数据库课程设计

系统详细设计说明书

目录

(一)　系统功能概述

本数据库系统是一个类似于 12306 的网络购票系统，其中查询列车表是对非用户开放的。用户注册成为会员后，可以进行查询火车车次，查询票务信息，购买车票，生成电子票单，查询已购车票，退票等功能，当遇到问题时，还可以根据系统提示咨询客服寻求帮助。持有此系统的管理员可以对该系统进行操作。在满足用户需求的同时，我们的系统也为一些商家提供广告服务，支付一定的费用可以在平台投放广告，秉持着以方便网络购票为主，拉动商业经济为辅的设计理念。

(二)　系统功能模块结构

系统功能模块结构主要包括（前端功能、WEB 服务端、数据库端），
其中前两者主要依托 JSP 文件实现，后者使用 MYSQL 脚本语言

1. 前端功能

前端使用 html 与 CSS 样式混合搭建。
在前端，主要实现了与用户的交互，可以展开为以下几项:

- 显示对用户下一步操作的提示
  简单的 html 文本显示功能，在每个界面上设置提示语句与跳转链接，方便用户选择下一步操作。
- 获取用户的输入内容
  该部分通过<form>对表单<table>进行提交，提交至与数据库进行交互的 JSP 界面，编写查询条件。使用表单<table>，将<input>文本输入框嵌入，用户直接对页面的文本框输入，对用户输入的内容进行保存，通过<form>统一提交。
- 将后端与数据库的查询结果可视化
  WEB 服务得到查询结果或反馈，我们使用<% %>标志，可以将 html 与 java 语句嵌入同一个 JSP 文件中，可视化时，使用的是 html 语言。

本模块主要面向用户，由用户操作，功能包括：用户注册信息，修改个人密码，查询个人信息，查询车次信息，订购车票和退车票以及查看通知等。
⑴注册信息：主要是用户在使用此系统之前向系统数据库中注册个人信息，
便于系统以后的管理和保障系统的安全。
⑵修改个人密码：主要是为了保障用户信息安全，用户可以对自己密码进行
替换和重新设置。
⑶查询个人信息：主要是用户对自己的信息查询。
⑷查询车次信息：主要是用户根据自己所想要订购的车票，查询其相
应的火车及其线路的相关信息。
⑸订购车票：用户订购自己所需要的车票。

2. WEB 服务端

WEB 服务端可视为前端与数据库的一个接口。
Java 语言中连接数据库采用的是 JDBC（Java Data Base Connectivity）技术，JDBC 提供了连接各种数据库的能力。在连接 jdbc 中可以连接多种数据库，例如 MYSQL，Oracle,SQLServer,DB2 等，本系统选用的是 MYSQL。
JSP 代码结构:
创建一个以 JDBC 连接数据库的程序，包含 7 个步骤:

- 加载 JDBC 驱动程序:

成功加载后，会将 Driver 类的实例注册到 DriverManager 类中。

- 提供 JDBC 连接的 URL

连接 URL 书写形式: 协议: 子协议: 数据源标识。协议: 在 JDBC 中总是以 jdbc 开始 。子协议: 是桥连接的驱动程序或是数据库管理系统名称。数据源标识: 标记找到数据库来源的地址与连接端口。

- 创建数据库的连接

使用 DriverManager 的 getConnectin(String url , String username , String password)方法传入指定的欲连接的数据库的路径、数据库的用户名和密码来获得。

- 创建一个 Statement

要执行 SQL 语句，必须获得 java.sql.Statement 实例
1，执行静态 SQL 语句。通常通过 Statement 实例实现。2，执行数据库存储过程。通常通过 CallableStatement 实例实现。

- 执行 SQL 语句

Statement 接口提供了三种执行 SQL 语句的方法：executeQuery 、executeUpdate 和 execute。ResultSet executeQuery(String sqlString)：执行查询数据库的 SQL 语句，返回一个结果集（ResultSet）对象。

- 处理结果

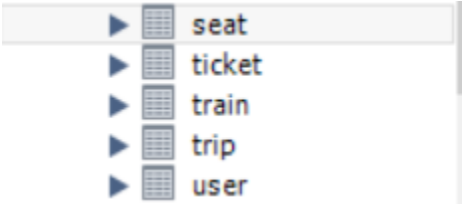
ResultSet 包含符合 SQL 语句中条件的所有行，并且它通过一套 get 方法提供了对这些。使用结果集（ResultSet）对象的访问方法获取数据:

