

**QQ 学习交流群:732021751**

**技术咨询微信/QQ:84985152**

**课程咨询助教 QQ : 484166349**

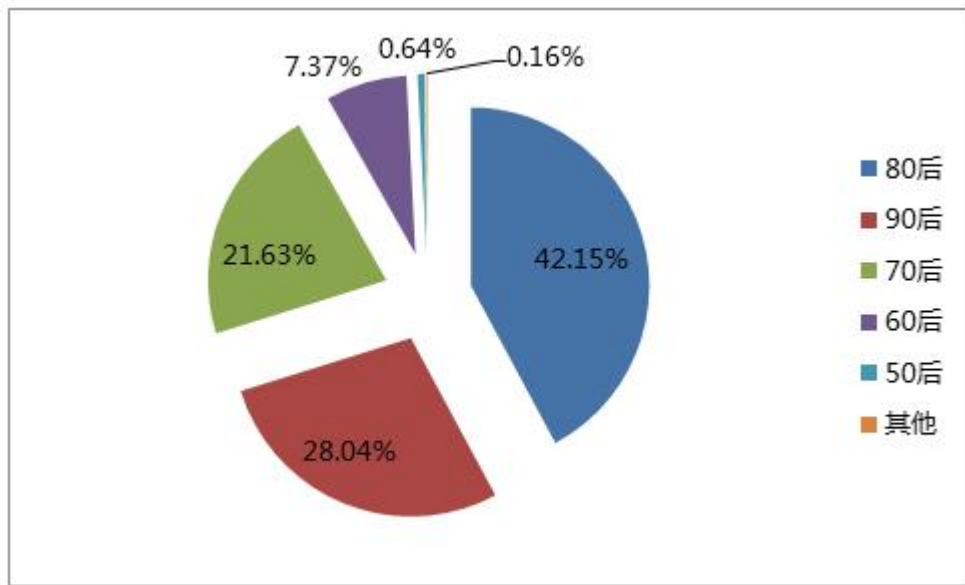
## **（一）大数据项目需求分析与设计**

### **1.项目需求分析**

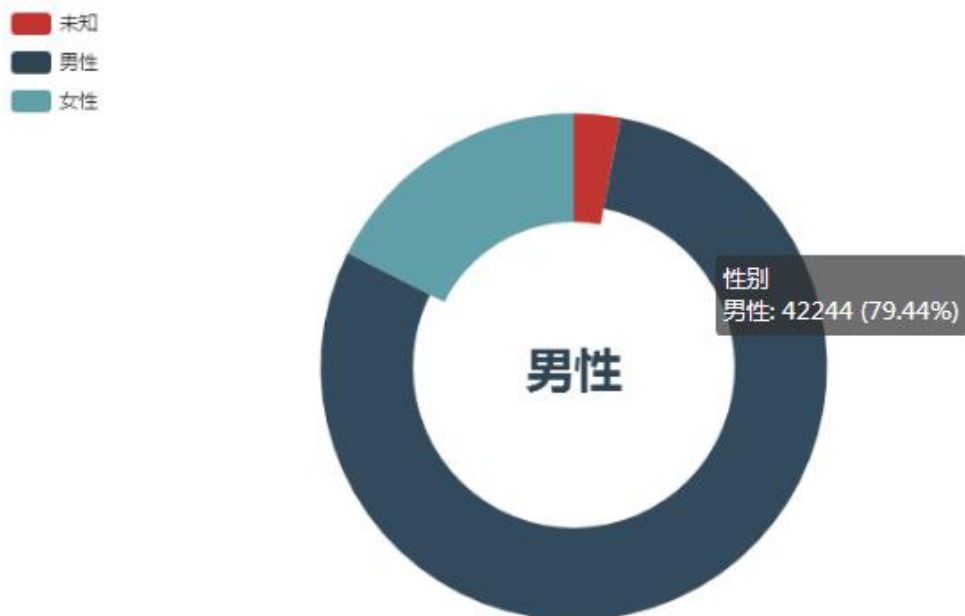
