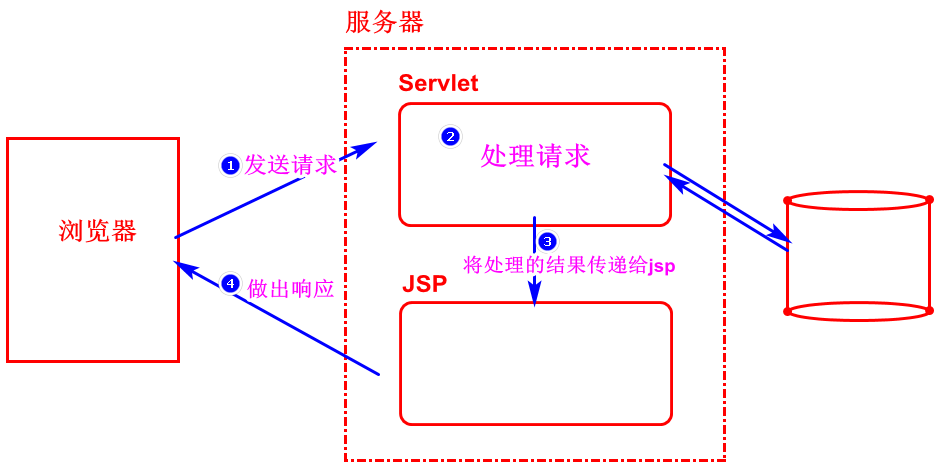
静态Web资源: 任何人在任何条件下访问的效果都是相同的,这种叫做静态Web资源,

html、css、js、flash

动态Web资源: 在不同的条件下访问时，看到的是不同的效果。

Servlet、JSP、PHP、.NET

运行在服务器端的Servlet程序，其作用是什么？



总结：运行在服务器中的Servlet程序的作用是：

在服务器将请求接收过来后，对请求进行处理，即**处理请求!**

## 开发Servlet程序

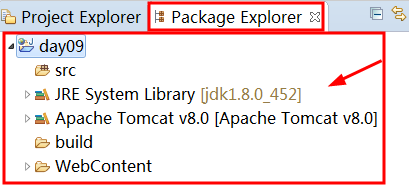
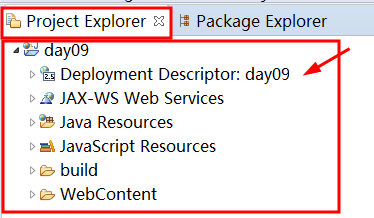
### 开发Servlet程序的步骤

第一步: 写一个类，实现一个Servlet接口，并实现其中的方法

直接实现Servlet接口或者实现该接口的子类。

第二步: 在web应用的web.xml文件中配置Servlet程序对外访问的路径。

### 使用Eclipse创建Web项目

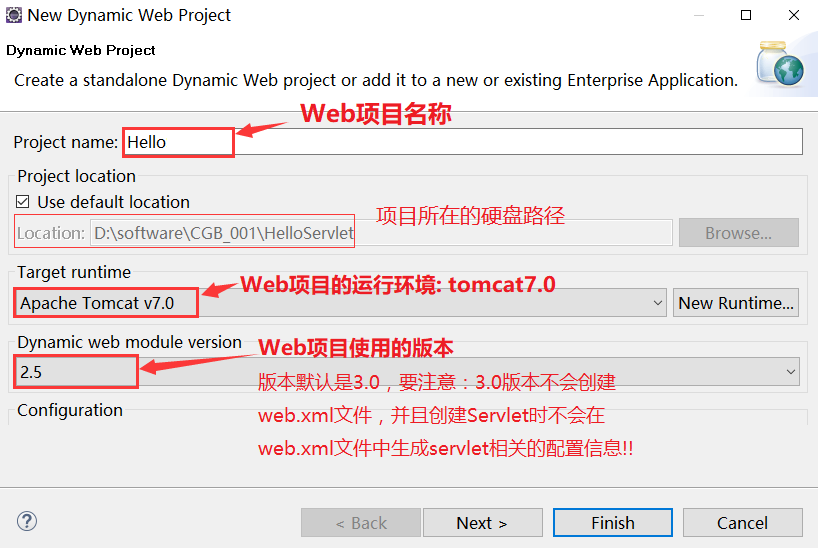


以上是Web项目在工程视图(Project)和包视图（package）下结构，推荐使用包视图！

**1、创建一个Web工程:** 在左侧窗口中, 点击**鼠标右键** ---> **New** ---> **Dynamic Web Project**。



2、接着会弹出如下窗口:



**注意：**

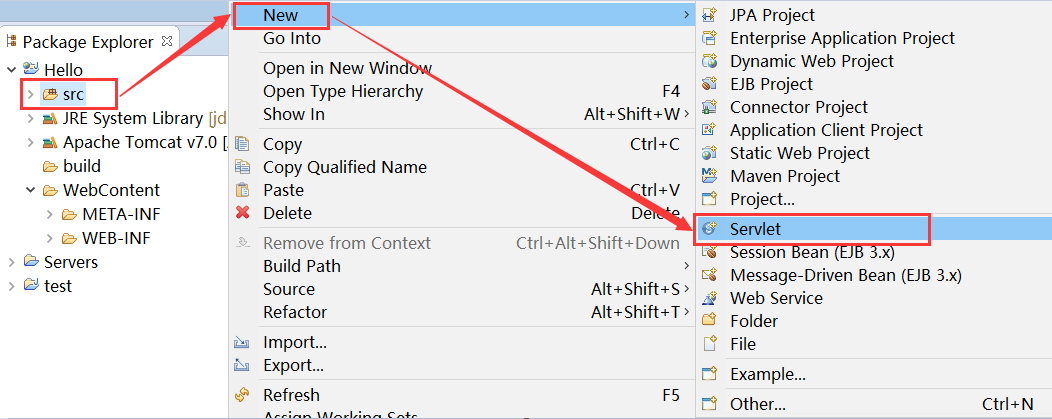
(1) 3.0版本不会创建web.xml文件, 并且创建Servlet时也不会在web.xml文件中生成Servlet相关的配置信息, 记得改为2.5。

(2) Target runtime选项中如果没有可选的服务器，可点击右侧的"New Runtime…"进行配置。

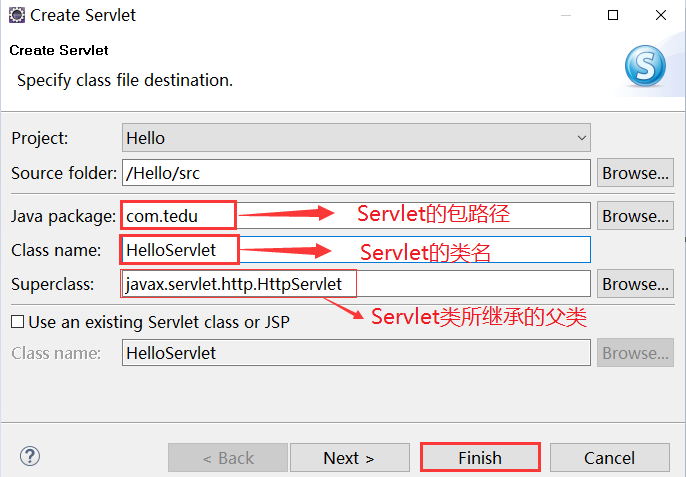
详细操作步骤在《5.2配置Target runtime（Web项目运行环境）》

### 使用Eclipse创建Servlet

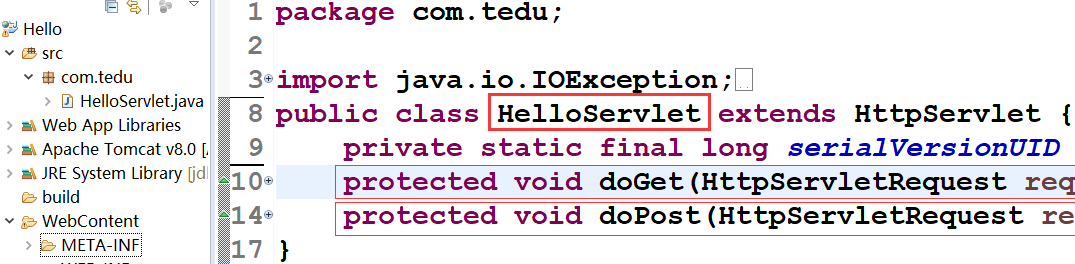
1、选中项目中的src目录，鼠标右键 ---> New ---> Servlet



2、在弹出的窗口中，根据提示填写内容：



3、点击finish即可完成Servlet创建过程, 创建好的Servlet如下:



通过Eclipse创建Servlet，默认继承HttpServlet。由于HttpServlet也是Servlet接口的子类，让HelloServlet继承HttpServlet，相当于间接实现了Servlet接口。

继承HttpServlet类，默认会覆盖doGet方法和doPost方法，两个方法的作用为：

**\* doGet方法：**当浏览器发送请求的方式为GET提交时, 将会调用doGet方法来处理请求

**\* doPost方法：**当浏览器发送请求的方式为POST提交时, 将会调用doPost方法来处理请求

提示：如果当GET提交和POST提交处理代码相同时，可以将代码写在其中一个方法里（例如写在doGet中），并在另外一个方法（例如doPost）中调这个方法。这样一来,不管是GET提交还是POST提交,最终doGet方法都会执行,都会对请求进行处理!!

### Servlet在web.xml中的配置

在通过Eclipse创建Servlet时，会自动在web.xml文件中进行Servlet相关信息的配置

（注意：如果是复制Servlet类文件，但配置信息不会跟着复制，需要自己手动添加配置，否则复制的Servlet将无法访问！）

<**servlet**>

<servlet-name>HelloServlet</servlet-name>

**<servlet-class>com.tedu.HelloServlet</servlet-class>**

</**servlet**>

<**servlet-mapping**>

<servlet-name>HelloServlet</servlet-name>

**<url-pattern>/HelloServlet</url-pattern>**

</**servlet-mapping**>

**关于上面的配置信息：**

**a)** Eclipse每创建一个Servlet, 就会在web.xml文件中添加两个标签：<servlet>和<servlet-mapping>标签（可以将这两个标签看成一个组的标签）

**b)** <servlet>和<servlet-mapping>标签内都会有一个<servlet-name>标签，标签的内容可以更改，但要求更改后的这两个<servlet-name>标签的内容也必须一致。

**c)** <servlet-class>标签用于配置Servlet类的全路径名(即包名+类名)

需要注意：如果在创建Servlet后修改了Servlet类的名称，这个地方也要一起更改，否则将会出现"ClassNotFoundException" 即类找不到异常

**d)** <url-pattern>标签用于配置浏览器以什么路径访问当前Servlet（即Servlet对外访问的路径），默认的路径是：/类名

例如：上面为HelloServlet配置的<url-pattern>为 **/HelloServlet**，因此我们在浏览器中的访问路径则为：

<http://主机名/web>项目访问路径**/HelloServlet**

### 运行Servlet程序、访问测试

**1、访问Servlet方式一：**

若是第一次运行，需要先创建tomcat服务器，即在Servers窗口中点击链接可创建一个tomcat服务器，且只需创建一次即可！

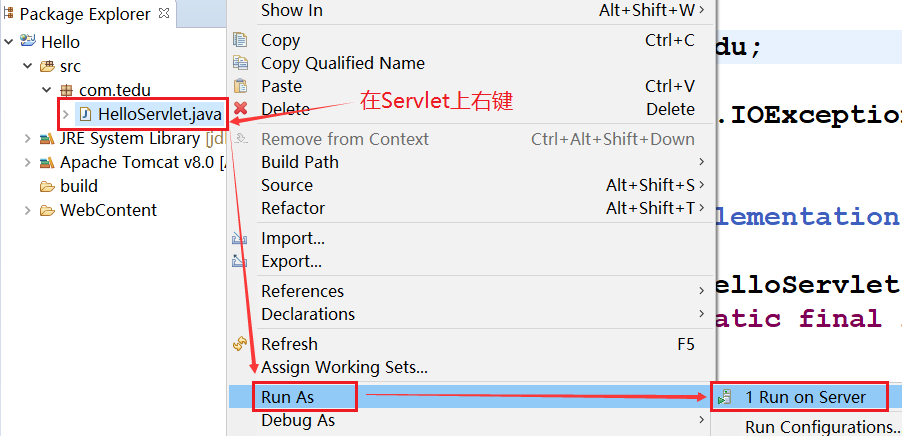
(1)在服务器上右键 --> 点击 "**add and remove"** 将当前web项目发布到服务器中，并点击完成。

(2)启动tomcat服务器：在服务器上右键 Start 即可启动服务器

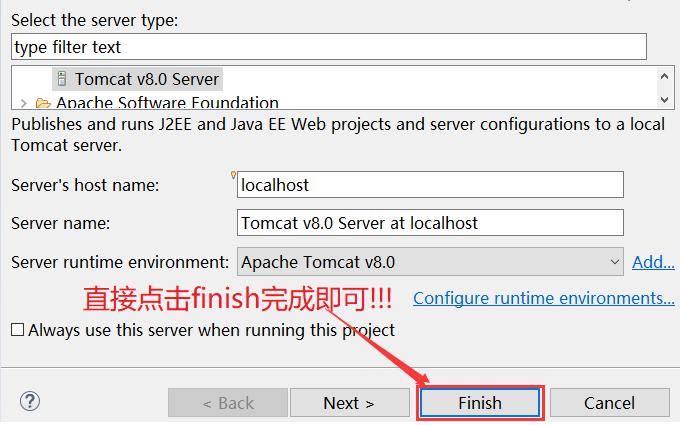
(3)打开本地浏览器，通过路径访问：<http://localhost:8080/项目名称/HelloServlet>即可访问Servlet程序

**2、访问Servlet方式二：**

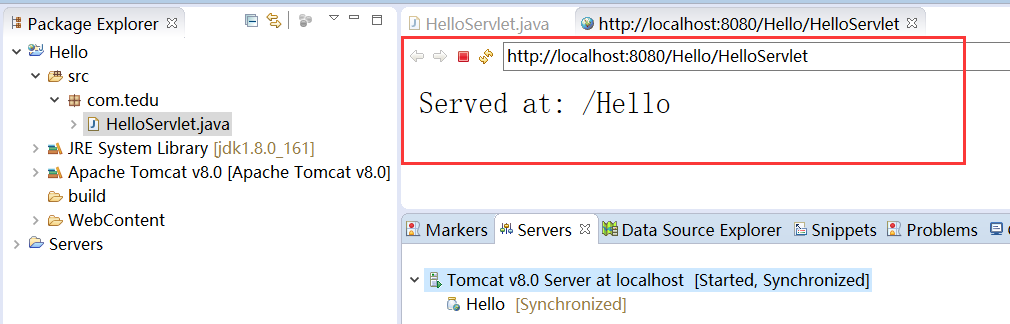
**(1) 在运行的Servlet上点击右键 --**-> **"Run As"** ---> **"1 Run on Server"**



(2) 在弹出的窗口中，直接点击完成即可！！！

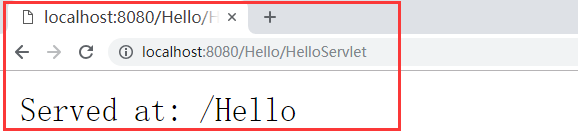


(3) 运行结果如下：



或者打开浏览器，复制上图中的路径：

http://localhost:8080/Hello/HelloServlet，粘贴到浏览器的地址栏中，回车访问：



### Eclipse默认发布Web应用的位置

Tomcat服务器中默认只有一台虚拟主机,叫做localhost主机

而localhost主机发布web应用的位置是webapps。

将day09发布到localhost主机中,但为什么day09目录没有在webapps目录下?

默认情况下,发布一个Web应用到localhost主机中， 只需要将Web应用的目录拷贝到webapps目录下即可完成发布!

而将Eclipse和Tomcat整合之后，通过Eclipse发布一个web应用到tomcat服务器中，发布的路径默认被改成了:

[workspace]\.metadata\.plugins\org.eclipse.wst.server.core\tmp0\wtpwebapps

如果要修改Eclipse默认发布Web应用的目录位置:

(1)关闭服务器,将服务器中的所有应用移除

(2)在服务器上右键 --> clean

(3)双击tomcat服务器,在弹出窗口中找到 Server location, 选择第二个选项

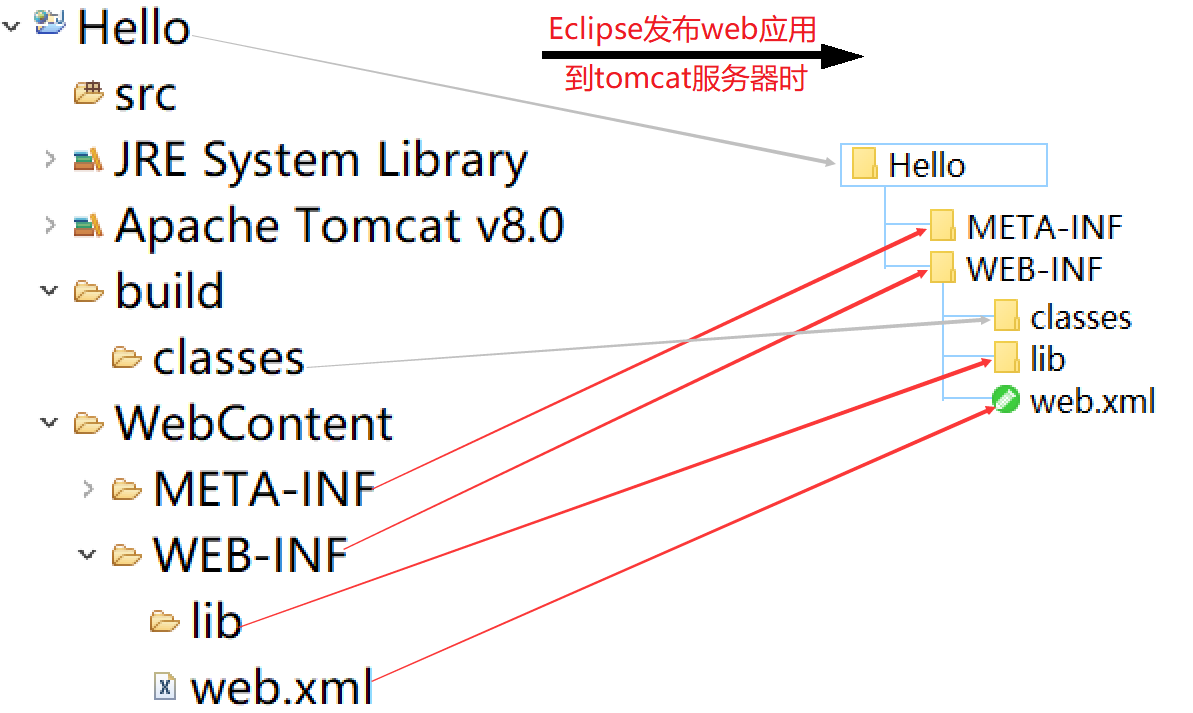
并将下方的Deploy Path改为: webapps

改完后，Ctrl+s保存配置即可!!

### Eclipse如何发布一个Web应用

当通过eclipse将day09项目发布到服务器中,是直接将day09拷贝到服务器中对应的目录下吗?