- 关闭 JDBC 对象

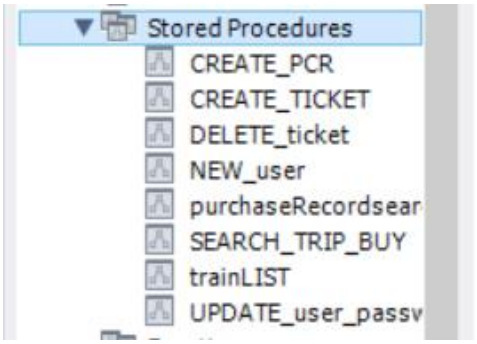
①关闭记录集②关闭声明③关闭连接对象

3. 数据库端

数据库主要负责存储数据，当 WEB 服务层发来查询请求时，向 WEB 层提供查询结果。使用 MYSQL 脚本语言，在建库，建立触发器，存储过程等时，使用了 MYSQL workbench 进行操作或输入命令，可以进行可视化，十分便捷。本数据库一共包含 5 个表:



存储过程：



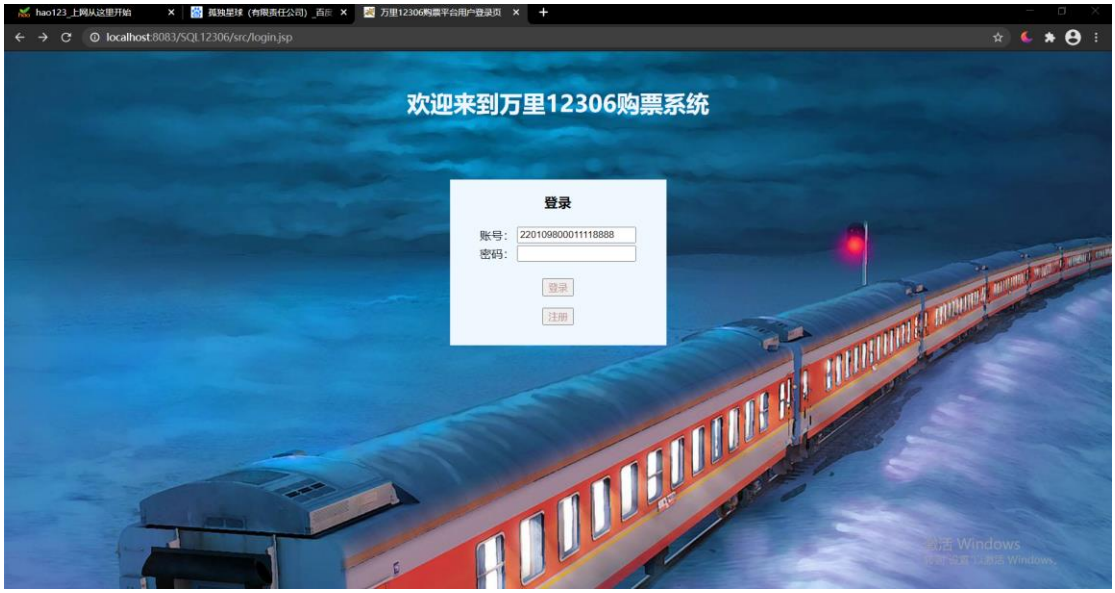
后两个部分主要面向三种类型的管理员，对数据库进行增删改查，对 train 与 trip 表进行信息修正与增加，对异常的购买信息与交易记录进行处理，辅助用户找回密码。