#### **1.1 金融 app 用户信用卡持卡用户特征分析**

1. 80 后、90 后不同年龄段，持有信用卡占比

## 大讲台----课程资料



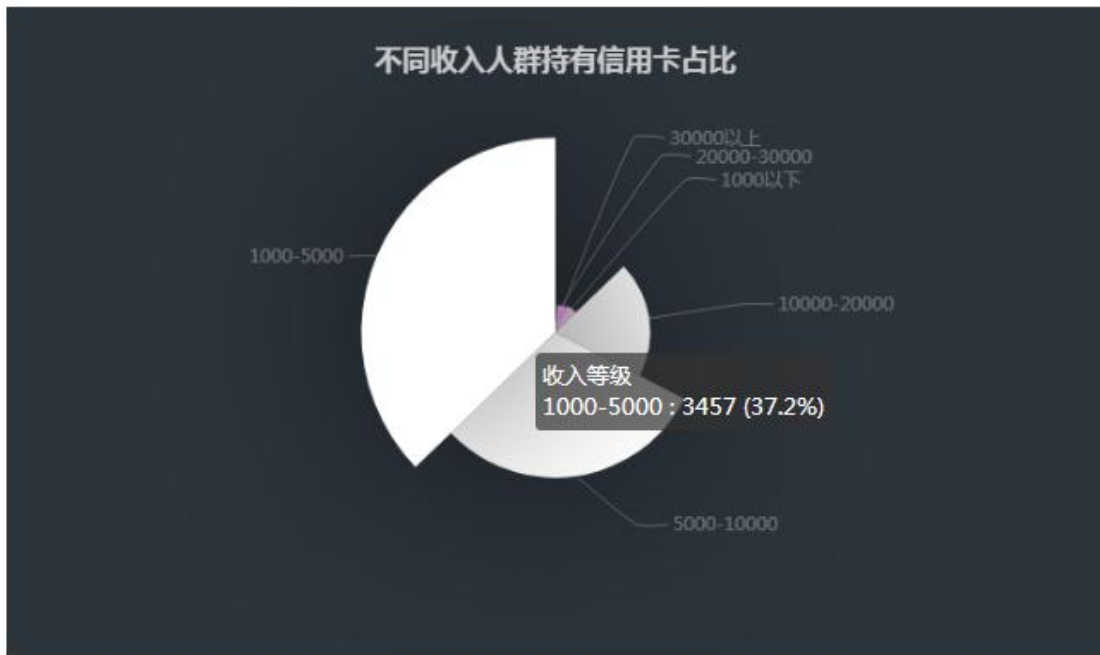
### 2. 男、女不同性别，持有信用卡占比



### 3. 