发布的过程如下：



## Servlet继承关系



Servlet接口 -- 定义了一个Servlet应该具备哪些功能

|

|-- GenericServlet抽象类，实现了Servlet接口，并且实现了接口中大部分的方法，但是没有实现service，service方法是处理请求的核心方法，应该由开发人员自己来实现！

|

|-- HttpServlet抽象类，继承了GenericServlet，并且实现了service方法，在service方法中，判断请求方式，根据不同的请求方式，调用不同的doXxx方法。

因此在开发中，我们只需要写一个类，继承HttpServlet，重写父类中的doGet方法和doPost方法。

当浏览器访问一个Servlet时，首先调用Servlet类的service方法（从HttpServlet类中继承过来的），根据请求方式调用对应的doGet方法或者doPost方法，对请求进行处理！

## Servlet调用过程

通过浏览器访问服务器中的一个Servlet程序，这个Servlet程序是如何执行的？又是如何被调用的？

localhost/Hello/HelloServlet

参考<<Servlet调用过程图>>

## 扩展内容

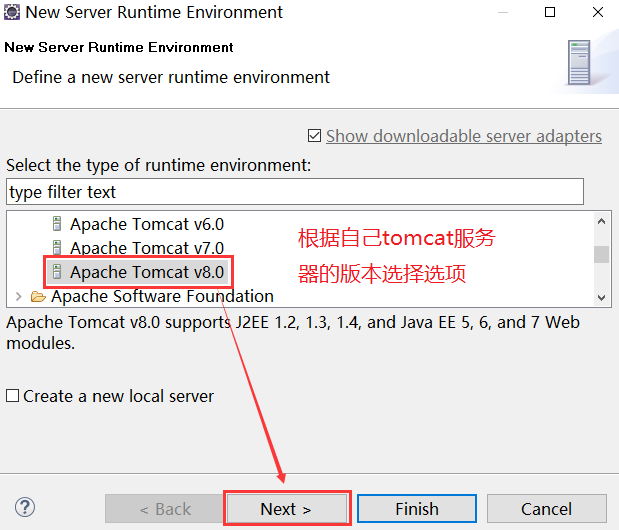
### 配置Target runtime（Web项目运行环境）

1、如果在创建Web项目时，Target runtime选项中没有配置可选的服务器，可以点击右面的选项进行配置



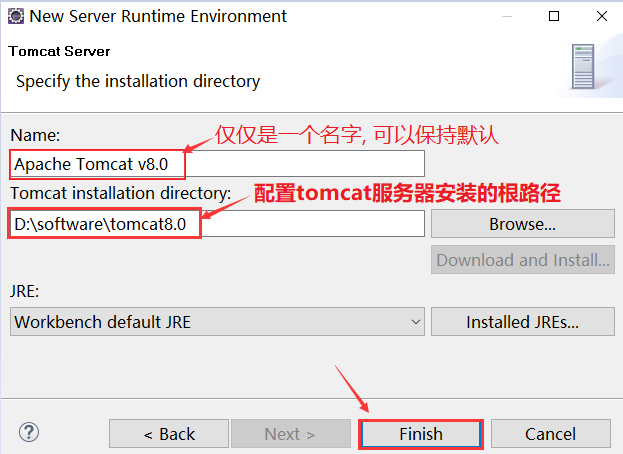
点击后进入下一步操作。

2、接着会出现如下窗口，根据tomcat服务器的版本选择配置：

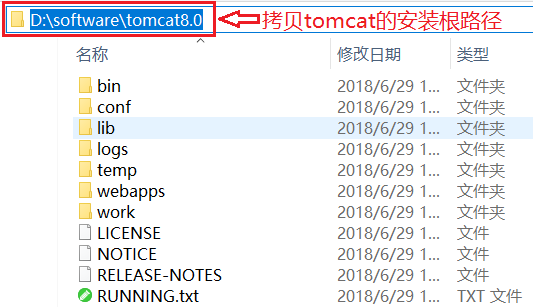


选择后点击next，进入下一步。

3、配置tomcat服务器的安装根路径：

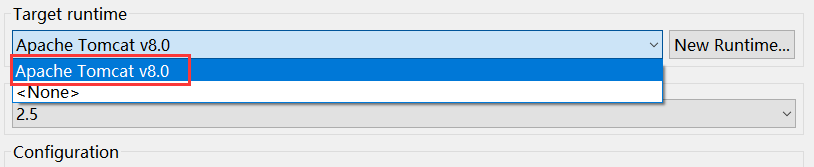


例如：找到tomcat的安装根目录，并拷贝路径，将路径粘贴到上面的输入框中。



配置完后点击完成即可

4、上一步完成后，回到Web项目创建的视图窗口，再查看"Target runtime"选项，如下：



### 修改Servlet模版

通过Eclipse可以直接创建一个Servlet类，这相比通过记事本等文本编辑工具创建Servlet，可以节省配置Servlet的时间，提高了我们的开发效率。

但是通过Eclipse生成的Servlet类中包含了许多我们不需要的注释和默认实现代码，这些每次都删除也非常占用时间。

接下来可以通过添加模版代码的形式，来生成Servlet的内容，以便于提高我们的开发效率。

**1、先创建一个Servlet，将其中的内容修改为自己期望的模版格式，并复制其中的内容，例如：**

package com.tedu;

import java.io.IOException;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

/\*\*

\*

\*/

public class TestServlet extends HttpServlet {

private static final long serialVersionUID = 1L;

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

//>>处理POST请求参数乱码

request.setCharacterEncoding("utf-8");

//>>处理响应正文乱码

response.setContentType("text/html;charset=utf-8");

//TODO...

}

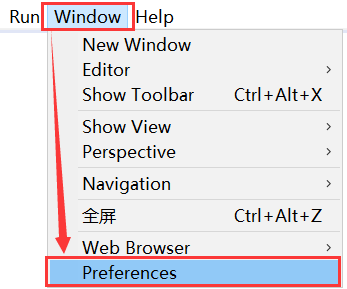
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

doGet(request, response);

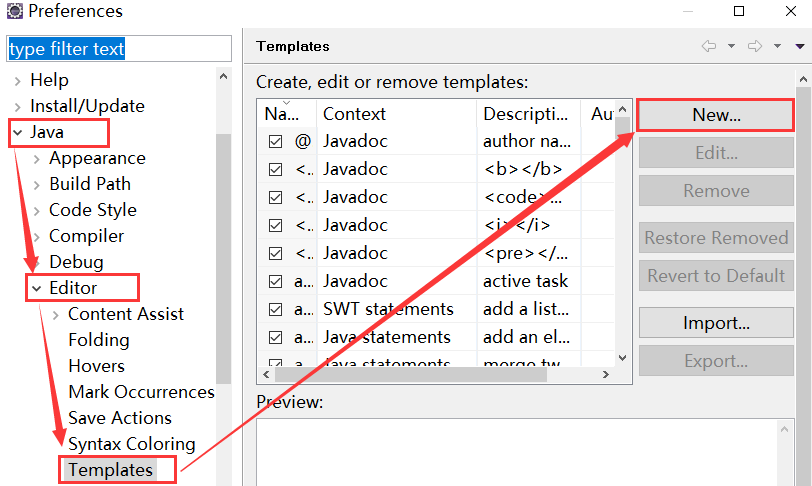
}

}

**2、点击菜单栏中的window --> Preferences**：



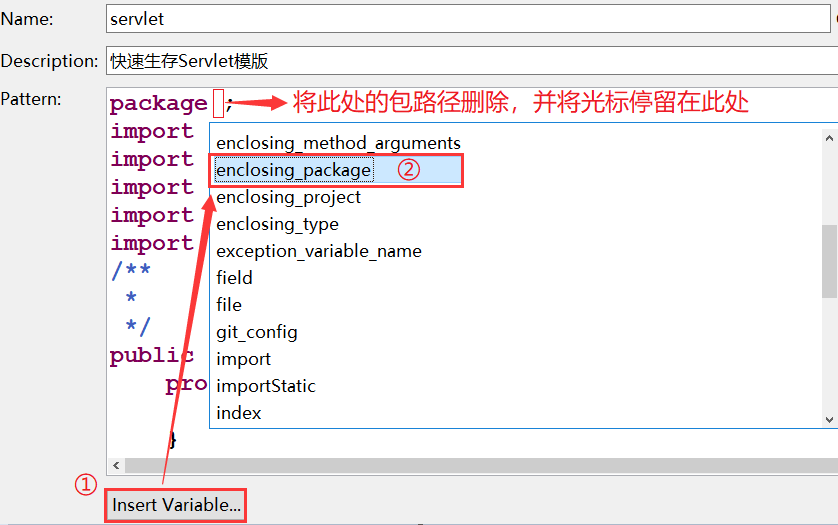
**3、在出现的窗口左侧依次点击：Java --> Editor --> templates -->(在右边的窗口中) 点击New… ：**



**4、在出现的新窗口中填写如下内容：**

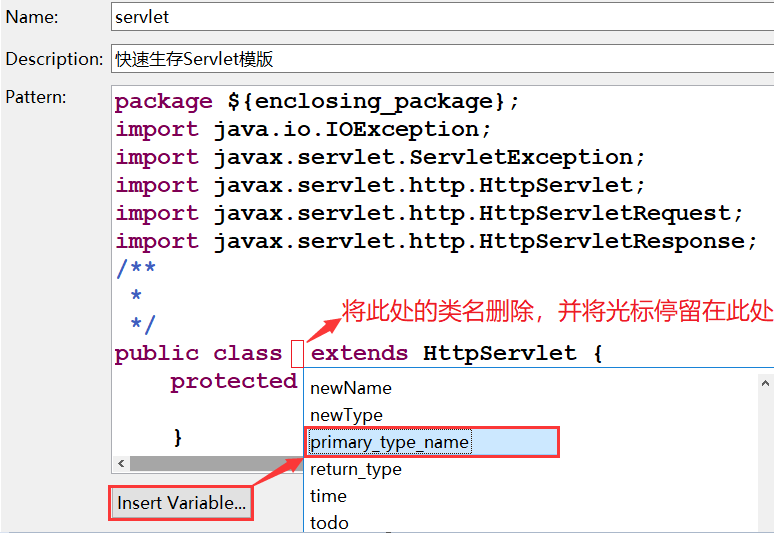


**5、替换包路径和类名（作用是在新建Servlet生成的Servlet模版中使用当前类的包路径和类型）**



效果如下：



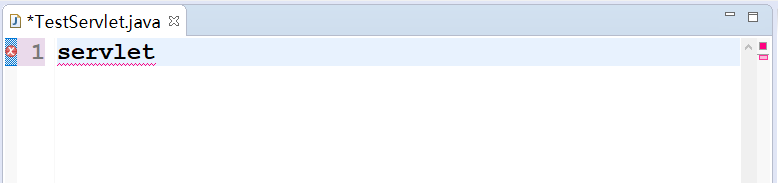


效果如下：

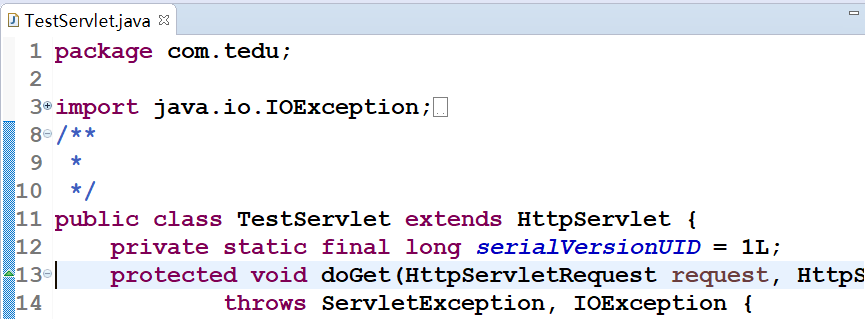


**6、点击OK保存，创建新的Servlet文件，测试**：

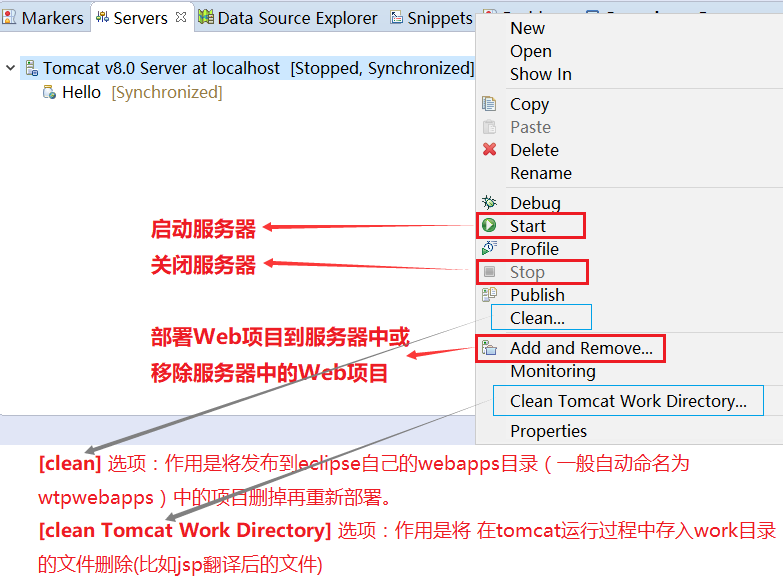
将Servlet中的所有内容全选删除，并输入"servlet"，接着按 "Alt+ /" 提示即可生成自己想要的Servlet模版内容!



效果如下:



### tomcat右键选项介绍



a) Start：用于启动tomcat服务器，如果已启动，则显示 ReStart，作用是重启服务器

b) Stop：用于停止服务器

c) Add and Remove：将Web应用部署到tomcat服务器中，或者移除服务器中部署的Web应用

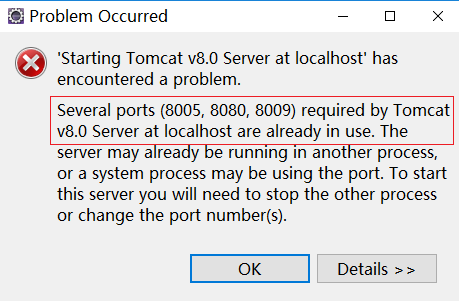
d) Clean：作用是将发布到Eclipse自己的webapps目录中的项目删除再重新部署

e) Clean Tomcat Work Directory：作用是将 在tomcat运行过程中存入work目录的文件删除

### FAQ常见问题

**问题1：tomcat服务器启动失败**

如果在启动服务器时，服务器启动失败，并弹出窗口显示如下异常信息：

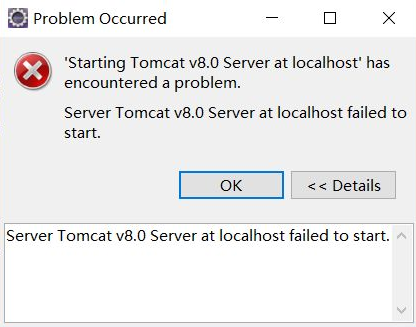


根据上面的描述信息，可以看出是8005、8080、8009端口被同时占用了，此时只有一种可能，就是之前已经启动了tomcat或者之前开启的tomcat没有完全关闭导致的。

解决方式：到tomcat安装目录找到bin目录中的shutdown.bat文件，双击运行将服务器关闭，再到Eclipse中启动服务器即可！

**问题2：tomcat服务器启动失败-2**

如果在启动服务器时，服务器启动失败，并弹出窗口显示如下异常信息：



**解决方法：**

(1) 可以先将服务器中所有的Web应用移除（服务器右键**Add and Remove**-->**Remove All**-->**Finish**）

(2) 再分别执行服务器右键选项中的**clean**和**Clean Tomcat Work Directory**

(3) 再次启动服务器！，如果还是启动失败，可以将tomcat服务器重新配置一次到Eclipse中(将服务器和Server项目删除, 再点击链接重新创建服务器)

## request和response介绍

request是代表Http请求信息的对象，其中封装了浏览器向服务器发送的请求信息，将request传递给service方法，在处理请求的过程中，可以通过request对象获取请求相关的信息。

response对象是代表Http响应信息的对象，其中将会封装服务器向浏览器做出的响应信息，将response传递给service方法，在处理请求的过程中，如果有任何数据要发送给浏览器，可以通过response获取流进行发送。

response.getWriter();

response.getOutputStream();

## request对象

### 获取请求参数

**问题1、什么是请求参数？**

通过浏览器向服务器发送请求时，在请求中可以携带一些数据，这些数据就称之为请求参数，例如：

http://localhost/day10/RequestDemo1?user=张三**&**like=篮球**&**like=足球&age=18

**问题2、如何获取请求参数？**

request.getParameter(String paramName);

-- 通过请求参数的名字，获取对应的参数值。返回值是一个字符串。

request.getParameterValues(String paramName);

-- 通过请求参数的名字，获取对应的所有参数值组成的数组。

因此，以后获取单值的参数时，推荐使用getParameter方法。

如果获取的多值的参数（即一个参数名对应多个参数值），推荐使用getParameterValues方法

示例:

//获取请求参数(user和like)

String user = request.getParameter("user");

System.***out***.println("user="+user);

String like = request.getParameter("like");

System.***out***.println("like="+like);

String[] likes = request.getParameterValues("like");

System.***out***.println( "likes="+ Arrays.*toString*( likes ) );

**问题3、获取请求参数时的中文乱码问题？**

(1)如果是GET提交，并且tomcat服务器是8.0及以后的版本，GET提交的中文参数是没有乱码问题的。（因为tomcat自8.0版本以后，已经处理了GET提交的中文参数乱码问题！）

(2)如果是GET提交，并且tomcat服务器是7.0及以前的版本，GET提交的中文参数也会有乱码问题。如何处理？？

在服务器的server.xml文件中的修改端口的Connector标签上添加一个属性, URIEncoding="utf-8"，如下：

<Connector connectionTimeout=*"20000"*

port=*"80"* protocol=*"HTTP/1.1"*

redirectPort=*"8443"*

URIEncoding=*"utf-8"*/>

这样，在tomcat7.0及以前的版本中，GET提交也没有乱码问题！

(3)如果是POST提交，无论是哪个版本的tomcat服务器，POST提交的中文参数都会有乱码。处理方式如下：

//在任何获取请求参数的代码之前，加上一行代码：

request.setCharacterEncoding("utf-8");

//获取请求参数...

...

### 实现请求转发

请求转发是服务器内部资源的一种跳转方式，即当浏览器发送请求访问服务器中的某一个资源时，该资源将请求转交给另外一个资源进行处理的过程，就叫做请求转发。

请求转发和重定向都是资源的跳转方式，但是跳转的过程有所不同。

**请求转发的特点：**

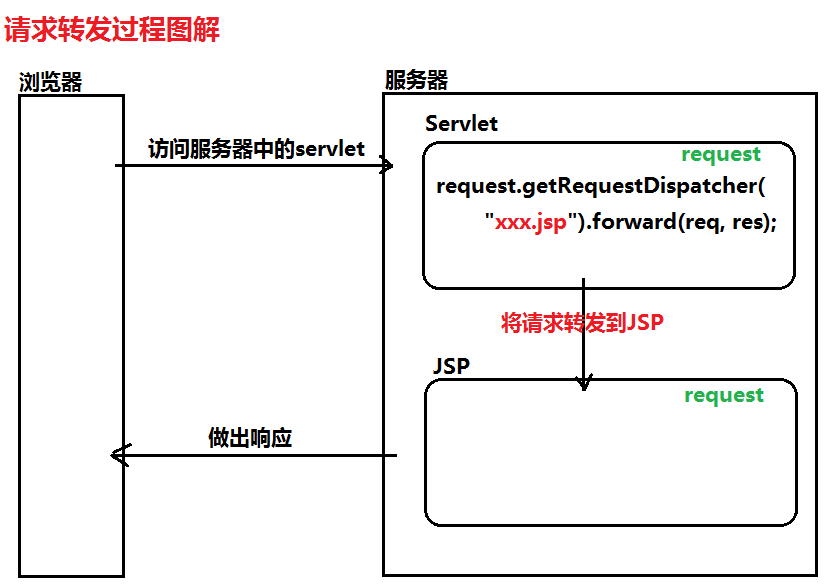
(1)请求转发前后是一次请求、一次响应

(2)请求转发前后，浏览器地址栏地址不会发生变化（转发是服务器内部的跳转，浏览器看不到）

(3)请求转发前后的request对象是同一个（因为请求只有一次，服务器根据一次请求，只会创建一个request对象，转发前后都需要request，所以只能在转发时，将request对象传递给转发后的那个资源）。

(4)请求转发只能在同一个web应用内部中的两个资源之间进行转发。

即转发前后的两个资源必须属于同一个web应用。



请求转发实现:

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

### 作为域对象使用

request在实现转发时，通过request.setAttribute方法和request.getAttribute方法带数据到目的地时，就是通过request对象中的map集合带数据，这个request对象上的map集合以及request对象所在的范围即称之为是一个域对象。

如果一个对象具备可以被访问的范围，通过这个对象上的map集合可以在整个范围内实现数据的共享。这样的对象就叫做域对象。

在request对象上提供了往域对象中存数据的方法以及取数据的方法:

request.setAttribute(String attrName, Object attrValue);

-- 将属性存入request域中(request对象的map集合里)，其中属性名只能是字符串，属性值可以是任意类型。

request.getAttribute(String attrName)

-- 根据属性名从request域中获取对应的属性值。返回值是一个Object。

**request域对象所具备的三大特征:**

**生命周期：**一次请求开始时，会创建代表请求的request对象，在一次请求结束时（响应已完成），会销毁request对象。

**作用范围：**一次请求范围内(由于请求转发前后是一次请求，所以在请求转发前后可以通过request域带数据到目的地；如果是重定向，由于重定向前后是两次请求，request对象也不是同一个，因此在重定向前后不能通过request域带数据。)

**主要功能：**带数据到目的地。

### 案例:模拟查询所有门店功能

**1、创建一个Servlet程序，用于处理查询所有门店信息请求**

**public** **class** DoorListServlet **extends** HttpServlet {

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

**protected** **void** doGet(HttpServletRequest request, HttpServletResponse response)

**throws** ServletException, IOException {

response.setContentType("text/html;charset=utf-8");

PrintWriter out = response.getWriter();

//1.模拟查询数据库, 查询所有门店集合

List<String> doorList = **new** ArrayList();

doorList.add("01, 永和北三环西路店, 010-67676767");

doorList.add("02, 永和西直门店, 010-68976347");

doorList.add("03, 永和东直门店, 010-78397647");

doorList.add("04, 永和北京西店, 010-78764397");

doorList.add("05, 永和天安门店, 010-78769743");