(三)　系统界面设计

为了方便用户区分个界面的功能，在页面显示文字提示的同时，不同子功能界面的背景也设置为不同的。

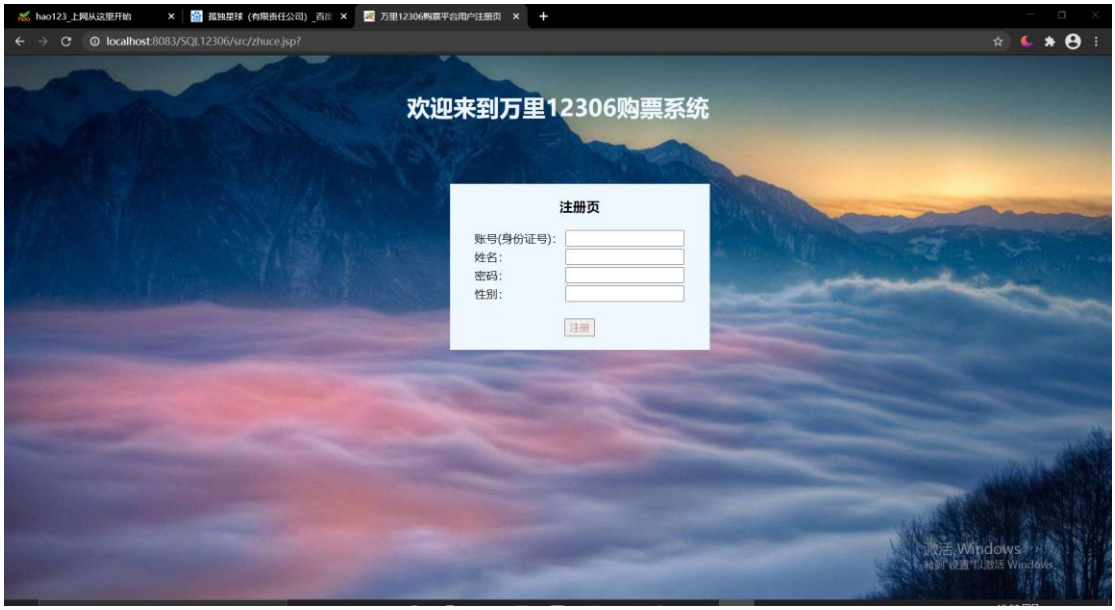
1. 登陆界面

登陆界面主要设计了两个输出窗口，用户输入完成后，点击登录验证正确后即可进入系统。



2. 注册界面

注册界面需要用户填写自己的全部信息（对应 user 表中的四项）。



3. 操作成功界面提示

当注册成功时，进行提示，并可以跳转进行登录。

4. 操作失败界面提示

当注册或登录失败时，会有提示页面出现：

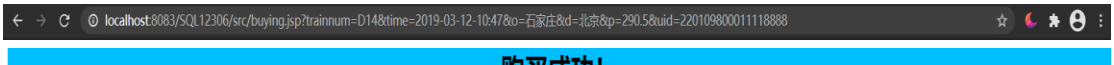
5. 导航页

用户登陆成功后，导航页显示全部的功能，在下方输入要查询的车次信息。

6. 购票页

该页会显示全部的查询结果，同时，显示购票链接。

7. 购买后打印电子车票页



购买成功!

为您打印车票信息，请留存:

车票编号：D14C6CY3T2PD
车厢：2
座位：98
姓名：熊二
身份证号：220109800011118888
价格：290.5
出发时间：2019-03-12-10:47

8. 退票页与结果提示页

9. 查询购买记录验证与结果页

(四) 系统物理模型

1. 表

汇总：
User(id,name,password,gender);
Train(trainnum,origin,destination,maxmem,normalprice)
Trip(trainnum,time,nowmem,nowprice)
Ticket(ticketnum,seatnum,trainnum,id)
Seat(seatnum,trainnum,time,sit)

- 用户

用户	**User**

数据库用户		Root			
主键		ID			
外键					
排序字段		ID			
索引字段		ID			
字段名称	**数据类型**	**允许为空**	**唯一**	**默认值**	**约束条件**
ID	Varchar	N	Y		主键
Name	varchar	N	N		
Password	varchar	N	N		
Gender	varchar	N	N		

- 列车

列车		**Train**			
数据库用户		root			
主键		trainnum			
外键					
排序字段		trainnum			
索引字段		Trainnum,origin,destination			
字段名称	**数据类型**	**允许为空**	**唯一**	**默认值**	**约束条件**
trainnum	Varchar	N	Y		主键
origin	varchar	N	N		
destination	varchar	N	N		
maxmem	int	N	N		

normalprice	float	N	N		

- 车次

车次		**Trip**			
数据库用户		root			
主键		Trainnum,time			
外键					
排序字段		Trainnum,time			
索引字段		Trainnum,time			
字段名称	**数据类型**	**允许为空**	**唯一**	**默认值**	**约束条件**
trainnum	Varchar	N	N		主键
time	varchar	N	N		
nowmem	int	N	N		
nowprice	float	N	N		

- 车票

车票		**Ticket**			
数据库用户		root			
主键		ticketnum			
外键		Seatnum,id			
排序字段		ticketnum			
索引字段		ticketnum,id			
字段名称	**数据类型**	**允许为空**	**唯一**	**默认值**	**约束条件**

ticketnum	Varchar	N	Y		主键
seatnum	varchar	N	N		外键
trainnum	varchar	N	N		
time	varchar	N	N		
id	varchar	N	N		外键

- 座位

座位		**Seat**			
数据库用户		root			
主键		Seatnum,trainnum,time			
外键					
排序字段		Seatnum,trainnum,time			
索引字段		Seatnum,trainnum,time,sit			
字段名称	**数据类型**	**允许为空**	**唯一**	**默认值**	**约束条件**
Seatnum	Varchar	N	Y		主键
trainnum	varchar	N	N		主键
time	varchar	N	N		主键
Sit	int	N	N		

2. 视图与索引

- 针对于最常见的列车——D 开头的动车建立了视图，对动车的查询进行优化。

```
CREATE
    ALGORITHM = UNDEFINED
    DEFINER = `root`@`localhost`
    SQL SECURITY DEFINER
```

```
VIEW `wanli_12306`.`trian_dongche` AS
    SELECT
        `wanli_12306`.`train`.`TrainNum` AS `TrainNum`,
        ` wanli_12306`.`train`.`Origin` AS `Origin`,
        ` wanli_12306`.`train`.`Destination` AS `Destination`,
        ` wanli_12306`.`train`.`Destination` AS `maxmem `,

    FROM
        ` wanli_12306`.`train`
    WHERE
        (`wanli_12306`.`train`.`TrainNum` LIKE 'D%')

CREATE
    ALGORITHM = UNDEFINED
    DEFINER = `root`@`localhost`
    SQL SECURITY DEFINER
VIEW `wanli_12306`.`trip_dongche` AS
    SELECT
        `wanli_12306`.`trip`.`TrainNum` AS `TrainNum`,
        ` wanli_12306`.`trip`.`Origin` AS `Origin`,
        ` wanli_12306`.`trip`.`Destination` AS `Destination`,
        ` wanli_12306`.`trip`.`time` AS `time`,
    FROM
        ` wanli_12306`.`trip`
    WHERE
        (`wanli_12306`.`trip`.`TrainNum` LIKE 'D%')
```

- 索引

  MYSQL 会为主键自动增加索引，因此我们只需要针对于最经常被查询的属性建立索引即可：(主要是 train 表的目的地与出发地，seat 表的是否有人占用，ticket 表的用户 id)

```
ALTER TABLE `wanli_12306`.`train`
ADD INDEX `origin` USING BTREE (`origin`) VISIBLE;
;
ALTER TABLE `wanli_12306`.`train`
ADD INDEX `destination` USING BTREE (`destination`) VISIBLE;
;
ALTER TABLE `wanli_12306`.`seat`
ADD INDEX `sit` USING BTREE (`sit`) VISIBLE;
;
ALTER TABLE `wanli_12306`.`ticket`
ADD INDEX `id` USING BTREE (`id`) VISIBLE;
```

3. 存储过程

本系统中使用存储过程的理由如下：
①存储过程只在创造时进行编译，以后每次执行存储过程都不需再重新编译，而一般 SQL 语句每执行一次就编译一次所以使用存储过程可提高数据库执行速度。
②当对数据库进行复杂操作时(如对多个表进行 Update,Insert,Query,Delete 时）可将此复杂操作用存储过程封装起来与数据库提供的事务处理结合一起使用。
这些操作，如果用程序来完成，就变成了一条条的 SQL 语句，可能要多次连接数据库。而换成存储，只需要连接一次数据库就可以了。
③存储过程可以重复使用
可减少数据库开发人员的工作量。
④安全性高
可设定只有某此用户才具有对指定存储过程的使用权。
⑤更强的适应性：由于存储过程对数据库的访问是通过存储过程来进行的，因此数据库开发人员可以在不改动存储过程接口的情况下对数据库进行任何改动，
而这些改动不会对应用程序造成影响。
⑥分布式工作：应用程序和数据库的编码工作可以分别独立进行，而不会相互压制。存储过程的编写比基本 SQL 语句复杂。
本实验中，写了如下功能的存储过程：