//2.将数据存入request域中

request.setAttribute( "list", doorList );

//3.将请求转发到 door\_list.jsp 中, 取出所有门店显示在页面中

request.getRequestDispatcher("door\_list.jsp")

.forward(request, response);

}

}

**2、创建一个JSP，用于显示所有门店信息**

<body>

<h3>显示所有门店信息</h3>

<%

//获取request域中的门店信息集合

List<String> list = (List<String>)request.getAttribute("list");

//遍历门店集合, 将门店信息输出在网页上

**for**( String door : list ){

out.write( door +"<br/>");

}

%>

</body>

## response对象

### 向客户端发送数据

PrintWriter response.**getWriter**() -- 字符流

ServletOuptputStream response.**getOutputStream**() -- 字节流

PrintWriter out = response.getWriter();

out.write("hello..");

通过response获取字符流向浏览器发送中文数据时，会出现乱码问题。

出现乱码的原因为：服务器默认用iso8859-1编码向浏览器发送数据，而这个码表中没有中文字符，所以在发送中文数据时，必然会出现乱码。

解决方法：

/\* 通知服务器使用utf-8码表向浏览器响应数据

\* 通知浏览器也使用utf-8码表接收服务器响应的数据 \*/

response.setContentType("text/html;charset=utf-8");

PrintWriter out = response.getWriter();

out.write("你好");

### 实现重定向

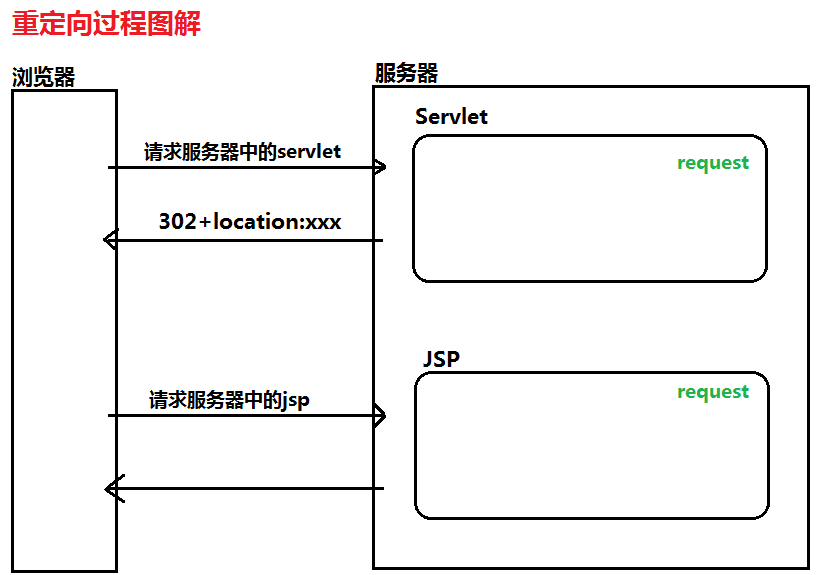
**重定向的特点：**

(1)重定向前后是两次请求，两次响应

(2)重定向前后，浏览器的地址栏地址会发生变化。（因为两次请求都是通过浏览器发起，浏览器知道这个跳转的过程，因此地址栏地址会变化）

(3)重定向前后的request对象不是同一个（因为重定向前后是两次请求，服务器根据两次请求会创建两个不同的request对象及response对象）

(4)重定向前后的两个资源可以是来自不同的web应用，甚至可以是来自不同的服务器。(进行跳转的两个资源之间没有限制)



实现代码：

response.sendRedirect("重定向到资源的地址");

//从当前Servlet重定向当前应用下的test.jsp

response.sendRedirect("test.jsp");

response.sendRedirect("http://localhost/day10/test.jsp");

response.sendRedirect("/day10/test.jsp");

//从当前Servlet重定向day09应用下的test.jsp

response.sendRedirect("http://localhost/day09/test.jsp");

response.sendRedirect("/day09/test.jsp");

//从当前Servlet重定向tmooc.cn的首页

response.sendRedirect("http://www.tmooc.cn");

# Day11 JSP、EL、JSTL

## JSP概述

### 什么是JSP

JSP是由SUN公司提供的一门动态Web资源开发技术

JSP看起来像一个HTML，但和html不同的是，JSP中可以书写Java代码，可以通过Java代码获取动态的数据。所以JSP可以开发动态网页。

JSP其实本质上就是一个Servlet程序!

**思考1：为什么要学习JSP?**

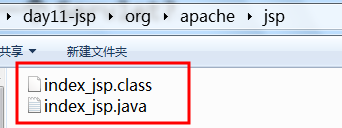
Servlet本质上是一段Java程序，擅长处理业务逻辑，但是不适合向外输出一个网页。

HTML可以用于开发网页，但仅限于开发静态网页，无法显示动态数据，JSP中可以书写Java代码，可以开发一个动态的网页。

JSP的出现既解决了Servlet不适合输出网页的问题，同时也解决html无法显示动态数据的问题！

**思考2：为什么说JSP本质是一个Servlet?**

JSP在第一次被访问时，会翻译成一个Servlet程序。



在访问JSP后看到的网页，其实就是翻译后的Servlet程序输出的网页！

## JSP执行过程

**访问JSP的过程为**: 访问**index.jsp** **--翻译->** **index\_jsp.java**(Servlet) **--编译->** **index\_jsp.class**(运行) **--结果->** **输出一个网页到浏览器**

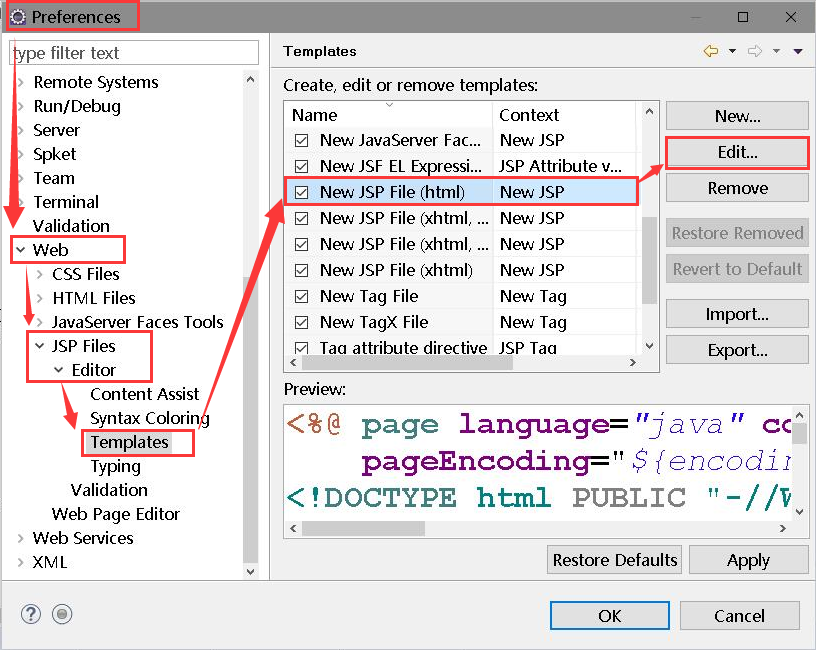
**如果是访问一个HTML**: 访问index.html ---> 服务器根据访问的路径去寻找index.html文件 ---> 如果能找到就直接将index.html文件的内容响应给浏览器，让浏览器解析。---> 如果找不到就响应一个404页面。

localhost/day11/index.html

总结：由于访问JSP需要在服务器端翻译、编译、运行，最终响应给浏览器一个html网页，而访问html会直接将html发送给浏览器解析，因此html运行效率比jsp高！

## 修改JSP模版

修改JSP模版步骤: 点击菜单栏中的 window --> Preferences，出现如下窗口:



点击edit编辑JSP模版，修改为如下:

<%@ page language="java" pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8"/>

<title></title>

</head>

<body>

${cursor}

</body>

</html>

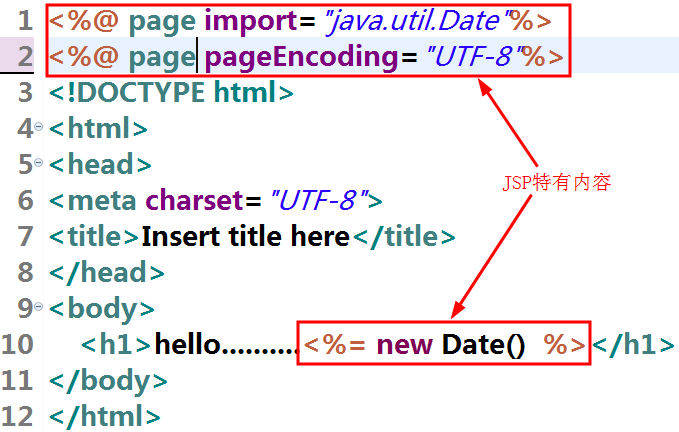
## JSP语法

### 模版元素

除了JSP特有内容以外的其他内容都属于模板元素

例如:下图中除了红色框中的内容，其他的都是模板元素。

模板元素在翻译后的servlet中，是直接被out.write()原样发送给浏览器，由浏览器负责解析并显示。



### JSP表达式

格式: **<%= 常量/变量/表达式 %>**

作用: 在翻译后的Servlet中，计算表达式的值，将计算的结果输出到浏览器，由浏览器负责解析并显示！

在JSP中：

<%= "Hello JSP" %>

Hello JSP

<%= 123+234 %>

<!-- 声明一个变量 -->

<% String name = "韩少云"; %>

<%= name %>

在翻译后的Servlet中：

out.print( "Hello JSP" );

out.write("\r\n");

out.write("\tHello JSP\r\n");

out.write("\t");

out.print( 123+234 );

out.write("\r\n");

out.write("\t<!-- 声明一个变量 -->\r\n");

out.write("\t");

String name = "韩少云";

out.write('\r');

out.write('\n');

out.write(' ');

out.print( name );

### JSP脚本片段

格式：<% 若干Java语句 %>

作用：在翻译后的Servlet中，将脚本片段中的java语句复制粘贴到对应的位置执行！例如:

在JSP中:

<!-- 循环输出5行"hello jsp..." -->

<%

**for**(**int** i=0;i<5;i++){

out.write("hello jsp...<br/>");

}

%>

在翻译后的Servlet中:

out.write("\t<!-- 循环输出5行\"hello jsp...\" -->\r\n");

out.write("\t");

for(int i=0;i<5;i++){

out.write("hello jsp...<br/>");

}

其他形式，在JSP中：

<% **for**(**int** i=0;i<5;i++){ %>

hello jsp~~~<br/>

<% } %>

在翻译后的Servlet中：

for(int i=0;i<5;i++){

out.write("\r\n");

out.write("\t\thello jsp~~~<br/>\r\n");

out.write("\t");

}

写在某一个脚本片段中的Java代码可以是不完整的，但是最终的所有脚本片段中的内容加在一起，必须是完整符合Java语法的！

### JSP注释

格式: <%-- 注释内容 --%>

作用: (1)为代码添加解释说明

(2)将一些暂时不需要执行的代码注释掉。

关于JSP注释，在翻译时，直接将JSP注释丢弃，不参与翻译过程。

在JSP中可以书写JSP注释、java注释、html注释：

**<%--**

<% out.write("aaa<br/>"); %>

**--%>**

上面的JSP注释包括其中的注释内容,在翻译时直接被抛弃,不参与翻译。

所以其中的aaa不会发送给浏览器，也不会显示在页面上。

<% **//**out.write("bbb<br/>"); %>

上面的java注释内容放在脚本片段中，会参与翻译，但是由于被java注释注释了，所以也不会执行，bbb也不会发送给浏览器，不会显示在页面上。

**<!--**

<% out.write("ccc<br/>"); %>

**-->**

上面的html注释会被当做模板元素被out.write发送到浏览器, 其中包裹的脚本片段中的内容也会执行。ccc也会发送到浏览器。但由于前后被html注释包裹了。浏览器不会解析html注释中的内容，因此也不会显示。

### JSP指令

指令的格式: <%@ 指令名称 若干属性声明... %>

指令的作用: 用于指挥JSP解析引擎如何将一个JSP翻译成一个Servlet程序。

1、page指令

page指令的作用是,用于声明JSP文件的基本属性信息(比如,jsp使用的开发语言,jsp使用的编码,jsp中需要引入哪些包等.)

<%@ page language=*"java"*%>

-- language属性用于指定jsp使用的开发语言, 指定为java。默认也是java.

<%@ page import=*"java.util.Date"*%>

import是用于导入包用的, 类似于java程序中的import关键字

**import** java.io.IOException;

**import** java.io.PrintWriter;

**import** javax.servlet.ServletException;

**import** javax.servlet.http.HttpServlet;

<%@ page pageEncoding=*"UTF-8"*%>

pageEncoding属性是用于指定JSP使用的编码。

只要保证pageEncoding属性指定的编码和JSP文件另存时的编码一致。

JSP中就不会有乱码问题。（在JSP中获取请求参数乱码另说）

可以将上面的属性都声明在一个page指令上。

<%@ page language=*"java"* pageEncoding=*"UTF-8"* import=*"java.util.Date"*%>

2、taglib指令

taglib指令是用于引入JSTL标签库或者其他标签库时用的

后面在讲解JSTL标签库时,会讲解其用法。

## JSP标签技术

在JSP页面中写入大量的java代码会导致JSP页面中html代码和java代码混杂在一起，会造成页面非常的混乱，难于维护。

于是在JSP的2.0版本中，sun提出了JSP标签技术，推荐使用标签来代替JSP页面中java代码，并且推荐，JSP2.0以后不要在JSP页面中出现任何一行java代码。

### EL表达式

格式: ${ 常量、表达式、变量 }

注意：放在EL中的变量必须得先存入域才可以获取到。

作用：(1)计算表达式的值，将计算的结果输出

(2)主要作用是用于从作用域中获取数据。

**1、获取常量、表达式、变量的值（变量得先存入域中）**

${ "Hello EL" } Hello EL

${ 100+123 }

<hr/>

<% //声明一个name变量,并存入request域中

String name = "林青霞";

request.setAttribute( "n", name );

%>

${ n }

<%--

在EL中书写变量, 底层会根据变量名(比如name)到四大作用域中

寻找该名称的属性值, 如果找到就直接返回。如果找不到，就什么

也不输出！寻找时的顺序是按照域的大小, 按照从小到大去寻找:

pageContext < request < session < application

--%>

**2、获取作用域中数组或集合中的元素**

${ ns[0] } <br/>

${ ns[1] } <br/>

${ ns[2] } <br/>

${ ns[3] } <br/>

**3、获取作用域中map集合中的元素**

${ map1.name } <br/>

${ map1.age } <br/>

${ map1.nickname } <br/>

**4、获取作用域中JavaBean对象的属性值**

${ user.getUsername() } <br/>

${ user.getPassword() } <br/>

${ user.getAddr() } <br/>

<hr/>

${ user.username } <br/><!-- 底层还是调用getUsername方法 -->

${ user.password } <br/><!-- 底层还是调用getPassword方法 -->

${ user.addr } <br/><!-- 底层还是调用getAddr方法 -->

### JSTL标签库

JSTL标签库是为JavaWeb开发人员提供的一套标准通用的标签库；

JSTL标签库和EL配合使用可以取代JSP中大部分的Java代码；

在使用JSTL标签库之前需要完成:

(1)导入JSTL的开发包

(2)在使用JSTL标签库的JSP中引入JSTL

其中常用的标签如下:

**1、<c:set></c:set>**

-- 往四大作用域中添加域属性，或者修改四大作用域中已有的属性

(1) 往四大作用域中添加一个域属性

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

${ name }

(2) 修改四大作用域中已有的属性

如果重复添加相同的属性，值会发生覆盖，相当于修改.

<c:set var=*"name"* value=*"张三丰"* scope=*"request"*/>

${ name }

**c\_set标签属性总结:**

(1)var -- 指定存入作用域中的属性名称

(2)value -- 指定存入作用域中属性的值

(3)scope -- 指定将属性存入哪一个作用域中

可取值: a)page表示pageContext域 b)request表示request域 ­­

c)session表示session域 d)application表示ServletContext域

**2、<c:if></c:if>** -- 构造if…else…语句

<h3>往域中存入一个成绩, 根据成绩判断成绩所属的等级</h3>

<c:set var=*"score"* value=*"-58"*/>

<c:if test=*"*${ score<=100 && score > 80 }*"*>

优秀!

</c:if>

<c:if test=*"*${ score<=80 && score >= 60 }*"*>

中等!

</c:if>

<c:if test=*"*${ score < 60 && score >= 0 }*"*>

不及格!

</c:if>

<c:if test=*"*${ score < 0 || score > 100 }*"*>

您输入的成绩不合法!

</c:if>

test属性用于指定判断的条件，注意:JSTL中没有提供else对应的标签

**3、<c:forEach></c:forEach>**

-- 对集合或数组等中元素进行循环遍历或者是执行指定次数的循环.

(1) 遍历域中数组或集合中的元素

<%

//声明一个数组, 并将数组存入域中

String[] names = {"王海涛","齐雷","陈子枢"};

request.setAttribute("names", names);

%>

<c:forEach items=*"*${names}*"* var=*"name"*>

${ name }

</c:forEach>

(2) 遍历域中map集合中的元素

<%

//声明一个Map, 并将Map存入域中

Map map = **new** HashMap();

map.put("name", "关羽");

map.put("age", "38");

map.put("addr", "中鼎大厦B座");

request.setAttribute("map", map);

%>

<c:forEach items=*"*${ map }*"* var=*"entry"*>

${ entry.key } : ${ entry.value } <br/>

</c:forEach>

(3) 遍历0~100之间的整数，将是3的倍数的数值输出到浏览器中

<c:forEach begin=*"0"* end=*"100"* var=*"i"* >

${ i%3==0? i : "" }

</c:forEach>

**c\_forEach标签属性总结:**

(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

# Day12 Maven

今日学习目标：

(1)了解什么是Maven及Maven的作用

(2)掌握Maven安装及整合到Eclipse中

(3)掌握如何使用Maven构建Java项目和Web项目

(4)掌握使用Eclipse导入现有的Maven项目

(5)了解Maven的三种仓库(本地仓库、镜像仓库、中央仓库)

(6)了解Maven如何管理依赖（即管理jar包）

## Maven介绍

### Maven是什么?



**Maven**: 翻译为"专家"、"内行"，是**Apache**下的一个纯Java开发的一个开源项目。

**Maven**是一个项目管理工具，使用Maven可以来管理企业级的Java项目开发及依赖的管理。

使用**Maven**开发，可以简化项目配置，统一项目结构。总之，Maven可以让开发者的工作变得更简单。

*什么是依赖管理？要明白依赖管理，首先要知道什么是依赖？*

*一个Java项目中往往会依赖一些第三方的jar包。比如JDBC程序中要依赖数据库驱动包，或者在使用c3p0连接池时，要依赖c3p0的jar包等。 这时我们称这些Java项目****依赖****第三方jar包。*

*而所谓的依赖管理，其实就是对项目中所有依赖的jar包进行规范化管理。*

### 为什么要使用Maven?

传统的项目（工程）中管理项目所依赖的jar包完全靠人工进行管理，而人工管理jar包可能会产生诸多问题。

**1、不使用Maven，采用传统方式管理jar包的弊端：**

(1)在一些大型项目中会使用一些框架，比如SSM或者SSH框架，而框架中所包含的jar包非常多（甚至还依赖其他第三方的jar包），如果这些jar包我们手动去网上寻找，有些jar包不容易找到，比较麻烦。

(2)传统方式会将jar包添加到工程中，比如Java工程中将jar包放在工程根目录或者放在自建的lib目录下；JavaWeb工程会将jar包放在:/WEB-INF/lib目录下，这样会导致项目文件的体积暴增（例如，有些项目代码本身体积可能仅仅几兆，而加入jar包后，工程的体积可能会达到几十兆甚至百兆）。

(3)在传统的Java项目中是将所有的jar包统一拷贝的同一目录中，可能会存在jar包文件名称冲突的问题！

(4)在进行项目整合时，可能会出现jar包版本冲突的问题。

(5)在传统java项目中通过编译（手动编译或者在eclipse保存自动编译）、测试（手动在main函数中测试、junit单元测试）、打包部署(手动打war包/手动发布)、运行(手动启动tomcat运行)，最终访问程序。

**2、使用Maven来管理jar包的优势：**

(1)Maven团队维护了一个非常全的Maven仓库(中央仓库)，其中几乎包含了所有的jar包，使用Maven创建的工程可以自动到Maven仓库中下载jar包，方便且不易出错。

另外, 在Maven构建的项目中，如果要使用到一些框架，我们只需要引入框架的核心jar包，框架所依赖的其他第三方jar包，Maven也会一并去下载。

(2)在Maven构建的项目中，不会将项目所依赖的jar包拷贝到每一个项目中，而是将jar包统一放在仓库中管理，在项目中只需要引入jar包的位置(坐标)即可。这样实现了jar包的复用。

(3)Maven采用坐标来管理仓库中的jar包，其中的目录结构为【公司名称+项目/产品名称+版本号】，可以根据坐标定位到具体的jar包。即使使用不同公司中同名的jar包，坐标不同（目录结构不同），文件名也不会冲突。

(4)Maven构建的项目中，通过pom文件对项目中所依赖的jar包及版本进行统一管理，可避免版本冲突。

(5)在Maven项目中，通过一个命令或者一键就可以实现项目的编译（mvn complie）、测试（mvn test）、打包部署（mvn deploy）、运行（mvn install）等。

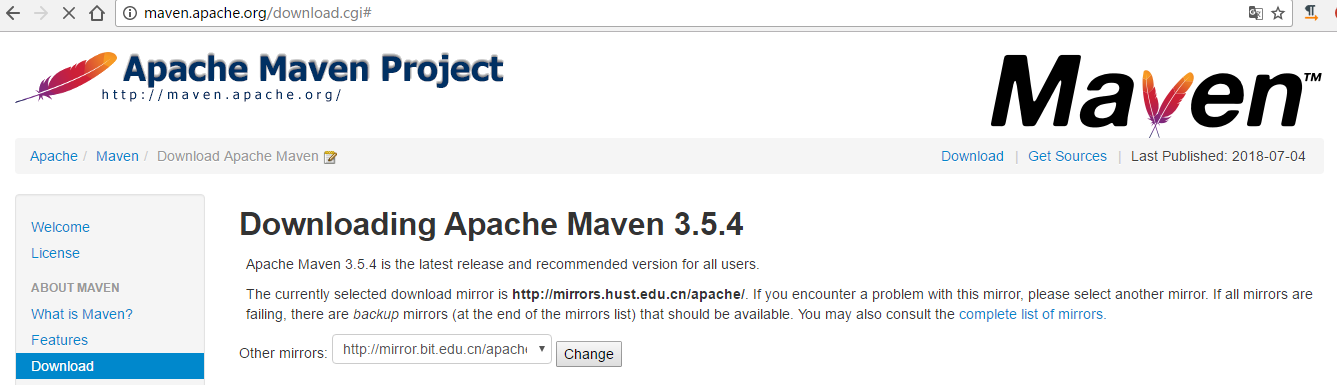
还有发布到tomcat服务器中运行: mvn tomcat7:run。如果想实现上面的所有过程，只需要记住一个命令：mvn install

总之，使用Maven遵循规范开发有利于提高大型团队的开发效率，降低项目的维护成本，大公司都会优先使用Maven来构建项目.