- 创建车票

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `CREATE_TICKET`(in tic varchar(45),in sn varchar(45) ,in tn varchar(45),in ti varchar(45),in uu varchar(45))
BEGIN
    insert into ticket(ticketnum,seatnum,trainnum,time,id)VALUES(tic,sn,tn,ti,uu);
END
```

- 删除车票

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `DELETE_ticket`(in uid varchar(45),in ticket varchar(45))
BEGIN
    delete from ticket where ticketnum=ticket and uid=id;
END
```

- 创建用户

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `NEW_user`(in uid varchar(45),in uname varchar(45),in upassword varchar(45),in ugender varchar(45))
BEGIN
    insert into user(id,name,password,gender)VALUES(uid,uname,upassword,ugender);
END
```

- 查询车票

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `purchaseRecordsearch`(in sid varchar(45))
BEGIN
    select ticket.ticketnum,ticket.trainnum,ticket.time,ticket.seatnum,trip.nowprice
```

```
        from ticket,trip
        where ticket.trainnum=trip.trainnum and ticket.time=trip.time and ticket.id=sid;
END
```

- 显示可买车次

```
    CREATE DEFINER=`root`@`localhost` PROCEDURE `SEARCH_TRIP_BUY`(in o varchar(45),in d varchar(45))
    BEGIN

    SELECT train.trainnum,train.origin,train.destination,trip.time,trip.nowprice
    FROM train,trip
    where train.origin=o and train.destination=d ;

END
```

- 显示全部车次

```
    CREATE DEFINER=`root`@`localhost` PROCEDURE `trainLIST`()
    BEGIN
        select * from train;
END
```

- 修改密码

```
    CREATE DEFINER=`root`@`localhost` PROCEDURE `UPDATE_user_password`(in uid varchar(45),in upassw varchar(45))
    BEGIN
        update user set password=upassw where uid=id;
END
```

4. 触发器

触发器是一种用来保障参照完整性的特殊的存储过程，它维护不同表中数据间关系的有关规则。当对指定的表进行某种特定操作（如：Insert,Delete 或 Update）时，触发器产生作用。触发器可以调用存储过程。

Ticket 表

- **插入一条车票元组后，车票对应的车次空闲位置-1，同时，车票对应的座位状态变为 1（占用）**

```
CREATE DEFINER=`root`@`localhost` TRIGGER `ticket_AFTER_INSERT` AFTER INSERT ON `ticket` FOR EACH ROW BEGIN
    update seat set sit=1 where seatnum=new.seatnum AND trainnum=new.trainnum AND time=new.time;
    update trip set nowmem=nowmem-1 where trainnum=new.trainnum AND time=new.time;
END
```

- **删除一条车票元组后（退票），车票对应空闲位置数+1,同时，车票对应的座位状态变为 0（无人购买）**

```
CREATE DEFINER=`root`@`localhost` TRIGGER `ticket_BEFORE_DELETE` BEFORE DELETE ON
```

```
`ticket` FOR EACH ROW BEGIN
    update seat set sit=0 where seatnum=old.seatnum AND trainnum=old.trainnum AND time=old.time;
    update trip set nowmem=nowmem+1 where trainnum=old.trainnum AND time=old.time;
END
```

Trip 表

- **生成一条 Trip 元组后，在 seat 表中自动分配对应数目的空座位**

```
CREATE DEFINER=`root`@`localhost` TRIGGER `trip_AFTER_INSERT` AFTER INSERT ON `trip` FOR EACH ROW BEGIN
DECLARE num int (11);
DECLARE base int (11);
DECLARE x int (11);
    set num = (select maxmem from train where trainnum = new.trainnum);
        set x = 1;
            while x <= num do
                insert into seat(seatnum,trainnum,time,sit)VALUES(cast(x as char(45)),new.trainnum,new.time,0);
            set x=x+1;
            end while ;
END
```

- **Trip 元组被删除后，对应的 seat 元组必须删除，防止用户买到不存在的车次**

```
    CREATE DEFINER = CURRENT_USER TRIGGER `wanli_12306`.`trip_BEFORE_DELETE` BEFORE DELETE ON `trip` FOR EACH ROW
BEGIN
    delete from seat where trainnum=old.trainnum and time=old.time;
END
```

User 表

- **用户修改 ID（身份证件信息变更），则在 ticket 里留存的用户信息必须被更新，防止出现无人持有的“幽灵”票**

```
CREATE DEFINER=`root`@`localhost` TRIGGER `user_BEFORE_UPDATE` BEFORE UPDATE ON `user` FOR EACH ROW BEGIN
    update ticket set id=new.id where id=old.id;
END
```

(五)　系统安全体系与设计

1. 用户管理与控制

- 数据库中，我们可以设计不同的角色，授予角色不同的权限，用户使用这些角色的权限。分级地对数据库进行操作与交互，可以增强数据库的安全性，一定程度防止被人恶意破

坏。

分析整个模型，车票系统需要四种角色：

① 普通用户：他们有 user 表的 INSERT 的权限，和 train 表 SELECT 权限，即他们可以将自己的信息 INSERT 操作写入 user 表中，也可以查看车次和车站的全部信息，无法进行其他操作。

② 普通用户管理员：他们拥有对所有表的 SELECT 操作和对 User 表的 UPDATE，操作，用于完成日常用户出票退票，修改密码，修改查询个人信息的请求。

③车次信息管理员：拥有对 seat 表，ticket 表的全部权限

④数据库维护人员：是系统的超级用户，主要负责对数据库的修改维护 ，拥有所有权限。

```
CREATE ROLE Normal_user;
CREATE ROLE User_admini;
CREATE ROLE Train_admini;
GRANT INSERT ON TABLE user To Normal_user;
GRANT SELECT ON TABLE train To Normal_user;
GRANT SELECT,UPDATE,INSERT ON TABLE user To User_admini;
GRANT SELECT ON TABLE ticket To User_admini;
GRANT SELECT ON TABLE trip To User_admini;
GRANT SELECT ON TABLE train To User_admini;
GRANT SELECT ON TABLE seat To User_admini;
GRANT SELECT,INSERT, UPDATE,delete ON TABLE ticket To Train_admini;
GRANT SELECT,INSERT, UPDATE,delete ON TABLE train To Train_admini;
```

- 有时候，用户使用功能需要更高级的权限，这时，我们可以让用户的使用某功能请求先提交至服务器，依据功能的种类，为用户分配一个具有该权限的管理员（一个管理员可同时服务多个用户，且不需要是人工管理），进行相关操作，最终将操作的结果通过接口，回到前端反馈给用户。