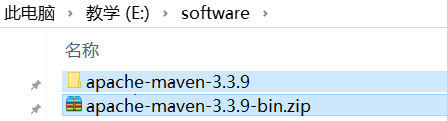
## Maven安装

### 下载、安装Maven

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



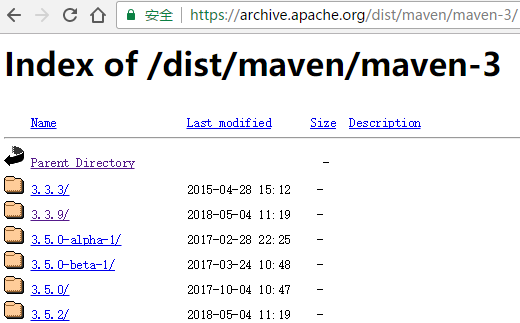
2、下载绿色版，解压之后就可以使用。





原则: 安装的路径中不要有中文和空格!!

3、若要下载旧版本Maven，可以访问：<https://archive.apache.org/dist/maven/maven-3/>



## Maven相关配置

在开发中更多是通过Eclipse+Maven来构建Maven项目，所以这里我们需要将Maven配置到Eclipse开发中。

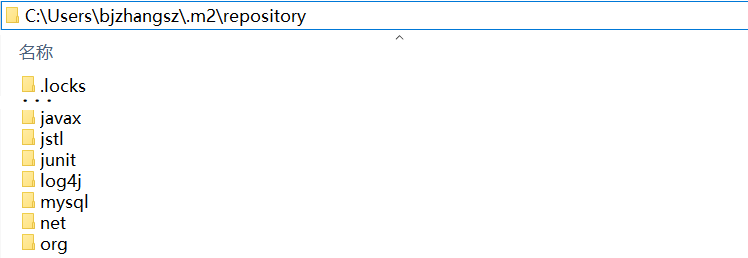
在将安装好的Maven工具配置的Eclipse开发中之前，需要做一些配置。

### 配置本地仓库位置

本地仓库：其实就是本地硬盘上的某一目录，该目录中会包含项目中所需要的所有jar包及插件。当所需jar包在本地仓库没有时，从网络上下载下来的jar包也会存放在本地仓库中。

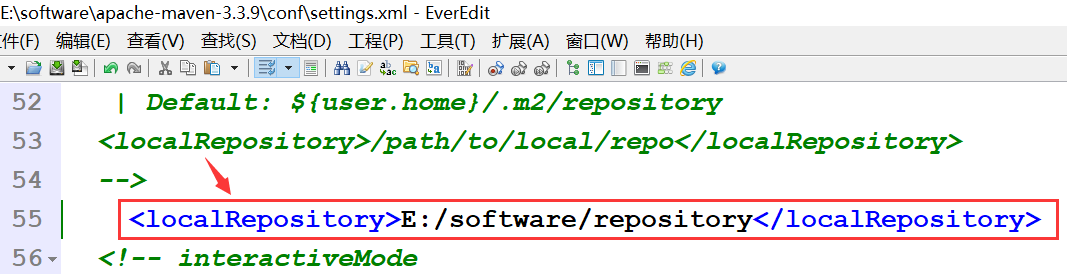
因此本地仓库其实就是一个jar包的仓库

maven指定的本地仓库的默认位置是在c盘，默认在：C:\Users\{当前用户}\.m2\repository，例如:



可以保持默认，当然也可以修改其位置到别的盘符路径。

修改方法：找到[MAVEN\_HOME]/conf/目录中的配置文件settings.xml，修改maven仓库的路径。



配置该目录后，以后通过maven下载的jar包将会保存在配置的目录下。

### 配置镜像服务器

当maven项目中需要依赖jar包时，只需要在项目的pom文件中添加jar对应的坐标，Maven就会到Maven的本地仓库中引用相应的jar包，如果本地仓库没有，就会到远程仓库去下载jar包。

如果不配置默认连接的是中央仓库，由于中央仓库面对的是全球用户，所以在下载jar包时，效率可能会比较低。

在settings文件中配置私服服务器为达内私服服务器（连接达内内网）或者阿里云服务器（连接外网）。

1、如果连接的是达内内网，添加达内镜像服务器配置信息，否则将无法下载jar包

需要做的是，在settings.xml文件中的<settings>标签下的<mirrors>标签内部添加如下配置：

配置达内镜像仓库：

<mirror>

<id>nexus-tedu</id>

<name>Nexus tedu</name>

<mirrorOf>central</mirrorOf>

<url>http://maven.tedu.cn/nexus/content/groups/public/</url>

</mirror>

2、如果在家里连接的外网，无法连接达内的私服服务器，则可以配置连接阿里云私服服务器，配置如下：

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>

镜像仓库配置总结：

(1)当所需jar包在本地仓库没有时，会从网络上下载。配置镜像仓库其实就是配置，去网络中哪个位置下载jar包到本地。

(2)如果在公司，并且公司有搭建镜像服务器，推荐使用公司的镜像服务器下载jar包，速度会更快。（如果在达内，使用的是内网，只能配置达内的镜像仓库，否则连接不了外网，jar包会下载失败！）

(3)如果在家里，使用的是外网，可以不配置镜像服务器，默认连接中央仓库下载jar包，或者配置阿里云的镜像服务器。（注意，如果配置阿里云镜像服务器，**不可使用手机热点网络！！**）

### 配置JDK版本

通过 Maven创建的工程，JDK版本默认是JDK1.5，每次都需要手动改为更高的版本。

这里可以通过修改maven的settings.xml文件, 达到一劳永逸的效果。

打开 %maven根目录%/conf/settings.xml 文件并编辑它：

在 settings.xml 的<profiles> 标签内部添加如下配置：

<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>

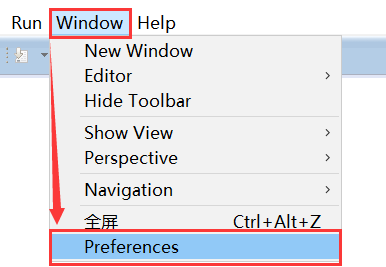
</properties>

</profile>

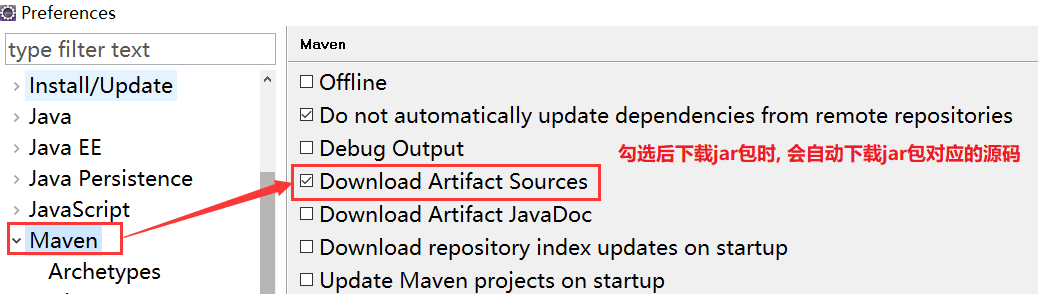
### 将Maven配置到Eclipse中

将Maven工具配置到Eclipse中，就可以通过Eclipse和自己安装的Maven创建Maven项目了。

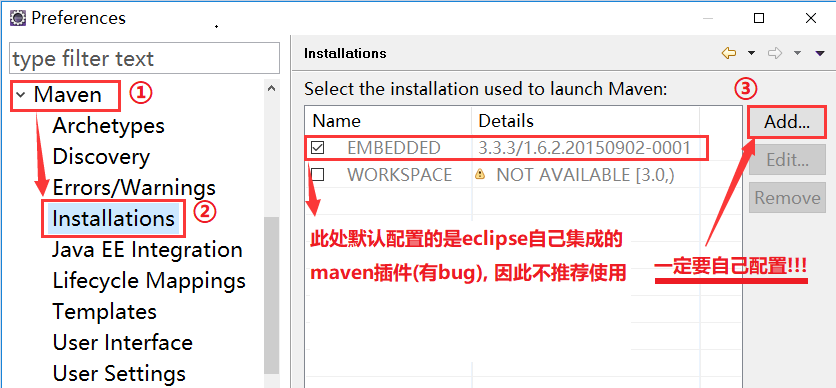
**1、window右键--> Preferences：**



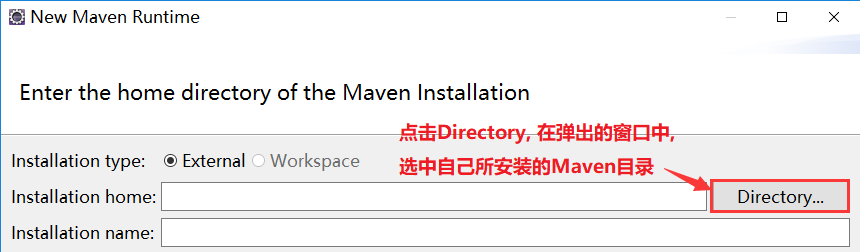
**2、点击Maven选项，在右侧选项中勾选 "Download Artifact Sources"：**

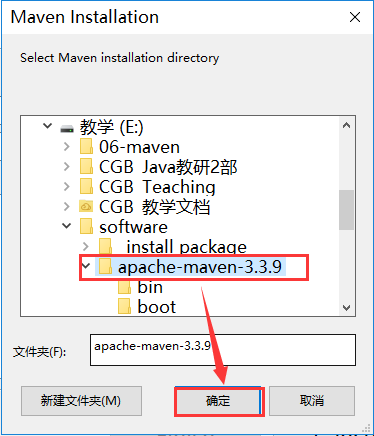


**3、点击add将自己安装的Maven添加进来：**

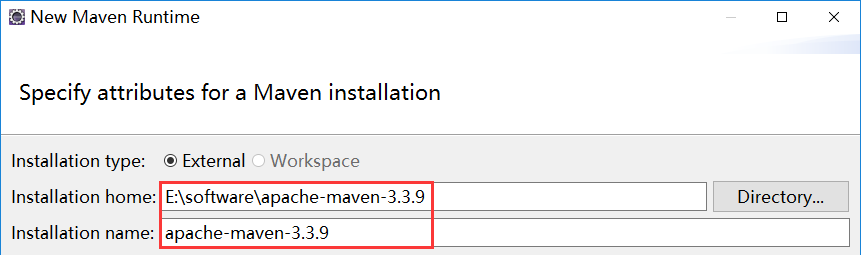


**4、添加自己安装的Maven：**

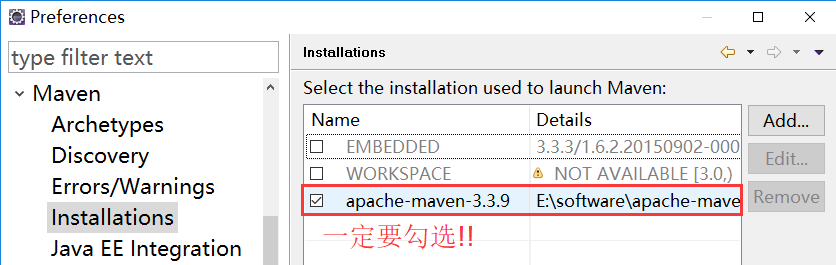




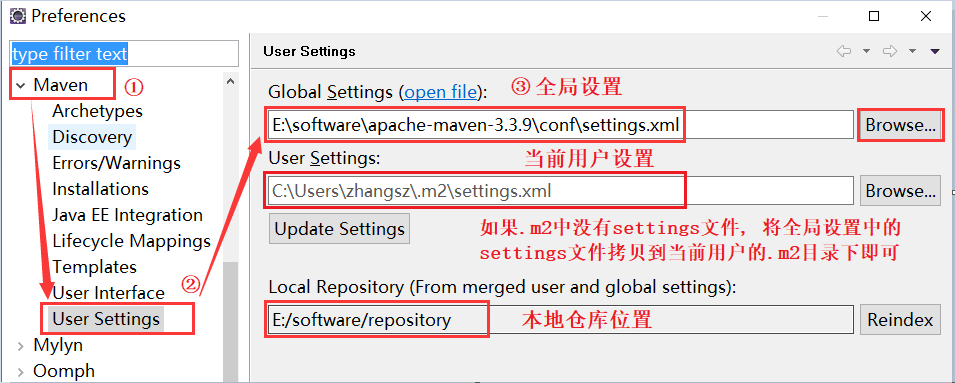
**一定要注意:** 自己安装的Maven不要放在桌面上(容易丢失，并且路径中往往有空格)，maven的安装路径中也不要包含中文和空格!!



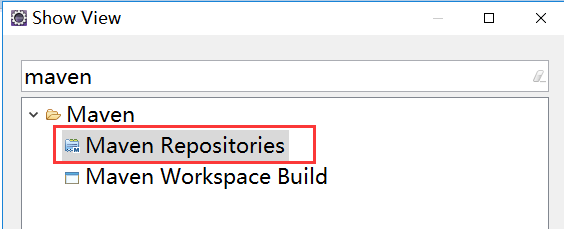
**5、将默认的maven切换为自己配置的maven：**



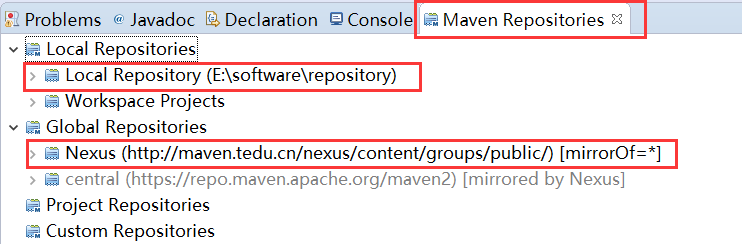
**6、设置maven的settings文件的位置：**



**7、测试是否配置成功：**window---> show view ---> other中搜索 "maven"，点击下面的选框中的选项

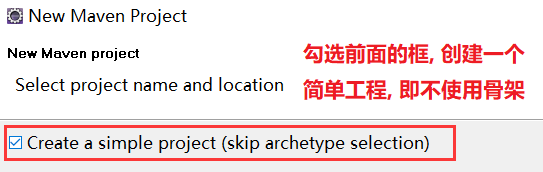


在弹出的窗口中，查看自己配置的本地仓库和远程仓库镜像:



## Maven项目构建

在利用Maven构建项目时分两种，第一种是：创建简单工程**（Create a simple project）**，即在创建时勾选前面的框。

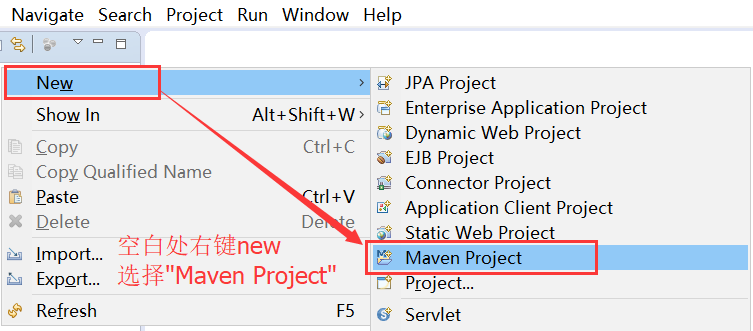


*(不勾选前面的框，即创建使用骨架（其实就是模版）创建Maven工程)*

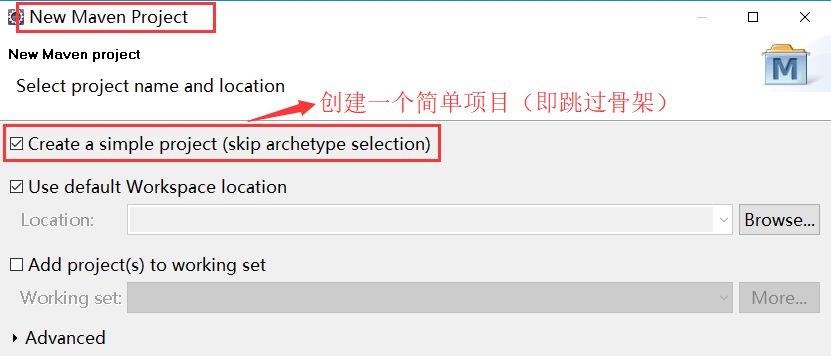
另，在创建简单工程时，还分为创建**Java工程**和**JavaWeb工程**。下面分别进行演示。

### 创建简单工程——Java工程

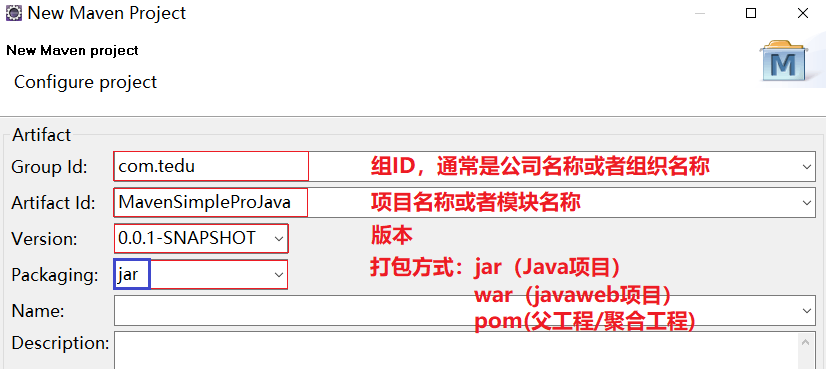
**1、空白处右键New ---> Maven Project：**



**2、在弹出的窗口中，勾选前面的框，创建一个简单工程（即不使用骨架），进入下一步。**



**3、在弹出的窗口中，填写内容(Package选择jar，即创建java工程)，点击完成即可。**



**在上述内容中，必填的内容有四项：**

(1)**Group Id** -- 组的名称，通常填写公司名称（比如 com.tedu）或者组织名称(org.apache..)