2. 存储与恢复

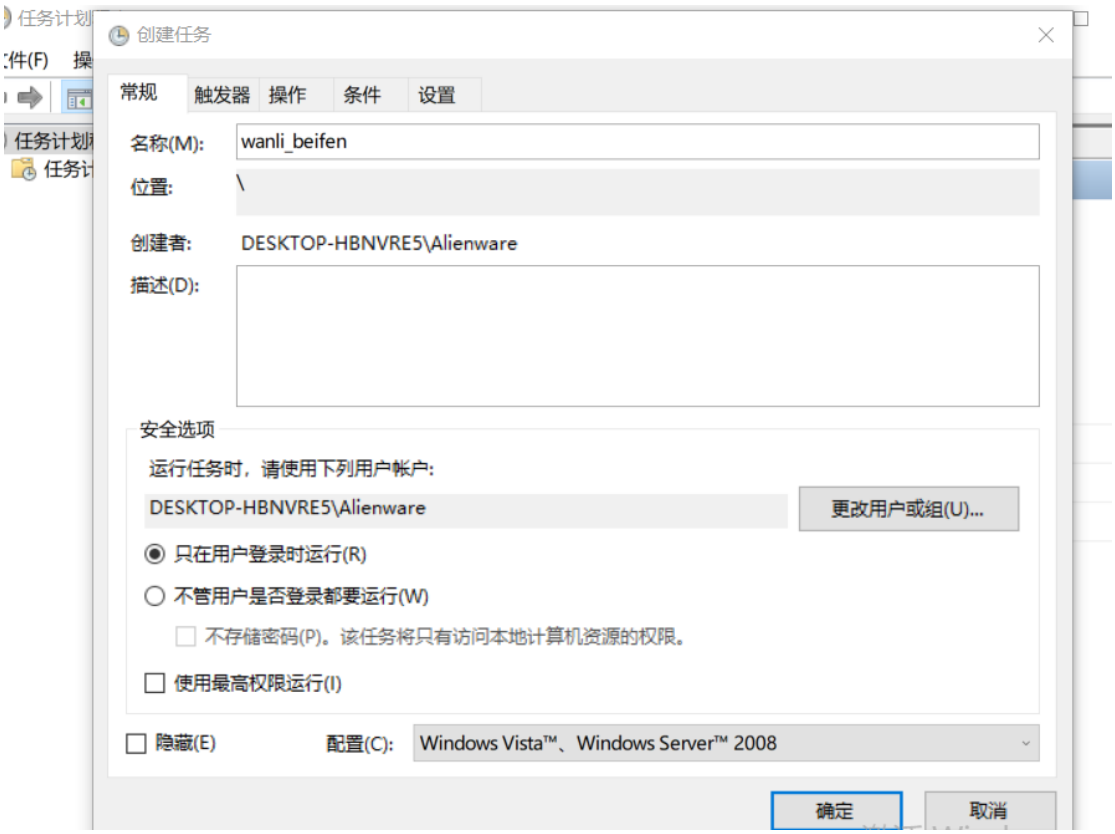
- 备份存储

我们利用导出整个库的方式，可以存储数据库的结构与全部的数据内容。为了保障数据库的安全性，以及面对突如其来的破坏最大程度保护数据的保险措施，我们可以采用编写脚本定时执行的方法，定期对数据库进行备份。

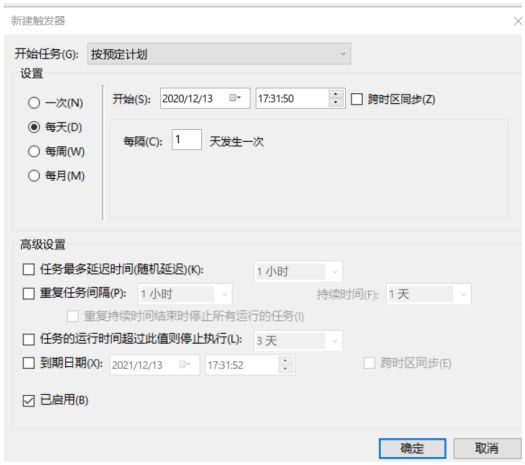
备份脚本 beifen.bat 如下：

```
@echo off
set "yMd=%date:~,4%%date:~5,2%%date:~8,2%"
set "hms=%time:~,2%%time:~3,2%%time:~6,2%"
"C:/Program Files /MySQL/MySQL Server 8.0/bin/mysqldump.exe" -uroot –p ssy0926 wanli_12306>D:/mysql_wanli12306 _%yMd%-%hms%.sql
@echo on
```

使用 Window 的任务计划


创建任务
常规
触发器
操作
条件
设置
名称(M):
wanli_beifen
位置:
\
创建者:
DESKTOP-HBNVRE5\Alienware
描述(D):
安全选项
运行任务时，请使用下列用户帐户:
DESKTOP-HBNVRE5\Alienware
更改用户或组(U)...
只在用户登录时运行(R)
不管用户是否登录都要运行(W)
不存储密码(P)。该任务将只有访问本地计算机资源的权限。
使用最高权限运行(I)
隐藏(E)
配置(C):
Windows Vista™、Windows Server™ 2008
确定
取消

定时执行导出脚本


新建触发器
开始任务(G):
按预定计划
设置
一次(N)
每天(D)
每周(W)
每月(M)
开始(S):
2020/12/13
17:31:50
跨时区同步(Z)
每隔(C):
1
天发生一次
高级设置
任务最多延迟时间(随机延迟)(K):
1 小时
重复任务间隔(P):
1 小时
持续时间(F):
1 天
重复持续时间结束时停止所有运行的任务(I)
任务的运行时间超过此值则停止执行(L):
3 天
到期日期(X):
2021/12/13
17:31:52
跨时区同步(E)
已启用(B)
确定
取消

(六)　系统运行环境设计与部署结构

1.运行环境

本实验的前端页面与 WEB 服务器需要部署在 tomcat 上
本实验需要的环境有：

- MYSQL 环境
- Tomcat 环境

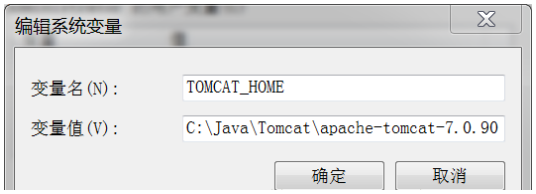
MYSQL 的环境配置之前已经完成了。
本系统需要使用 JSP 来完成了将数据库中存储的数据通过 WEB 页面形式展示出来的整个过程，因此需要配置 tomcat 服务器，并将数据库连接需要的 jar 包导入。
过程可分为如下几个步骤：
1.下载配置 tomcat 服务器。
2.下载 mysql-connector-java-8.0.11-bin.jar 包。
3.IDEA 数据库。新建 web 工程，拷贝.jar 文件到路径下。
4.将系统整个工程文件放入 tomcat 安装文件夹下的 webapps 文件夹下
5.本机输入 http://localhost:8083/SQL12306/src/login.jsp 即可访问页面