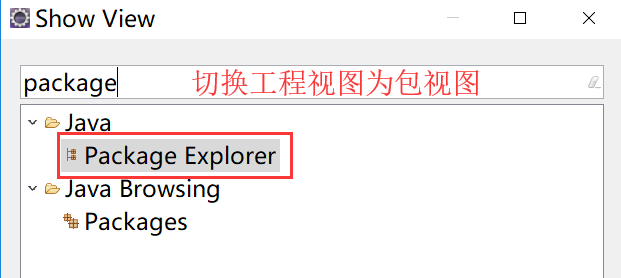
(2)**Artifact Id** -- 项目名称或者模块名称

(3)**Version** -- 项目的版本，创建的项目默认是0.0.1-SNAPSHOT 快照，也叫非正式版，正式版是RELEASE)

(4)**Package** -- 项目的类型: jar表示创建的是Java工程，war表示创建的是web工程，pom表示创建的是父工程（当然相对的还有子工程）或者聚合工程，pom目前我们不讨论。

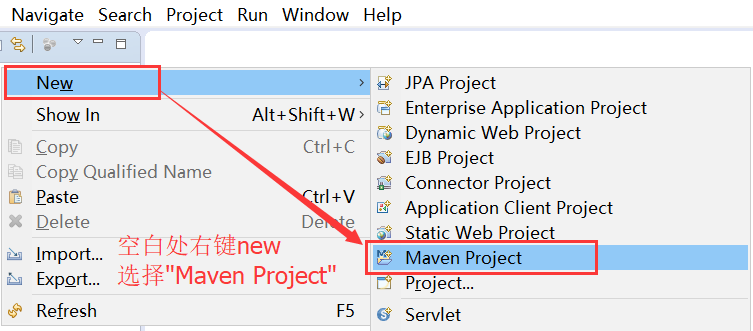
填写完毕后，点击完成即可完成创建简单Java工程

**4、切换工程视图为包视图：window --> show view，在弹出的窗口中搜索：**

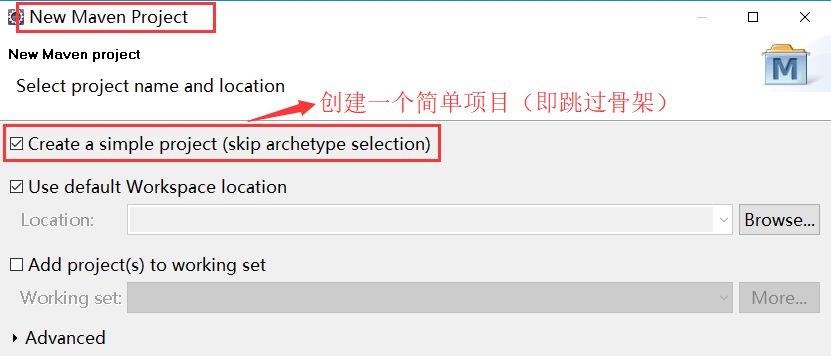


### 创建简单工程——JavaWeb工程

**1、空白处右键New ---> Maven Project：**



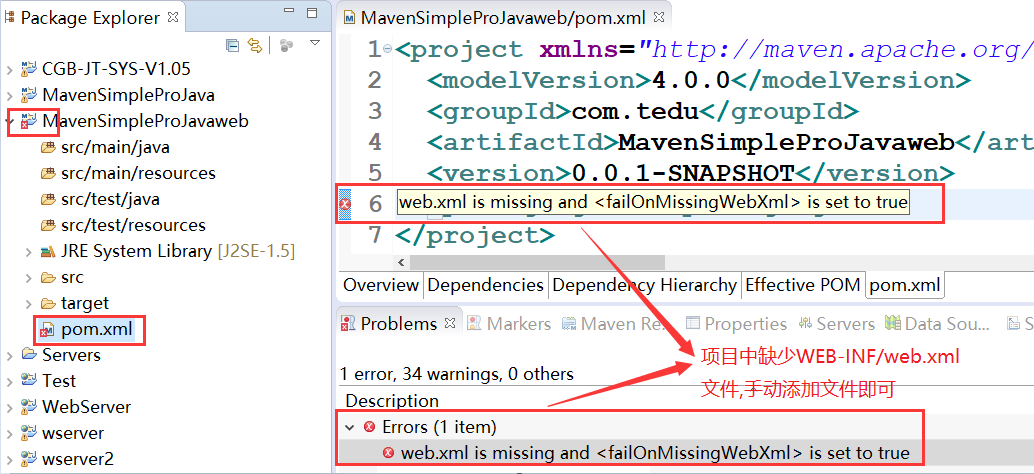
**2、在弹出的窗口中，勾选前面的框，创建一个简单工程（即不使用骨架），进入下一步。**



**3、在弹出的窗口中，填写内容(Package选择war，即创建web工程)，点击完成即可。**



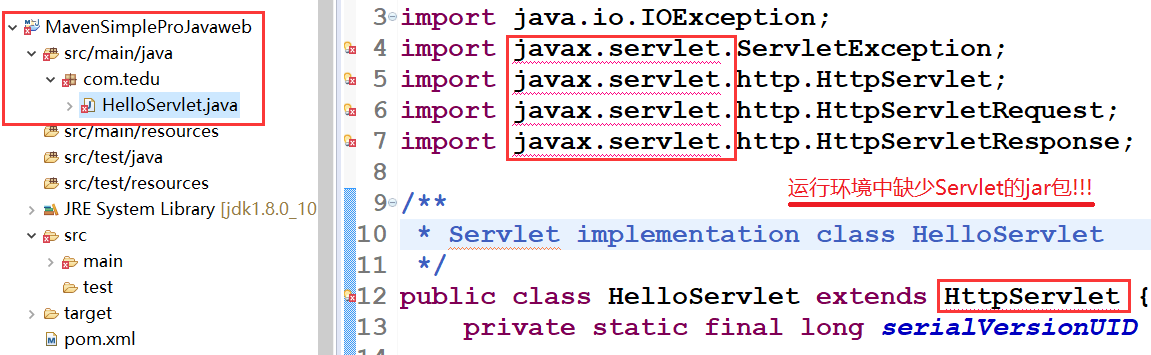
**4、创建完成后pom.xml文件会报错，说找不到web.xml文件，例如：**



**手动添加(拷贝)即可，例如:**



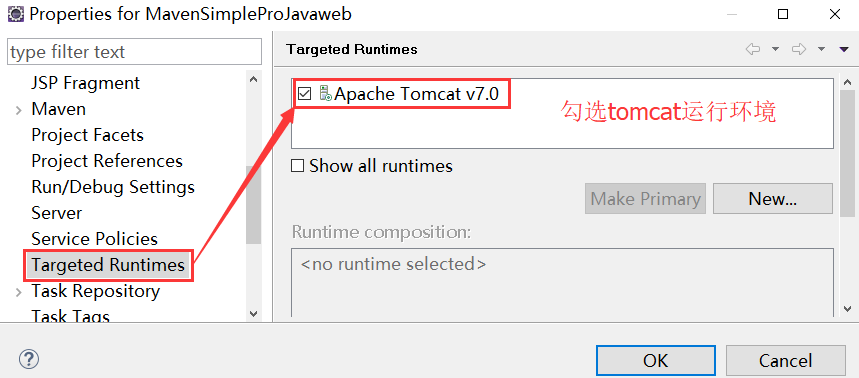
**5、创建Servlet程序，测试运行环境。**



上面的错误是因为运行环境中缺少Servlet的jar包，将tomcat运行环境添加过来即可！

**\*\*\* 缺少Servlet运行环境解决方案：**

**1、在项目上点击鼠标右键，选择 "Properties" ---> "Targeted Runtimes":**



**或者，**

**2、如果当前没有配置tomcat运行环境，在项目中的pom.xml文件中的根标签下添加Servlet的jar包的坐标，引入Servlet，如下：**

添加依赖：

<dependencies>

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>servlet-api</artifactId>

<version>2.5</version>

<scope>provided</scope>

</dependency>

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>jsp-api</artifactId>

<version>2.0</version>

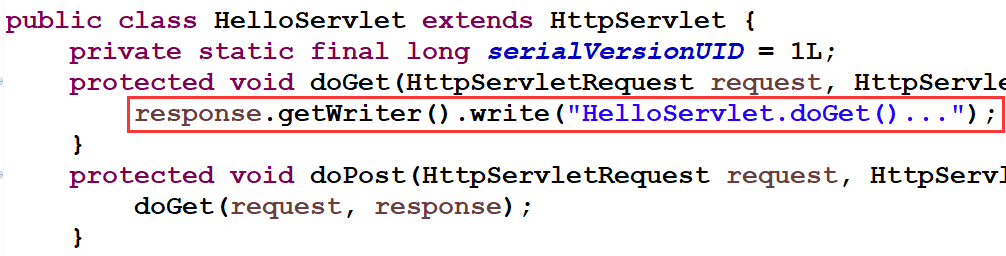
<scope>provided</scope>

</dependency>

</dependencies>

添加后保存pom文件。若还报错，在项目上右键选择 **"Maven"** ---> **"Update Project…"** 更新工程即可！

**6、实现Servlet程序**



**7、测试访问：**打开浏览器访问地址：[http://localhost/MavenSimpleProJavaweb/HelloServlet](http://localhost:8080/MavenSimpleProJavaweb/HelloServlet)



## 导入已有的Maven项目

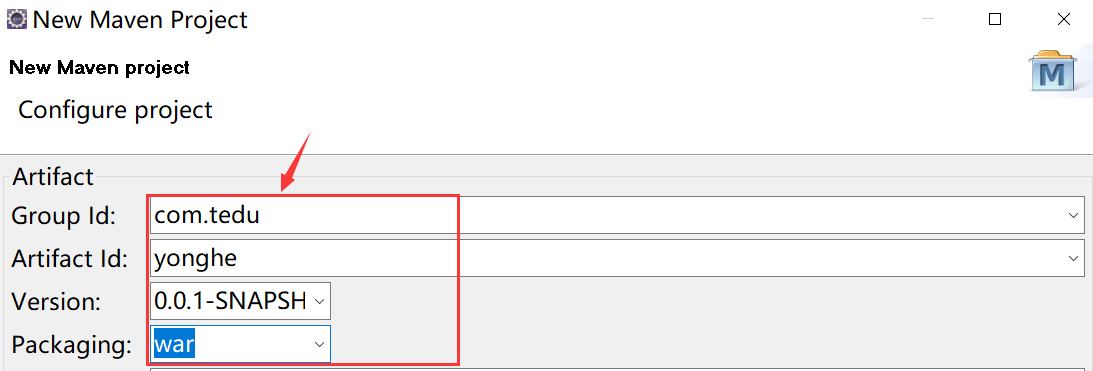
现将后面通过SSM框架实现的<<永和大王门店管理系统>>（Maven）项目导入到我们的Eclipse开发环境中。

在导入项目时我们通常会通过 "File" --> "Import..." 来导入项目，但是这样能会产生环境问题：如果项目本身自带的环境和我们当前使用的开发环境不一致，就会产生问题。

下面是导入的步骤：

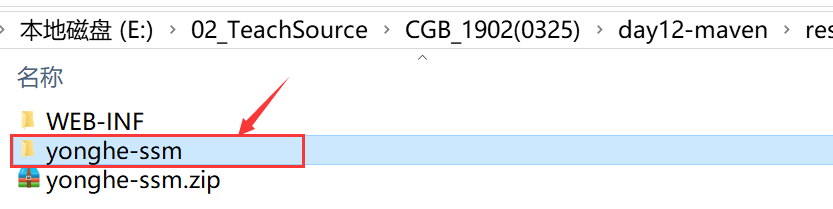
**1、创建一个新的Maven工程（JavaWeb工程）**

确保已经配置好Maven的环境后，在Eclipse中创建一个新的Maven工程（Javaweb工程），新工程的名字和所导入的工程的名字可以相同也可以不同，例如：

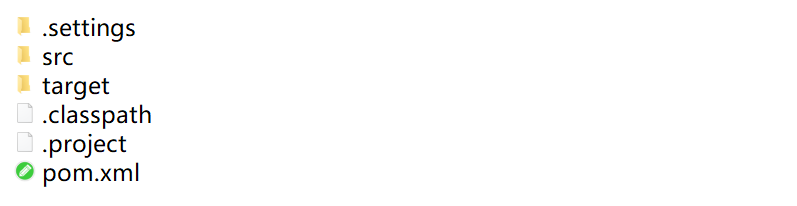


**2、解压CGB-JT-SYS-V1.05目录，复制src目录和pom文件**

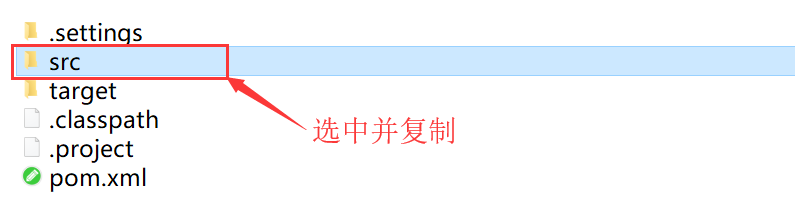
将下发的CGB-JT-SYS-V1.05项目解压出来：



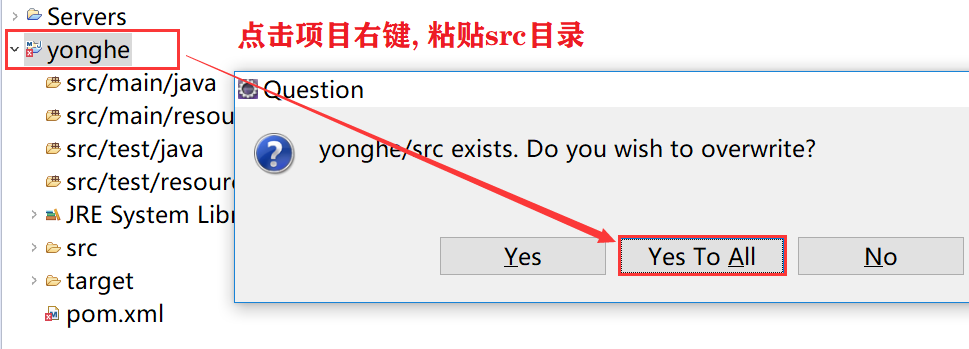
解压后的结构如下：



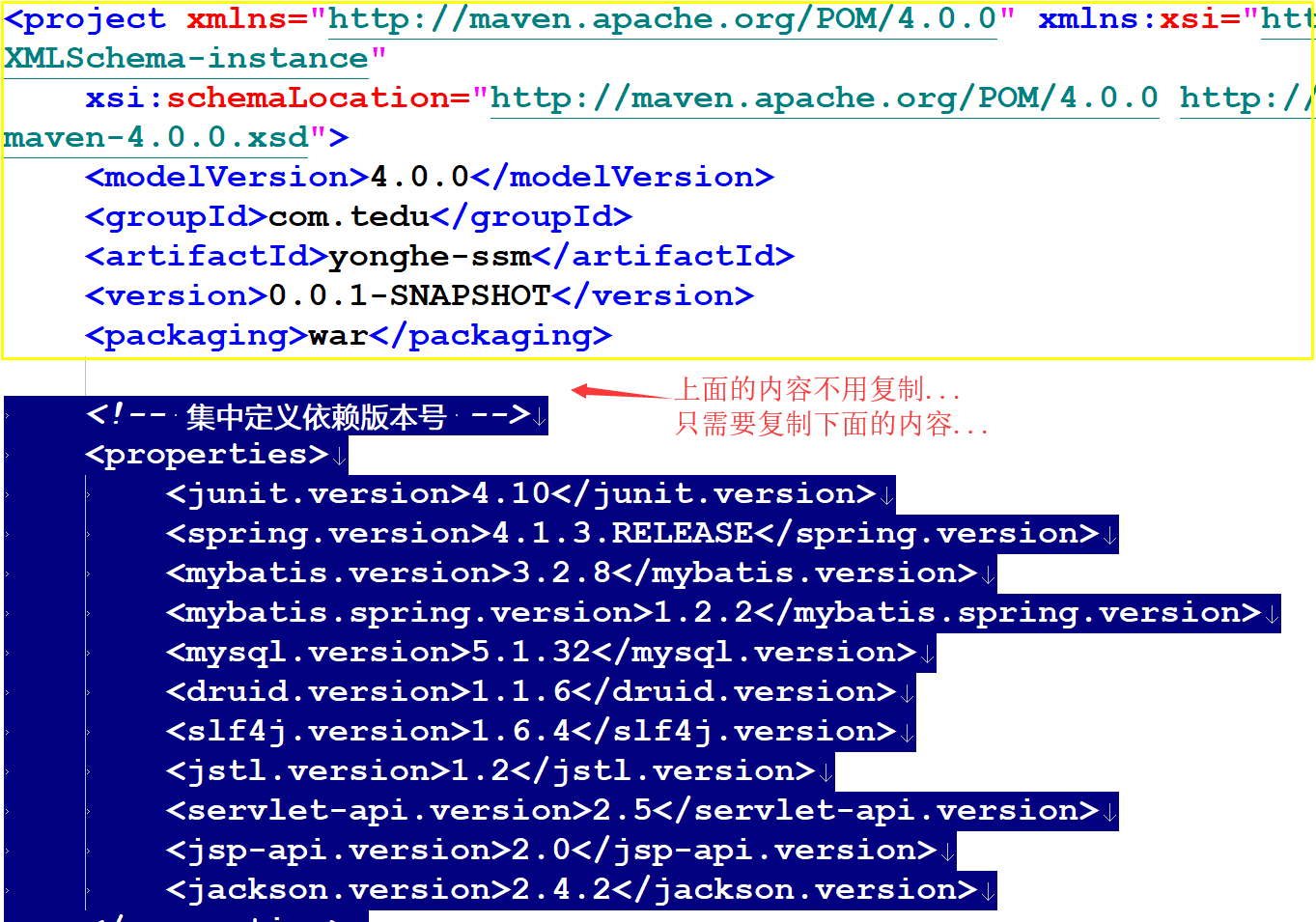
**3、将解压后的目录中的src目录选中并复制。**



**4、点击新创建的Maven工程右键粘贴，将复制的src目录粘贴到新建的工程中**



**5、打开解压后的目录中的pom.xml文件, 复制其中的内容：**



如果复制后项目或pom文件仍然报错, 可以更新Maven工程

**更新Maven工程：**在项目上右键选择 **"Maven"** ---> **"Update Project…"**，在弹出的窗口中直接点击OK即可！

**6、执行SQL脚本文件，导入数据**

打开cmd，连接mysql数据库，执行**yonghedb.sql**中的SQL语句，创建数据库、表及插入记录。

**7、部署项目到服务器并启动服务器，访问测试**

在正确完成上面的操作后，打开浏览器访问如下地址：<http://localhost/yonghe-ssm/index> ，可以看到如下界面：



## 依赖管理

### 依赖(jar包)管理

依赖管理即jar包的管理，那么jar包来自哪里？

**1、**在Maven创建的项目中，如果需要引用jar包，只需要在项目的pom.xml文件中添加jar包的坐标(GroupID + ArtifactID + Version)即可将jar包引进项目中，之后就可以在项目中使用所引入的jar包了。

例如，现在我们在pom.xml文件中，添加servlet的jar包的坐标如下：

<dependency>

<groupId>**javax.servlet**</groupId>

<artifactId>**servlet-api**</artifactId>

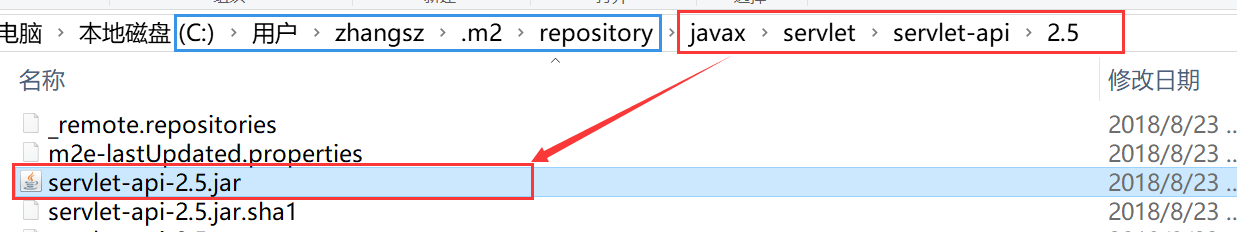
<version>**2.5**</version>

<scope>provided</scope>

</dependency>

**2、**那么问题来了，在pom文件中添加的servlet的jar包坐标，添加的仅仅是一个jar包对应的坐标，而这个jar包到底存放在哪里呢?

了解Maven管理jar包的规则之后，我们可以找到jar包的存放位置是在**本地仓库**（仓库默认是在：C:\Users\{当前用户}\.m2\repository）的: /**javax**/**servlet**/**servlet-api**/**2.5**/目录下，例如：



**从上面的例子中可以看出：**

**(1)** 在Maven中，所有的jar包全都存放在本地仓库的目录中，如果项目中需要使用某一个jar包，直接在项目的pom.xml文件中通过坐标(GroupID + ArtifactID + Version)引入指定位置的jar包即可。

**(2)** 这样可以将项目中所有使用的jar包集中在一个目录（本地仓库）中统一进行管理，需要时通过坐标直接引入即可，而不是在每个项目中都拷贝一份，减少了项目体积，也节省了磁盘空间。

**(3)** 将来如果别人需要导入你的项目，只需要将项目（当然包括pom.xml文件）代码整体传给对方，无需将jar包发送给对方，对方在配置Maven的环境后， Maven会自动根据项目中pom.xml文件里配置的坐标，引入（或下载后再引入）对应的jar包。

如果引入的jar包在本地仓库中没有呢？

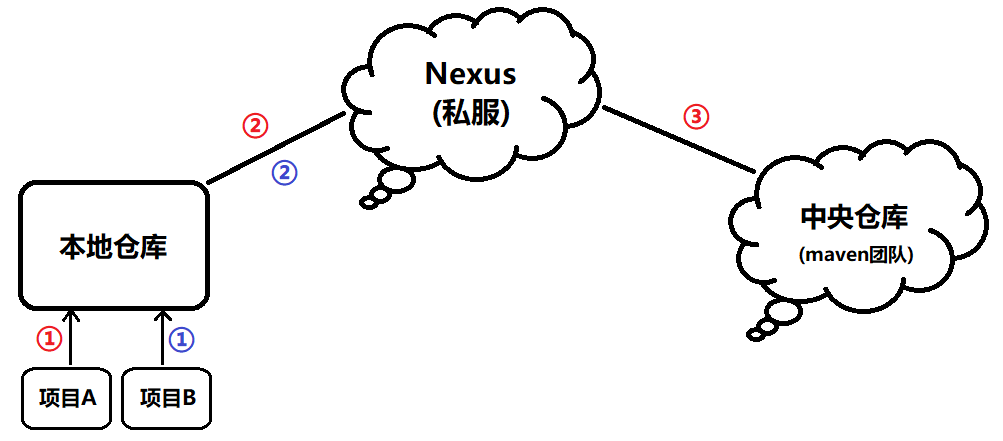
如果是刚配置的Maven环境，本地仓库中还没有太多jar包，此时在pom文件中通过坐标引入jar包，而本地仓库中没有这个jar包，这时会怎么样呢？

若本地仓库没有所需要的jar包，则会到镜像仓库（也叫私服）或者中央仓库（也叫公服）中下载。下面我们就来介绍Maven的这三种仓库。

### Maven三种仓库

在上面所提到的本地仓库、镜像仓库、中央仓库是用来Maven用来更好的管理jar包的所采用的一种方式。下面来了解Maven的**三种仓库**，以及三种仓库之间的潜在联系。

通过maven构建的项目，会通过项目中的pom.xml文件从远程仓库下载，并保存到本地仓库



**本地仓库**：默认的本地仓库位置在：c:/${user.dir}/.m2/repository，其中${user.dir}表示windows下的用户目录。本地仓库的作用是，用于保存（存储）从私服或者从中央仓库下载下来的jar包（或插件）。当项目中需要使用jar包和插件时，优先从本地仓库查找。

如果本地仓库中没有所需的jar包，可以到私服或者到中央仓库中下载后再保存到本地仓库。

**镜像仓库**：镜像仓库也叫做私服（Nexus），私服一般由公司搭建并维护（也可以自己搭建）。比如达内有搭建自己的私服服务器（**http://maven.tedu.cn/nexus/content/groups/public/**），以及阿里云私服服务器（**<http://maven.aliyun.com/nexus>/content/groups/public/**）。

如果项目中使用到的jar包或者插件本地仓库没有，则可以到私服中下载，如果私服中有就直接将jar包保存到本地仓库中；而如果私服中也没有所需的jar包，就到公服上下载所需要的jar包，下载之后先在私服上保存一份，最后再保存到本地仓库。

**中央仓库**：中央仓库也叫做公服，在maven软件中内置了一个仓库地址（<http://repo1.maven.org/maven2>）它就是中央仓库，服务于整个互联网，由Maven团队自己维护，里面存储了非常全的jar包，它包含了世界上大部分流行的开源项目的jar包。

那么我们在使用Maven构建的Java项目，项目中所使用的jar包会来自哪里呢？

例如，通过Maven先后构建项目A和项目B，在项目中都需要依赖第三方jar包：

1) 如果项目A中需要依赖第三方jar包，只需要在项目下的pom文件中引入jar包在**本地仓库**中的坐标即可使用。如果本地仓库没有所需要的jar包，则会连接私服（需要提前配置）下载所需jar包到本地仓库供项目使用。

2) 如果私服上也没有所需的jar包，则会连接中央仓库下载所需要的jar包保存到私服，再将jar包从私服下载至本地仓库，供项目使用。

3) 如果没有配置私服，则默认连接中央仓库下载所需要的jar包到本地仓库中供项目使用

4) 当项目B也需要依赖第三方jar包时，先到本地仓库中查找所需jar包，如果有则直接引用而无需再次下载，如果仍有部分jar包本地仓库中没有，则同上，即连接私服下载所需jar包到本地仓库。若私服中也没有所需jar包，则连接中央仓库下载jar包到私服，再从私服下载jar包到本地仓库中，供项目使用。

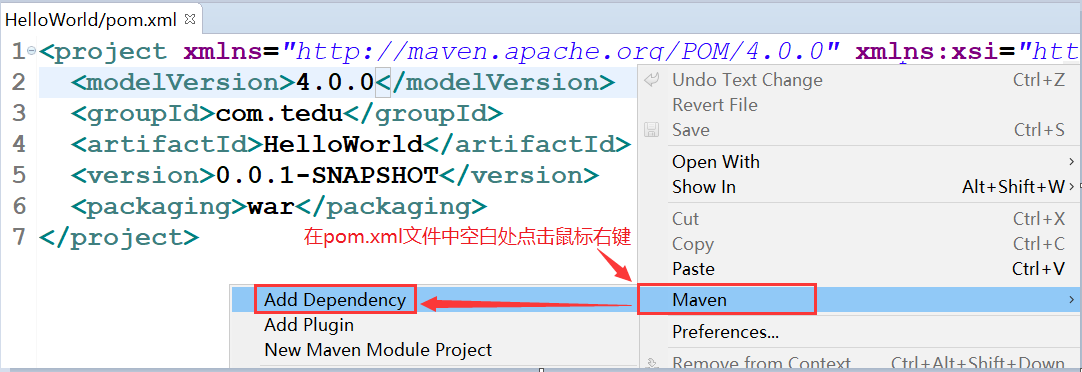
## 添加依赖

### 添加依赖：第一种方式

使用maven插件的索引功能快速添加jar包

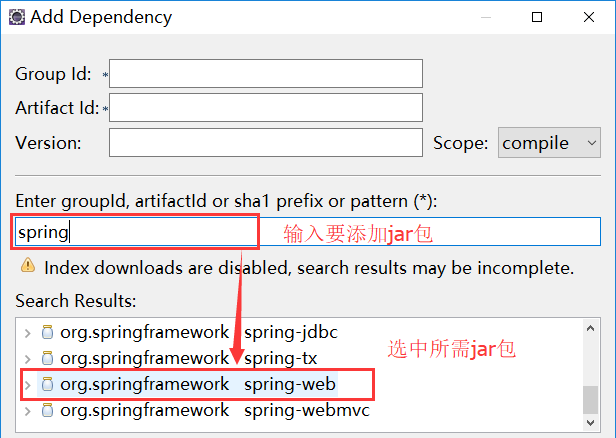
这种方式需要**本地仓库中已经包含了该jar包，否则搜索不到**!!!

1、如果本地仓库中有我们需要的jar包，可以在项目中的pom.xml文件中空白处右键--> Maven --> Add Dependency在弹出的窗口中添加所需要的依赖(jar包)，如图：

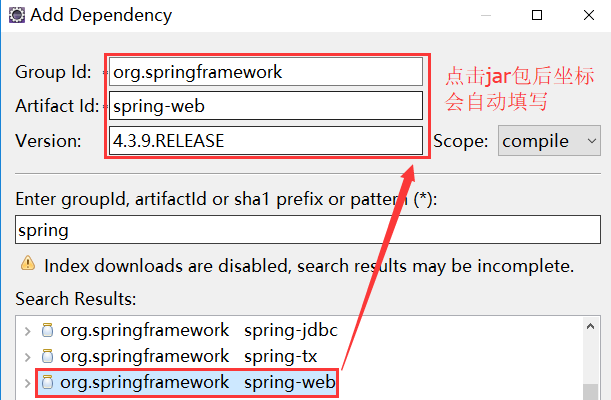


**2、**添加依赖示例：添加spring的jar包的坐标到项目中

(1) 在项目中的pom.xml文件中右键 -> Maven -> Add Dependency，在弹出的窗口中输入 "spring":

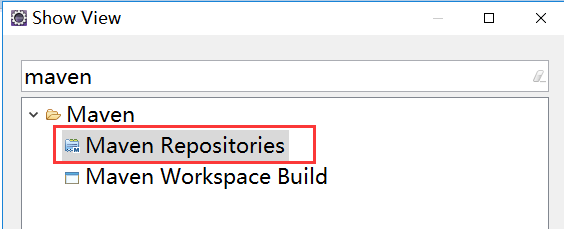


选中要添加的jar包（坐标会自动填写），点击OK即可完成添加



(2)如果搜索不到jar包（保证本地仓库中已经下载了该jar包），可以尝试重建索引。

在



"Maven Repositories" 视图窗口中可以看到如下内容：



在"Local Repositories"上右键选择 "Rebuild Index" 即可重建索引。

完成后，再尝试搜索jar包进行添加。

### 添加依赖：第二种方式

1、直接在pom.xml文件中的<dependencies>标签内部添加。例如：在pom.xml文件中添加如下配置，就可以将junit单元测试的jar包引入到项目中来了。

添加依赖：

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.9</version>

<scope>test</scope>

<dependency>

<dependencies>

2、手动添加依赖需要指定所依赖jar包的坐标，但是在大部分情况下，我们是不知道jar包的坐标的。可以通过访问如下网址，在互联网上搜索查询：

<http://mvnrepository.com>

或者在公司镜像仓库中搜索查询:

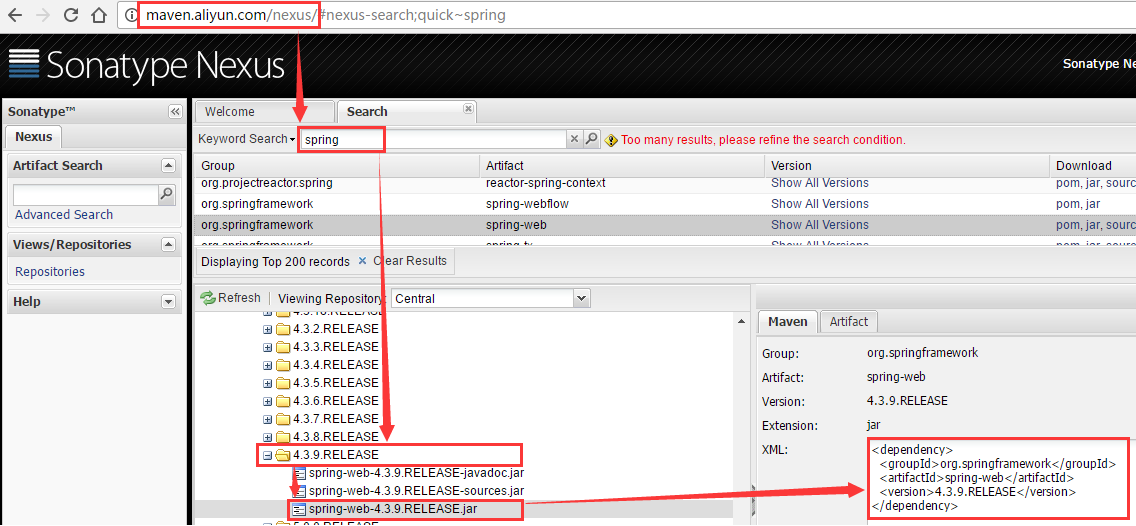
<http://maven.tedu.cn/nexus>

3、示例：添加spring的jar包的坐标到项目中

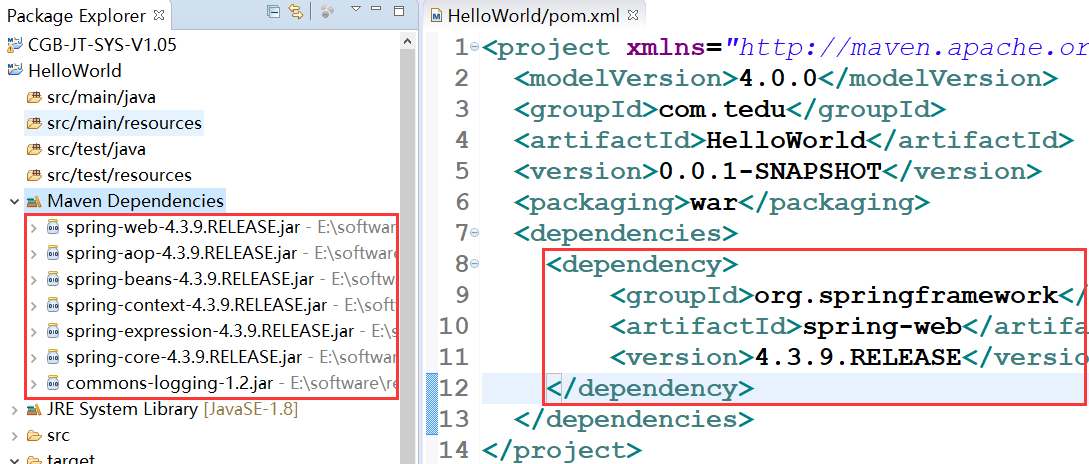
(1)访问上面其中的一个网址，在搜索框中搜索 "spring"

(2)在搜索出来的内容中，选择所需要的版本并点击版本，查看该版本的spring jar包所对应的坐标：

(3)将坐标直接拷贝到项目的pom.xml文件中即可：



4、将上图中右侧选中的坐标拷贝到pom.xml文件中:



## FAQ：常见问题

### 常见的Maven环境问题

在实际开发中，Maven的环境很容易遭到破坏。而Maven的环境一旦遭到破坏，项目运行就会失败，例如在项目上右键点击运行，在项目启动的过程中，报了如下错误：

...

[INFO] ------------------------------------------------------------

[ERROR] **No plugin found** for prefix 'tomcat7' in the current project and in the plugin groups [**org.apache.maven.plugins**, **org.codehaus.mojo**] available from the repositories [local (C:\Users\bjzhangsz\.m2\repository), nexus-tedu (http://maven.tedu.cn/nexus/content/groups/public/)] -> [Help 1]

[ERROR]

[ERROR] To see the full stack trace of the errors, re-run Maven with the -e switch.

[ERROR] Re-run Maven using the -X switch to enable full debug logging.

[ERROR]

[ERROR] For more information about the errors and possible solutions, please read the following articles:

[ERROR] [Help 1] <http://cwiki.apache.org/confluence/display/MAVEN/NoPluginFoundForPrefixException>

...

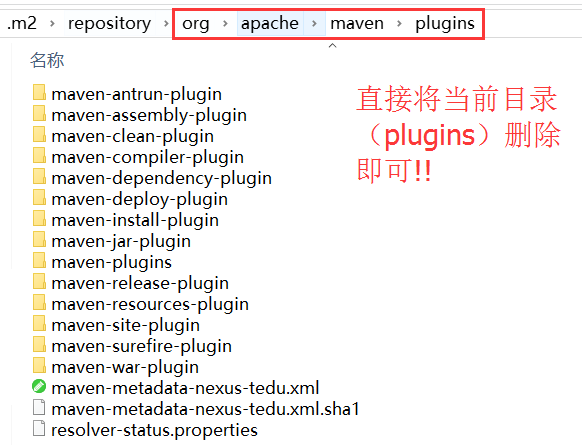
**问题分析：**从上面的错误信息中，我们可以提取主要的报错信息，可以发现是本地仓库中的 [org.apache.maven.plugins] 和 [org.codehaus.mojo] 两个目录中的插件找不到了。

**解决方案：**由于这两个目录中插件较多，不知道是哪一个插件丢失或遭破坏造成的，所以，我们可以将两个目录直接删除，让Maven再重新下载一次即可！

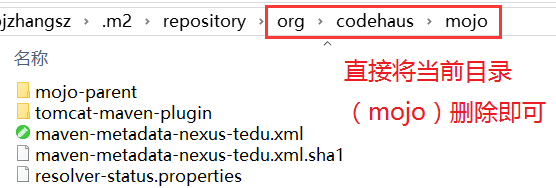
需要注意的是，此时直接删除会提示目录或文件正在被占用，需要将eclipse先关闭，待删除目录后，启动eclipse，重新更新工程（Update Project…），重新下载删除的内容即可解决！

(0)保存eclipse中的代码，关闭eclipse。

(1)删除本地仓库中的所缺少的目录, 比如上面提示: org.apache.maven.plugins(根据上面错误信息提示的jar包路径选择删除)目录：



(2)删除本地仓库中的org.codehaus.mojo(根据上面错误信息提示的jar包路径选择删除)目录：

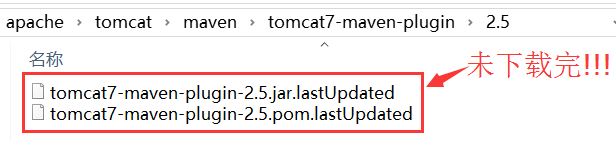


### 找不到jar包问题

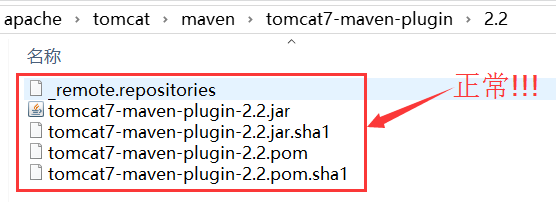
在项目中通过坐标引入了jar包（或者插件），并且本地仓库中也存在对应的jar包，但是项目还是报错，提示内容说找不到。

解决方法：如果引入的jar包，在本地仓库中存在，但是还是提示找不到，可以将本地仓库中jar包或插件的所在目录整个删除(如果删除时提示文件正在被占用，关闭eclipse再删除即可)，重新保存pom.xml文件，并更新工程，让maven再次下载上面的jar包即可！

未下载完全示例：



正常下载完全示例：

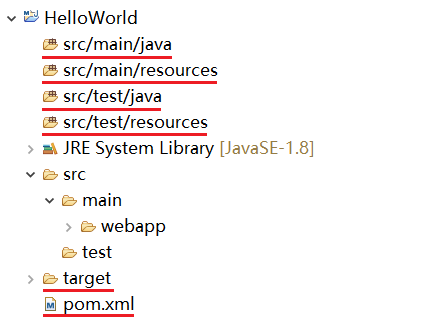


### 拷贝别Maven仓库

如果因为网络环境的问题，导致jar包无法下载，也可以将别人下载好的（完整的）Maven的本地仓库拷贝过来，放在自己配置的本地仓库中。因为Maven可以支持拷贝别人的仓库。

## 扩展内容：Maven项目结构

使用Maven创建的工程我们称它为Maven工程，Maven工程具有一定的目录规范，对目录结构有严格的要求，一个Maven工程要具有如下目录结构：



Maven项目目录介绍：

1）/src/main/java -- 主目录下的Java目录，用于存放项目中的.java文件

2）/src/main/resources – 主目下的资源目录，存放项目中的资源文件(如框架的配置文件)

3）/src/test/java -- 测试目录下的Java目录，用于存放所有单元测试类的.java文件，如Junit测试类

4）/src/test/resources – 测试目录下的资源目录，用于存放测试类所需资源文件(如框架的配置文件)

5）/target -- 项目输出目录，编译后的class文件、及项目打成的war包等会输出到此目录中

6）/pom.xml -- maven项目的核心配置文件，文件中通过坐标来管理项目中的所有jar包和插件。

**思考**: Maven的目录中为什么将测试类及相关文件单独存放?

Maven的pom.xml文件

通过Maven构建的项目中，有一个名称为pom的xml文件，这个文件的Maven项目的核心配置文件。通过该文件可以配置Maven工程所依赖的jar包或者是插件。

例1：通过pom.xml文件配置Maven工程所依赖的jar包。

<dependencies>

<!-- 添加Junit单元测试jar包 -->

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.9</version>

<scope>test</scope>

<dependency>

<dependencies>

例2：通过pom.xml文件配置Maven工程所依赖的插件。

<build>

<plugins>

<!-- 添加JDK插件(版本1.7) -->

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.5.1</version>

<configuration>

<source>1.7</source>

<target>1.7</target>

<encoding>UTF-8</encoding>

</configuration>

</plugin>

</plugins>

</build>

在pom.xml文件中配置了所依赖的jar包或者插件后，Maven会根据文件中的配置信息到本地仓库中寻找指定的jar包或者插件，如果本地仓库中没有就会到私服或者中央仓库中下载相应的jar包或者插件。再引入到项目中使用！