- 下载好压缩包后，直接解压至某一目录下，目录中不能包含中文。解压后如图所示：

- 将此文件夹拷贝到选择的目录下

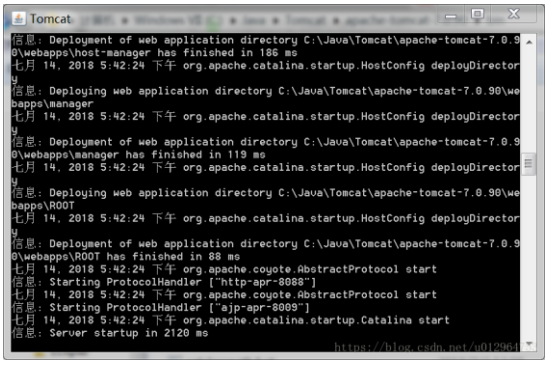
- 开始配置环境变量，打开环境变量同上。

然后新建一个系统变量：TOMCAT_HOME=C:\Java\Tomcat\apache-tomcat-7.0.90（此路径为你解压文件夹所在的绝对路径）。



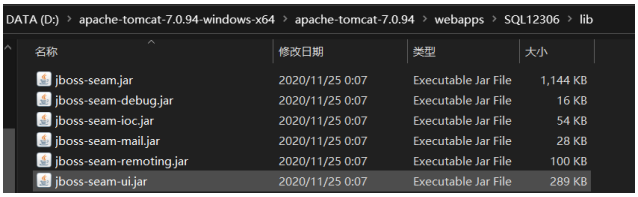
- 在 Classpath 中最后添加%TOMCAT_HOME%\lib\servlet-api.jar;在 Path 中最前添加%TOMCAT_HOME%\bin;

- 至此，配置工作完成。打开 bin 文件夹，双击 startup.bat，它会自动打开如下控制台界面。不要关闭控制台界面，打开浏览器，输入 http://localhost:8083，出现 Tomcat 的网页，证明安装并配置成功！最后关闭控制台，关闭控制台即关闭 Tomcat 服务。



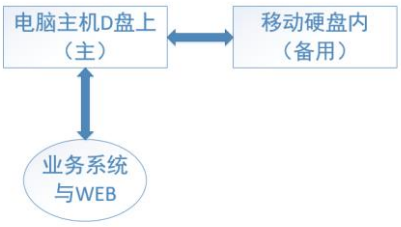
- 每次使用数据库系统前，都需要打开 startup.bat
- 导入 jar 包

将 jar 包放在如下目录/lib：



2. 系统部署结构

由于条件限制，我们采用主备架构，它的优点是，容易实现且很难出现数据不一致的问题，且容以其中主节点数据库在电脑的 D 盘，备用数据库存储在移动硬盘（外设）中，应用系统往数据库主节点写数据，并通过主节点查询。备节点正常情况下只是做备份，只有当主节点被破坏或崩溃了，才会对应用系统提供读服务。



(七)　源代码列表及说明

- Login.jsp
  登录界面
- Logincheck.jsp
  查询登录信息是否正确
- Loginsuccess.jsp
  成功登录后跳转的导航页，可以输入信息查询车次，也可以退票查票等
- Zhuce.jsp
  注册页面
- Zhucecheck.jsp
  查询能否注册（是否曾经注册过）
- zhucefail.jsp
  注册失败解决方案导航
- maichepiaocheck.jsp
  查询车次，并提供购买链接
- buying.jsp
  购买车票并打印电子票单
- searchrecord.jsp
  输入验证信息，查找用户记录
- searchrecordcheck.jsp
  查找并显示购票记录
- Tuipiao.jsp
  退票
- Tuipiaocheck.jsp
  退票操作成功与否反馈页