# Day13 Cookie、Session

## 什么是会话

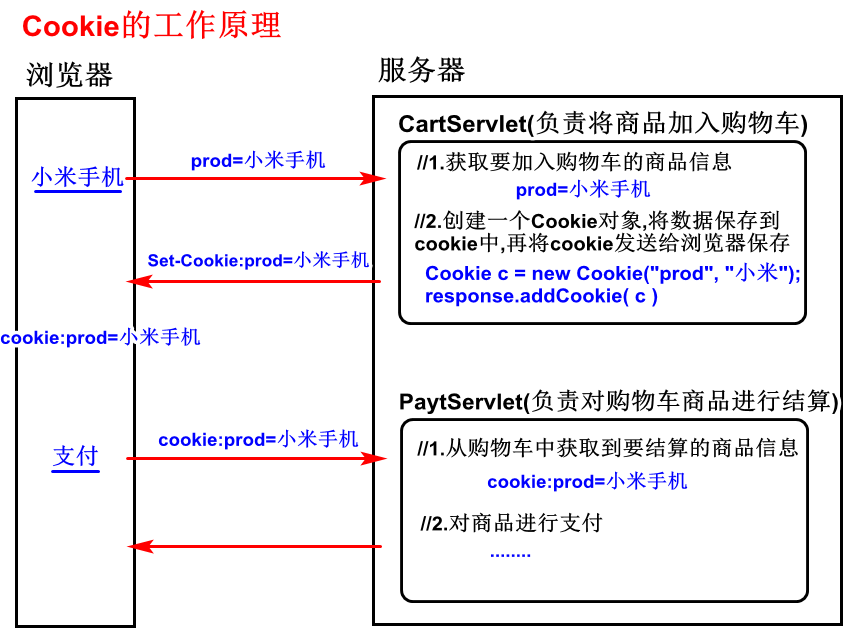
会话：从浏览器打开，访问服务器开始，一直到访问服务器结束，浏览器关闭为止，这期间浏览器和服务器之间产生的多次请求和响应加在一起，就称之为浏览器和服务器之间的一次会话。

在一次会话中，往往会产生一些数据，而这些数据又需要我们保存起来，如何保存会话中产生的数据？

可以使用cookie或者session保存会话中产生的数据。

## Cookie

### Cookie的工作原理



Cookie是将会话中产生的数据保存在客户端浏览器内部。Cookie是客户端技术。

Cookie是基于两个头进行工作的，分别是Set-Cookie响应头和Cookie请求头。

通过Set-Cookie响应头负责将Cookie信息发送给浏览器,让浏览器以Cookie的形式保存在内部(保存cookie)

浏览器内部一旦保存了Cookie，以后每次访问服务器都会通过一个Cookie请求头，将cookie再带给服务器（除非cookie过期）

### Cookie的API及应用

**1、创建Cookie对象**

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

-- 创建一个Cookie对象, 同时指定Cookie的名字和cookie中保存的值。

**2、将Cookie添加到response响应中**

response.addCookie( Cookie c );

-- 通过response提供的addCookie方法，将cookie对象保存到响应中。再发送给浏览器保存

-- 可以多次调用该方法，即可以添加一个以上的cookie到响应中。

**3、获取请求中的所有cookie对象组成的数组**

Cookie[] cs = request.getCookies();

-- 从请求中获取所有cookie对象组成的数组，如果请求中没有任何cookie，该方法将会返回null值。（不是空的数组）

**4、删除Cookie**

没有提供直接删除Cookie的方法。

可以向浏览器再发送一个同名的Cookie，设置Cookie的最大存活时间为零。

由于浏览器是根据Cookie的名字区分一个Cookie，因此两次发送相同名字的Cookie，浏览器会认为是同一个，后发送的Cookie将会覆盖之前发送的Cookie。而后发的Cookie，由于设置了生存时间为零，所以浏览器收到后也会立即删除！

代码示例：

**protected** **void** doGet(HttpServletRequest request, HttpServletResponse response)

**throws** ServletException, IOException {

response.setContentType("text/html;charset=utf-8");

PrintWriter out = response.getWriter();

//1.创建一个同名的Cookie,并设置生存时间为零

Cookie c = **new** Cookie("prod", "");

c.setMaxAge( 0 );

//2.将cookie添加到响应中,发送给浏览器

response.addCookie( c );

//3.做出响应。

out.write("成功删除了名称为prod的cookie!");

}

**5、Cookie的常用方法**

Cookie.getName() -- 获取cookie的名字

Cookie.getValue() -- 获取cookie中保存的值

Cookie.setValue() -- 设置cookie中的值

**6、setMaxAge方法**

-- 设置cookie的最大存活时间。

-- 如果不设置该方法，cookie默认是保存在浏览器的内存中，在浏览器关闭时，随着内存的释放，cookie也会销毁。同时，浏览器关闭，会话也结束了。因此默认cookie的存活时间是一次会话。

-- 如果要保证浏览器关闭后再打开Cookie仍然存在，可以为cookie设置一个最大生存时间。通过SetMaxAge方法可以设置cookie最大存活时间，设置后，cookie将不会保存在浏览器内存里，而是以文件形式保存在浏览器的临时目录下（硬盘上）

例如：

//2.创建一个cookie,将商品信息保存到cookie中

Cookie cookie = **new** Cookie( "prod", prod );

//设置cookie最大生存时间, 单位是秒

cookie.setMaxAge( 3600 \* 24 \* 30 );

//3.将cookie添加到响应中.

response.addCookie( cookie );

### Cookie中保存中文数据的问题

HTTP协议中规定了请求信息和响应信息中不能包含中文数据!!

因此通过浏览器向服务器发送中文数据时，浏览器会将中文数据进行URL编码，编码为下面这种格式：



http://localhost/day13-cookie/index.html?user=%E5%BC%A0%E9%A3%9E%E9%A3%9E

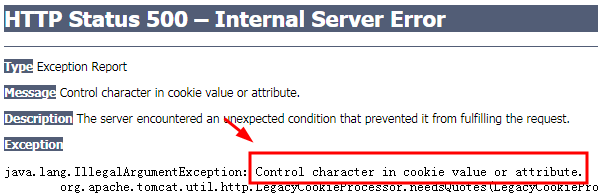
将中文数据转成下面这种格式，叫做URL编码：

张飞飞 ---**URL编码**---> %E5%BC%A0%E9%A3%9E%E9%A3%9E

将下面这种格式再次转回中文数据，叫做URL解码：

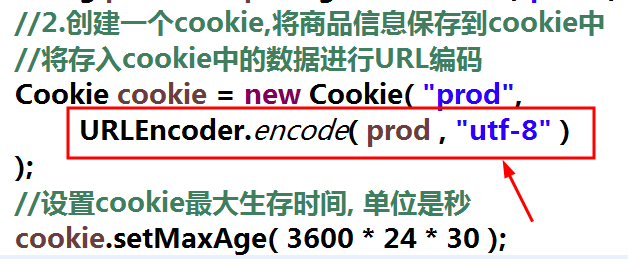
%E5%BC%A0%E9%A3%9E%E9%A3%9E ---**URL解码**---> 张飞飞

**问题：当cookie中保存中文数据，将cookie添加到响应中时，会报一个500异常，如下：**

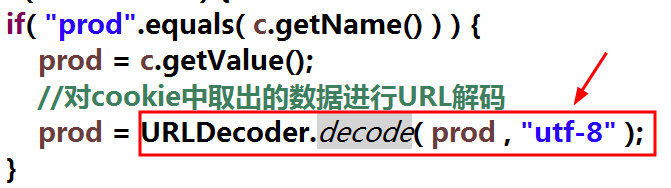


解决方法是：

(1)将存入cookie中的先进行URL编码，再存入Cookie中



(2)从cookie取出来的数据是进行URL编码后的数据，在使用之前需要进行URL解码：



### 案例：实现购物车

1.index.html

<body>

<!--

http://localhost/day13/CartServlet

http://localhost/day13/index.html

-->

<h3>点击下面商品,可以将商品加入购物车</h3>

<p><a href=*"CartServlet?prod=iphone11"*>iphone11</a></p>

<p><a href=*"CartServlet?prod=huaweimeta30"*>huaweimeta30</a></p>

<p><a href=*"CartServlet?prod=vivonex3"*>vivonex3</a></p>

<p><a href=*"CartServlet?prod=xiaomi"*>xiaomi</a></p>

<h3>点击下面商品,可以将商品加入购物车</h3>

<a href=*"PayServlet"*>为购物车中的商品进行支付</a>

</body>

2.CartServlet

**protected** **void** doGet(HttpServletRequest request, HttpServletResponse response)

**throws** ServletException, IOException {

response.setContentType("text/html;charset=utf-8");

PrintWriter out = response.getWriter();

//1.获取请求中携带的商品信息

String prod = request.getParameter("prod");

//2.创建一个cookie,将商品信息保存到cookie中

//将存入cookie中的数据进行URL编码

Cookie cookie = **new** Cookie( "prod",

URLEncoder.*encode*( prod , "utf-8" )

);

//设置cookie最大生存时间, 单位是秒

cookie.setMaxAge( 3600 \* 24 \* 30 );

//3.将cookie添加到响应中.

response.addCookie( cookie );

//4.做出响应.

out.write("成功将 ["+prod+"] 加入了购物车");

}

3.PayServlet

**protected** **void** doGet(HttpServletRequest request, HttpServletResponse response)

**throws** ServletException, IOException {

response.setContentType("text/html;charset=utf-8");

PrintWriter out = response.getWriter();

//1.获取请求中的所有cookie信息(cookie对象数组)

Cookie[] cs = request.getCookies();

//2.遍历cookie数组, 获取名称为prod的cookie

String prod = **null**;

**if**( cs != **null** ) {

**for** (Cookie c : cs) {

**if**( "prod".equals( c.getName() ) ) {

prod = c.getValue();

//对cookie中取出的数据进行URL解码

prod = URLDecoder.*decode*( prod , "utf-8" );

}

}

}

**if**( prod == **null** ) {

out.write( "您还没有将任何商品加入购物车..." );

}**else** {

out.write( "成功为 ["+prod+"] 支付了1000元..." );

}

}

## Session

### Session的工作原理



(1)Session是将会话中产生的数据保存在服务器端，属于服务器端技术

(2)一个浏览器在服务器端至多只对应一个Session对象。当需要session时，如果服务器内部有当前浏览器对应的session，则不再创建，而是直接返回已有session。如果没有对应session，才会创建一个新的session返回。

(3)通过session.setAttribute方法将数据保存在session域中

(4)通过session.getAttribute方法将数据从session中取出!

### Session是一个域对象

Session是一个域对象，因此session中也提供了存取数据的方法。

Session.setAttribute(String attrName, Object attrValue)

-- 将属性保存到session域中，属性名只能是字符串，属性值可以是任意类型。

session.getAttribute(String attrName)

-- 根据属性名获取对应的属性值，返回值是Object类型

Session域对象的三大特征：

(1)生命周期：

**创建session:** 第一次调用request.getSession方法时，创建session对象。或者当服务器中没有当前浏览器对应的session时，会创建一个新的session对象。

request.getSession();

request.getSession(**true**);

-- 上面两个获取session方法，其作用是：当服务器内部有session时，直接返回，当服务器内部没有为当前浏览器服务的session，就创建一个seession再返回。

request.getSession(**false**);

-- 而上面这个方法，其作用是：当服务器内部有session时，直接返回session，当服务器内部没有session，返回null。（即该方法只会返回现成的session对象，否则就返回null）

**销毁session：**

a)超时销毁: 当session超过30分钟没有被访问时, session将会超时销毁。可以通过方法设置或者web.xml文件修改session的超时时间,但是一般我们不会修改。

b)自杀销毁: 当调用session.invalidate方法时, session会立即销毁。

c)意外身亡: 当服务器非正常关闭（硬件损坏、断电、结束服务器的进程等），session会随着服务器的关闭而销毁。

当服务器正常关闭时，session不会销毁。在服务器关闭之前，会将服务器内部所有的session以文件形式序列化保存到硬盘上（保存在服务器的work目录下），这个过程叫做session的序列化或者钝化；当再次打开服务器，文件还会在回到服务器变回对象。这个过程叫做session反序列化或者活化。

(2)作用范围：一次会话范围内

(3)主要功能：在一次会话范围内，实现数据的共享。

### Session获取不到的问题

将商品保存到session中后，关闭浏览器再打开浏览器，访问服务器，此时获取不到之前的session。因为session是基于Cookie工作的。

在服务器创建一个session后，会为session分配一个独一无二的编号，称之为session的id，在此次响应时，服务器会将session的id以一个名称为JSESSIONID的cookie发送给浏览器保存到浏览器内部。

由于保存sessionid的cookie默认的会话级别的cookie，在浏览器关闭后，cookie销毁，sessionid也丢失了。因此下次访问服务器，没有session的id就获取不到之前的session。也获取不到session中的商品信息

解决方法：我们可以创建一个名称为JSESSIONID的cookie，其中保存session的ID，并设置cookie的最大存活时间，让cookie保存到硬盘上（即使浏览器关闭，cookie也不会销毁），这样下次访问服务器时，还可以将sessionid带给服务器，服务器可以通过sessionid获取到之前的session。从session中获取到商品信息



### 案例:实现购物车

1.prodlist.html

<body>

<!--

http://localhost/day13/CartServlet

http://localhost/day13/index.html

-->

<h3>点击下面商品,可以将商品加入购物车</h3>

<p><a href=*"CartServlet?prod=iphone11"*>iphone11</a></p>

<p><a href=*"CartServlet?prod=huaweimeta30"*>huaweimeta30</a></p>

<p><a href=*"CartServlet?prod=vivonex3"*>vivonex3</a></p>

<p><a href=*"CartServlet?prod=xiaomi">xiaomi</a></p>*

*<p><a href="CartServlet?prod=小米8">小米8</a><*/p>

<h3>点击下面商品,可以将商品加入购物车</h3>

<a href=*"PayServlet"*>为购物车中的商品进行支付</a>

</body>

2.CartServlet

**protected** **void** doGet(HttpServletRequest request, HttpServletResponse response)

**throws** ServletException, IOException {

response.setContentType("text/html;charset=utf-8");

PrintWriter out = response.getWriter();

//1.获取要加入购物车的商品信息

String prod = request.getParameter("prod");

//2.获取一个session对象

HttpSession session = request.getSession();

//设置session的超时时间 单位是秒

//session.setMaxInactiveInterval( 30 );

//创建一个名称为JSESSIONID的cookie,保存session的id值

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

c.setMaxAge( 3600\*24 );//设置一天

response.addCookie( c );

//3.将商品信息保存到session中

session.setAttribute("prod", prod );

//4.做出响应

out.write("成功将 ["+prod+"] 加入了购物车");

}

3.PayServlet

**protected** **void** doGet(HttpServletRequest request, HttpServletResponse response)

**throws** ServletException, IOException {

response.setContentType("text/html;charset=utf-8");

PrintWriter out = response.getWriter();

//1.获取之前的session对象。

HttpSession session = request.getSession( **false** );

**if**( session != **null** ) {

//2.从session中取出商品信息

String prod = (String) session.getAttribute( "prod" );

//3.模拟支付

out.write( "成功为 ["+prod+"] 支付了 2000元...." );

}**else** {

out.write( "您还没有将商品加入购物车...." );

}

}

## 总结：Cookie和session的区别

### Cookie的特点

1、cookie是将会话中产生的数据保存在**浏览器客户端**, 是客户端技术

2、cookie是将数据保存在客户端浏览器，容易随着用户的操作导致cookie丢失或者被窃取，因此cookie中保存的数据不太稳定，也不太安全。

但cookie将数据保存在客户端，对服务器端没有太多影响，可以将数据保存很长时间。

因此cookie中适合存储需要长时间保存的数据，但对安全性要求不高。

3、浏览器对cookie的大小和个数都有限制，一般推荐每一个站点给浏览器发送的cookie数量不超过20个，每一个cookie的大小不超过1kb。

Cookie的应用：实现购物车、记住用户名、30天内自动登录等。

### Session的特点

1、session是将会话中产生的数据保存在**服务器端**，是服务器端技术

2、session将数据存在服务器端的session对象中，相对更加的安全，而且更加稳定。不容易随着用户的操作而导致session中的数据丢失或者是被窃取。

但session是服务器端的对象，在并发量较高时每一个浏览器客户端在服务器端都要对应一个session对象，占用服务器的内存空间，影响效率。

因此session中适合存储对安全性要求较高，但不需要长时间保存的数据。

Session的应用：保存登录状态、保存验证码

# 数据库事务

今日目标:

(1)了解事务的作用

(2)掌握事务的四大特性(面试)

(3)了解事务的三个读问题

(4)掌握mysql开启和结束事务

(5)了解事物的四个隔离级别

## 事务及四大特性

### 什么是事务

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。

简单的说：事务就是将一堆的SQL语句(通常是增删改操作)绑在一起执行，要么都执行成功，要么都执行失败，即都执行成功才算成功，否则就会恢复到这堆SQL执行之前的状态。

下面以银行转账为例，张三转100块到李四的账户，这至少需要两条SQL语句：

* 给张三的账户减去100元；

update 账户表 set money=money**-100** where name='张三';

* 给李四的账户加上100元。

update 账户表 set money=money**+100** where name='李四';

如果在第一条SQL语句执行成功后，在执行第二条SQL语句之前，程序被中断了（可能是抛出了某个异常，也可能是其他什么原因），那么李四的账户没有加上100元，而张三却减去了100元，在现实生活中这肯定是不允许的。

如果在转账过程中加入事务，则整个转账过程中执行的所有SQL语句会在一个事务中，而事务中的所有操作，要么全都成功，要么全都失败，不可能存在成功一半的情况。

也就是说给张三的账户减去100元如果成功了，那么给李四的账户加上100元的操作也必须是成功的；否则，给张三减去100元以及给李四加上100元都是失败的。

### 事务的四大特性

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，在事中务查看数据更新时，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

在默认情况下，MySQL每执行一条SQL语句，都是一个单独的事务。如果需要在一个事务中包含多条SQL语句，那么需要开启事务和结束事务。

* 开启事务：start transaction；
* 结束事务：commit（提交事务）或rollback（回滚事务）。

在执行SQL语句之前，先执行strat transaction，这就开启了一个事务（事务的起点），然后可以去执行多条SQL语句，最后要结束事务，**commit**表示提交，即事务中的多条SQL语句所做出的影响会持久化到数据库中。或者**rollback**，表示回滚，即回滚到事务的起点，之前做的所有操作都被撤消了！

下面演示A账户给B账户转账的例子：

\*\* 准备数据：

-- 1、创建数据库jt\_db数据库(如果不存在才创建)

create database if not exists jt\_db charset utf8;

use jt\_db; -- 选择jt\_db数据库

-- 2、在 jt\_db 库中创建 acc 表(银行账户表), 要求有id(主键),name(姓名),money(账户金额)

drop table if exists acc;

create table acc(

id int primary key auto\_increment,

name varchar(50),

money double

);

-- 3、往 acc 表中, 插入2条记录

insert into acc values(null,'A',1000);

insert into acc values(null,'B',1000);

-- 查询acc表中的所有记录

select \* from acc;

下面分别演示事务开启及执行一系列SQL之后，回滚事务、提交事务及中断操作的效果。

rollback（回滚事务）：

-- 查询acc账户表中A和B的金额

select \* from acc;

-- 开启事务

start transaction;

-- 开始转账，A账户减去100元

update acc set money=money-100 where name='A';

-- 查询acc账户表中A和B的金额

select \* from acc;

-- B账户增加100元

update acc set money=money+100 where name='B';

-- 查询acc账户表中A和B的金额

select \* from acc;

-- 回滚事务

rollback;

-- 再次查询acc账户表中A和B的金额

select \* from acc;

commit（提交事务）：

-- 将上面的操作再做一次，最后将rollback替换为commit，即提交事务

commit;

中断操作

-- 将上面的操作再做一次，最后将rollback替换为quit，即中断操作

quit;

## 事务并发读问题

### 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。

在事务并发时，如果没有采取必要的隔离措施，可能会导致各种并发问题，破坏数据的完整性等。这些问题中，其中有三类是读问题，分别是：脏读、不可重复读、幻读。

(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

(3)**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)；

注意:mysql默认的是不允许出现脏读和不可重复读，所以在下面演示之前需要设置mysql允许出现脏读、不可重复读等。

set tx\_isolation='read-uncommitted';

脏读示例：

-- 在窗口1中，开启事务，执行A给B转账100元

set tx\_isolation='read-uncommitted'; -- 允许脏读、不可重复读、幻读

use jt\_db; -- 选择jt\_db库

start transaction; -- 开启事务

update acc set money=money-100 where name='A';

update acc set money=money+100 where name='B';

-- 在窗口2中，开启事务，查询B的账户金额

set tx\_isolation='read-uncommitted'; -- 允许脏读、不可重复读、幻读

use jt\_db; -- 选择jt\_db库

start transaction; -- 开启事务

select \* from acc where name='B'; -- 出现脏数据

-- 切换到窗口1，回滚事务，撤销转账操作。

rollback; -- 回滚事务

-- 切换到窗口2，查询B的账户金额

select \* from acc where name='B';

在窗口2中，B看到自己的账户增加了100元（此时的数据A操作事务并未提交），此种情况称之为"脏读"。

不可重复读示例：

-- 在窗口1中，开启事务，查询A账户的金额

set tx\_isolation='read-uncommitted'; -- 允许脏读、不可重复读、幻读

use jt\_db; -- 选择jt\_db库

start transaction; -- 开启事务

select \* from acc where name='A';

-- 在窗口2中，开启事务，查询A的账户金额减100

set tx\_isolation='read-uncommitted'; -- 允许脏读、不可重复读、幻读

use jt\_db; -- 选择jt\_db库

start transaction; -- 开启事务

update acc set money=money-100 where name='A'; -- A账户减去100

select \* from acc where name='A';

commit; -- 提交事务

-- 切换到窗口1，再次查询A账户的金额。

select \* from acc where name='A'; -- 前后查询结果不一致

在窗口1中，前后两次对同一数据(账户A的金额)查询结果不一致，是因为在两次查询之间，另一事务对A账户的金额做了修改。此种情况就是"不可以重复读"

幻读示例：

-- 在窗口1中，开启事务，查询账户表中是否存在id=3的账户

set tx\_isolation='read-uncommitted'; -- 允许脏读、不可重复读、幻读

use jt\_db; -- 选择jt\_db库

start transaction; -- 开启事务

select \* from acc where id=3;

-- 在窗口2中，开启事务，往账户表中插入了一条id为3记录，并提交事务。

-- 设置mysql允许出现脏读、不可重复度、幻读

set tx\_isolation='read-uncommitted';

use jt\_db; -- 选择jt\_db库

start transaction; -- 开启事务

insert into acc values(3, 'C', 1000);

commit; -- 提交事务

-- 切换到窗口1，由于上面窗口1中查询到没有id为3的记录，所以可以插入id为3的记录。

insert into acc values(3, 'C', 1000); -- 插入会失败!

在窗口1中，查询了不存在id为3的记录，所以接下来要执行插入id为3的记录，但是还未执行插入时，另一事务中插入了id为3的记录并提交了事务，所以接下来窗口1中执行插入操作会失败。

探究原因，发现账户表中又有了id为3的记录（感觉像是出现了幻觉）。这种情况称之为"幻读"

以上就是在事务并发时常见的三种并发读问题，那么如何防止这些问题的产生？

可以通过设置事务隔离级别进行预防。

### 事务隔离级别

事务隔离级别分四个等级，在相同数据环境下，对数据执行相同的操作，设置不同的隔离级别，可能导致不同的结果。不同事务隔离级别能够解决的数据并发问题的能力也是不同的。

set tx\_isolation='read-uncommitted';

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

### 设置隔离级别(了解)

0、MySQL查询当前的事务隔离级别

select @@tx\_isolation;

**1、MySQL设置事务隔离级别（了解）**

(1) set tx\_isolation='read-uncommitted';

安全性最差，容易出现**脏读**、**不可重复读**、**幻觉读**，但性能最高

(2) set tx\_isolation='read-committed';

安全性一般，可防止**脏读**，但容易出现**不可重复读**、**幻觉读**

(3) set tx\_isolation='repeatable-read';

安全性较好，可防止**脏读**、**不可重复读**，但是容易出现**幻读**

(4) set tx\_isolation='serialiable';

安全性最好，可以防止一切事务并发问题，但是性能最差。

**2、JDBC设置事务隔离界别**

JDBC中通过Connection提供的方法设置事务隔离级别：

Connection. setTransactionIsolation(int level)

参数可选值如下：

Connection.TRANSACTION\_READ\_UNCOMMITTED 1（读未提交数据）

Connection.TRANSACTION\_READ\_COMMITTED 2（读已提交数据）

Connection.TRANSACTION\_REPEATABLE\_READ 4（可重复读）

Connection.TRANSACTION\_SERIALIZABLE 8（串行化）

Connection.TRANSACTION\_NONE 0（不使用事务）

提示：在开发中，一般情况下不需要修改事务隔离级别

**3、JDBC中实现转账例子**

提示：JDBC中默认是自动提交事务，所以需要关闭自动提交，改为手动提交事务

也就是说, 关闭了自动提交后, 事务就自动开启, 但是执行完后需要手动提交或者回滚!!

(1)执行下面的程序，程序执行没有异常，转账成功！A账户减去100元，B账户增加100元。

(2)将第4步、5步中间的代码放开，再次执行程序，在转账过程中抛异常，转账失败！由于事务回滚，所以A和B账户金额不变。

**public** **static** **void** main(String[] args) **throws** SQLException {

Connection conn = **null**;

Statement stat = **null**;

ResultSet rs = **null**;

**try** {

//1.获取连接

conn = JDBCUtil.*getConn*();

//2.关闭JDBC自动提交事务（默认开启事务）

conn.setAutoCommit(**false**);

//3.获取传输器

stat = conn.createStatement();

/\* \*\*\*\*\* A给B转账100元 \*\*\*\*\* \*/

//4.A账户减去100元

String sql = "update acc set money=money-100 where name='A'";

stat.executeUpdate(sql);

//int i = 1/0; // 让程序抛出异常，中断转账操作

//5.B账户加上100元

sql = "update acc set money=money+100 where name='B'";

stat.executeUpdate(sql);

//6.手动提交事务

conn.commit();

System.***out***.println("转账成功！提交事务...");

} **catch** (Exception e) {

e.printStackTrace();

//一旦其中一个操作出错都将回滚，使两个操作都不成功

conn.rollback();

System.***out***.println("执行失败！回滚事务...");

} **finally**{

JDBCUtil.*close*(conn, stat, rs);

}

}

# 第二阶段课程介绍

## 讲师自我介绍



张慎政

QQ: 731471199

Tel: 18518923147

## Web课程简介

1、学习数据库（**MySQL**）的相关知识

2、学习如何利用Java程序访问数据库，实现增删改查操作（**JDBC**）

3、学习前端页面开发的相关技术（html，css，**JavaScript**，**jQuery**）

4、学习**Servlet/JSP**相关知识，为后面的框架打基础

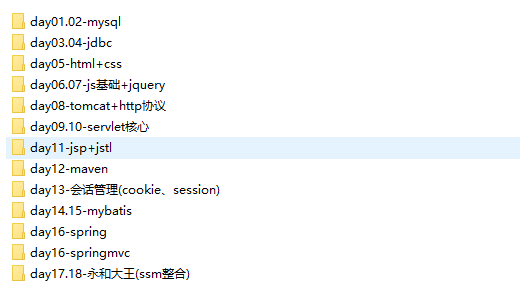
5、学习会话技术（**Cookie和Session**）

5、学习企业级开发项目管理工具-**Maven**，利用Maven构建项目

6、学习**Mybatis**框架（封装JDBC、简化JDBC）、**SpringMVC**框架（封装Servlet）、**spring**框架等三大框架的基础操作

7、学习**SSM三大框架整合**，使用SSM框架完成永和大王门店系统

关于SSM三大框架，Web阶段只是做一个框架的基础入门，只学习简单的框架使用及整合，框架的高级进阶和底层实现会在第三个阶段（框架阶段）进行学习！！！



## 学习方法建议

1、课上以听为主，注重**知识点的理解**和**程序的实现思路**，养成**多思考**的习惯

2、遇到不懂的做标记，遇到问题不要钻牛角尖，**坚持把课听完整**，课下再解决问题

3、遇到问题或BUG，**先自己分析问题**出现的原因或查看BUG描述（自己找5~10min）

~求助周围的同学

~求助项目经理

~求助讲师

4、**多敲代码！！！**（误区：听懂了就以为自己会了）

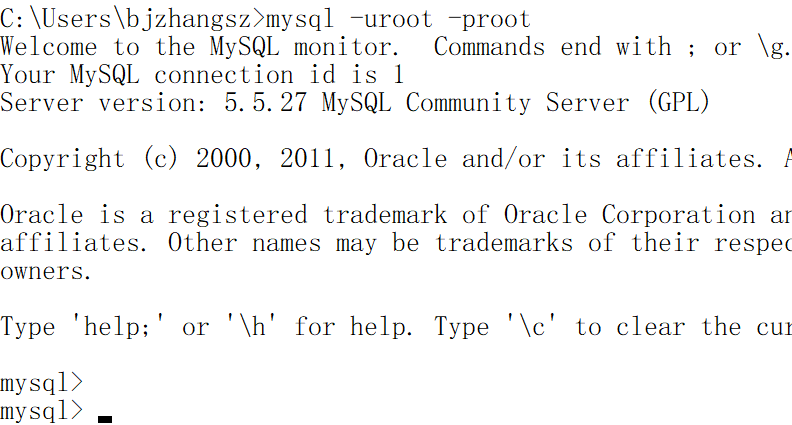
代码量和薪资成正比!!!

# MySQL环境检测

## 检查MySQL环境是否正常

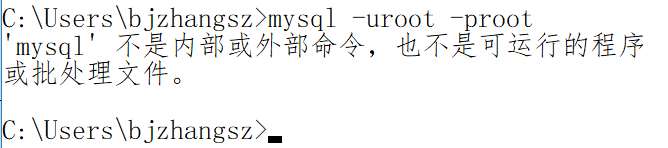
(1)打开CMD窗口，在cmd中输入：mysql -uroot -proot

若提示如下效果：



则说明MySQL环境正常，即当前电脑上安装了MySQL或mariadb软件，并且也配置了！！

(2)如果提示“mysql不是内部或外部命令”则说明MySQL环境有问题，



有如下两种可能：

**a）当前电脑安装了MySQL或mariadb，但没有配置环境变量**

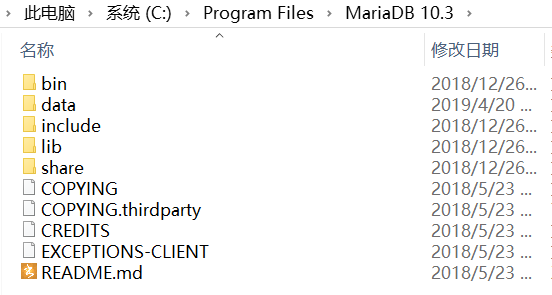
**b）当前电脑既没有安装MySQL或mariadb，也没有配置环境变量**

下面先来验证第一种情况

## 检查电脑上是否安装MySQL或mariadb

默认情况下，mysql或mariadb会安装在：C:\Program Files\ 目录下，

在C:\Program Files\目录下查看是否有MySQL或mariadb目录，如果有，并且目录结构如下，则说明当前电脑上安装了数据库软件。



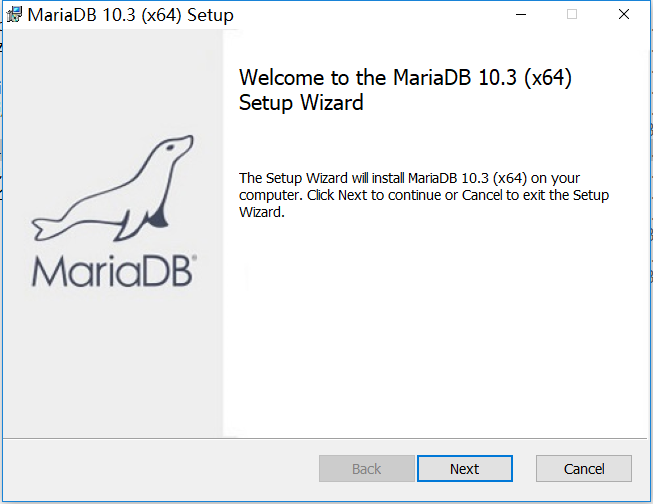
如果当前电脑上安装了数据库软件，则跳转第三步，直接执行第4步，配置mysql或mariadb的环境变量；

如果当前电脑上没有安装数据库软件，则执行第三步，先安装数据库软件，再执行第4步；

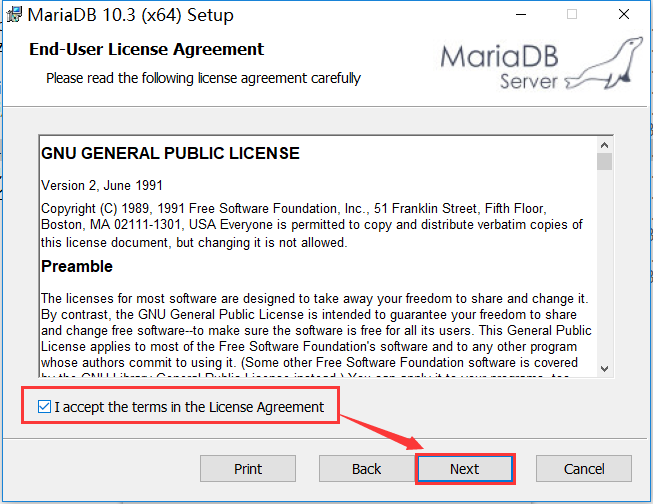
## 安装MySQL或mariadb数据库

在 ftp://cgb1xxx/02-web/software/ 已经上传了 mariadb 软件，大家可以下载下来进行安装，安装步骤如下。

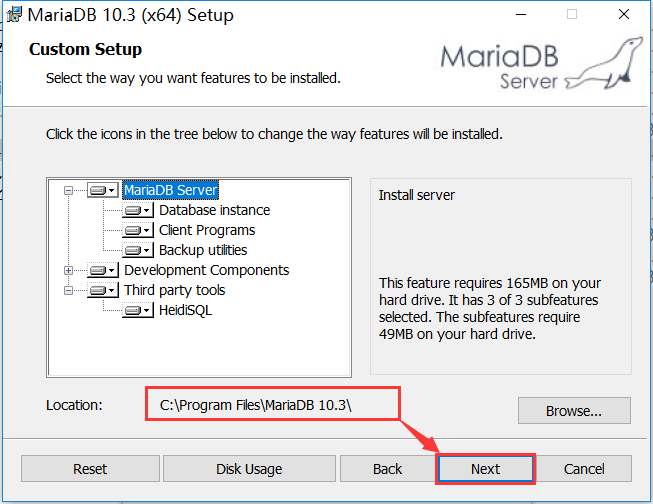
**(1)在下面窗口中直接点击下一步即可**



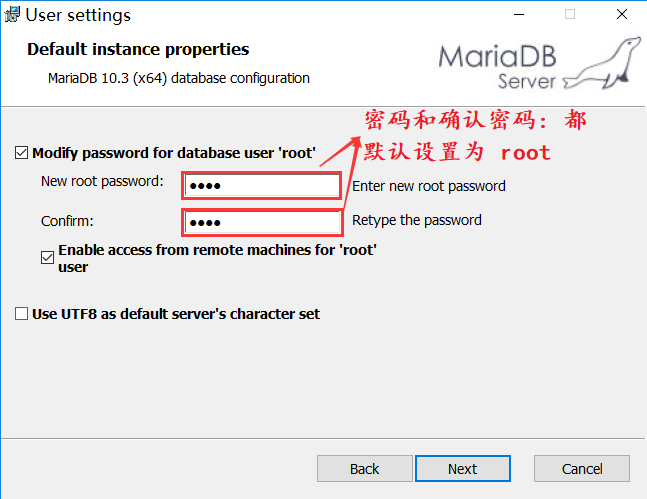
**(2)在下面窗口中勾选复选框，同意协议**



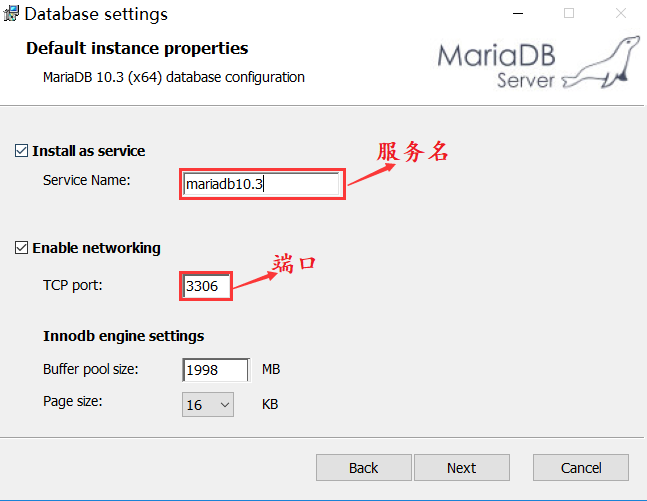
**(3)下面窗口要求设置软件安装路径，保持默认即可**



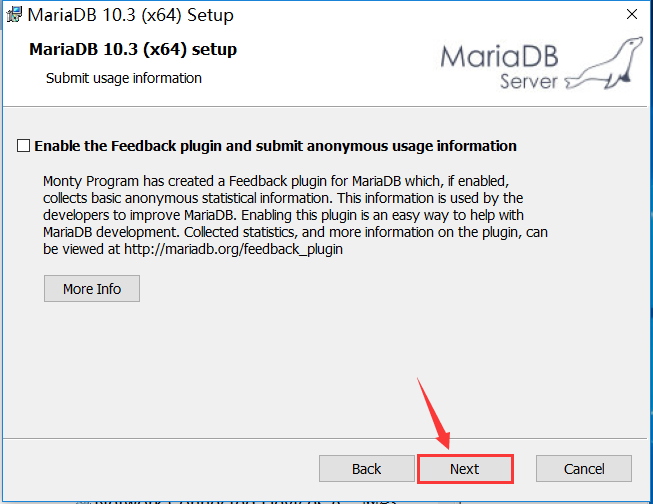
**(4)在下面窗口中设置密码和确认密码，可以统一设置为 root**



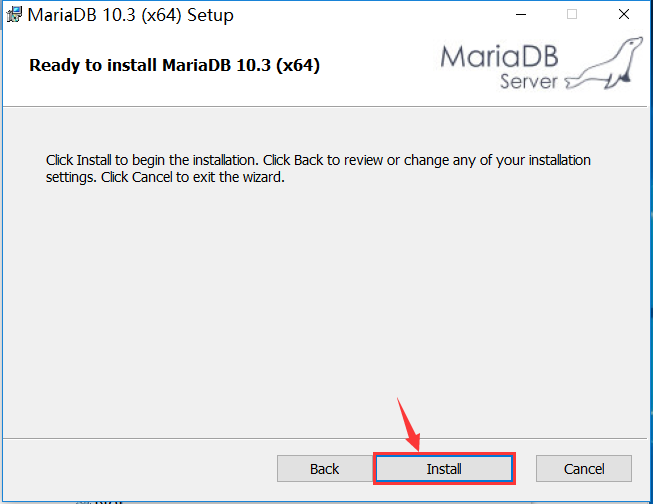
**(5)在下面窗口中设置服务的名称**（只要不和系统中存在的服务名称冲突即可） **和 端口**（只要不和系统中其他进程占用的端口冲突即可）



**(6)在下面窗口中直接点击下一步**

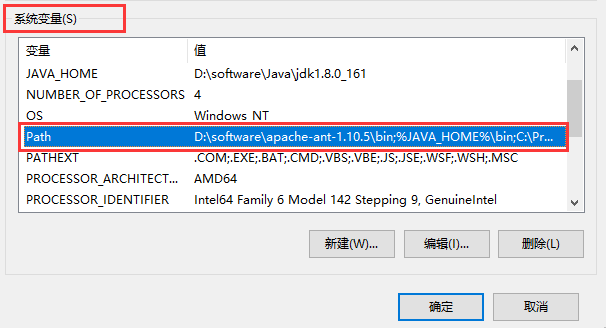


**(7)点击instal安装即可**



## 配置MySQL或mariadb环境变量

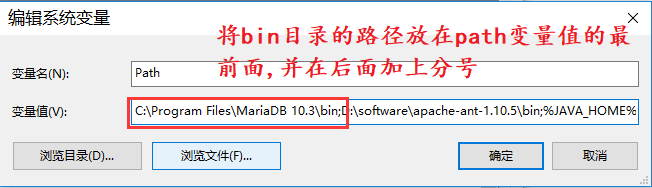
(1)在 **计算机/此电脑** 图标上**右键属性** ----> **高级系统设置** ----> **环境变量**，在弹出的窗口中找到 **系统变量**，在系统变量窗口中找到 **Path变量**，如图所示：



**(2)找到mysql或mariadb安装目录下的bin目录，复制bin目录的路径，例如：**



**(3)将复制的路径，配置到path环境变量中。如果是win7系统，配置如下：**



**(4)如果是win8或win10系统：**