不同省份，持有信用卡用户占比

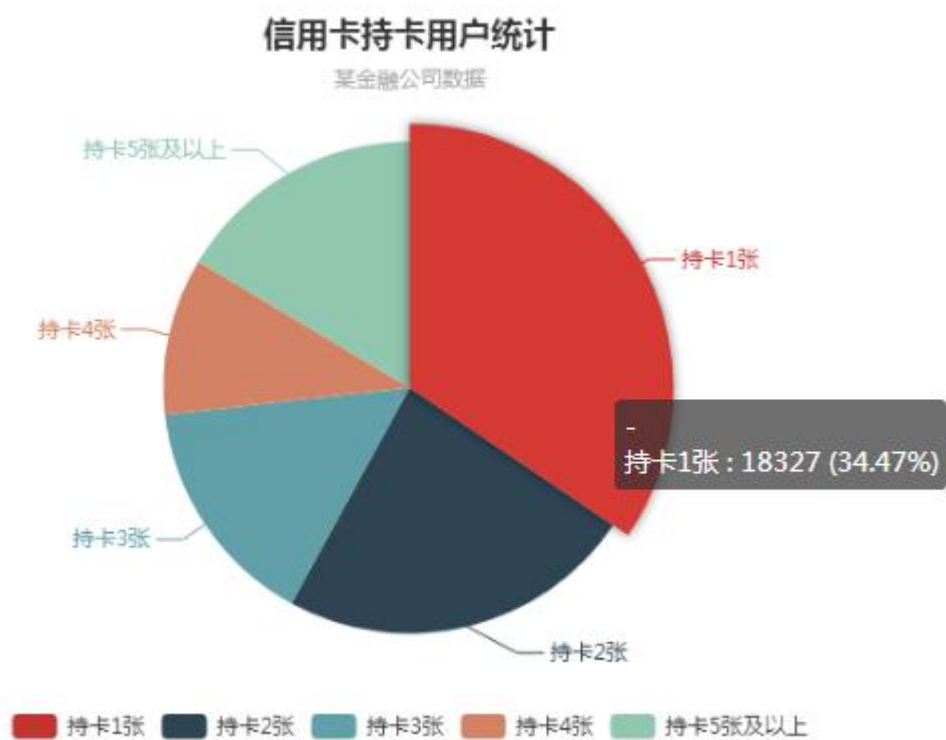


### 4. 不同收入等级，持有信用卡用户占比



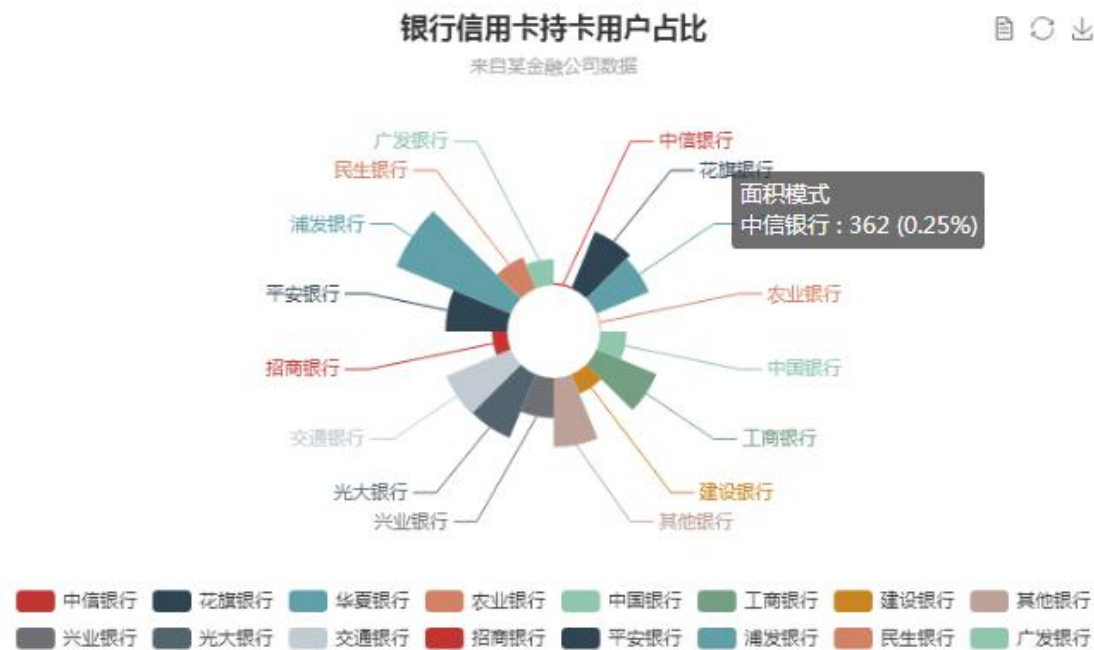
## 1.2 金融 app 信用卡用户消费行为分析

### 1. 持有多张信用卡的用户占比

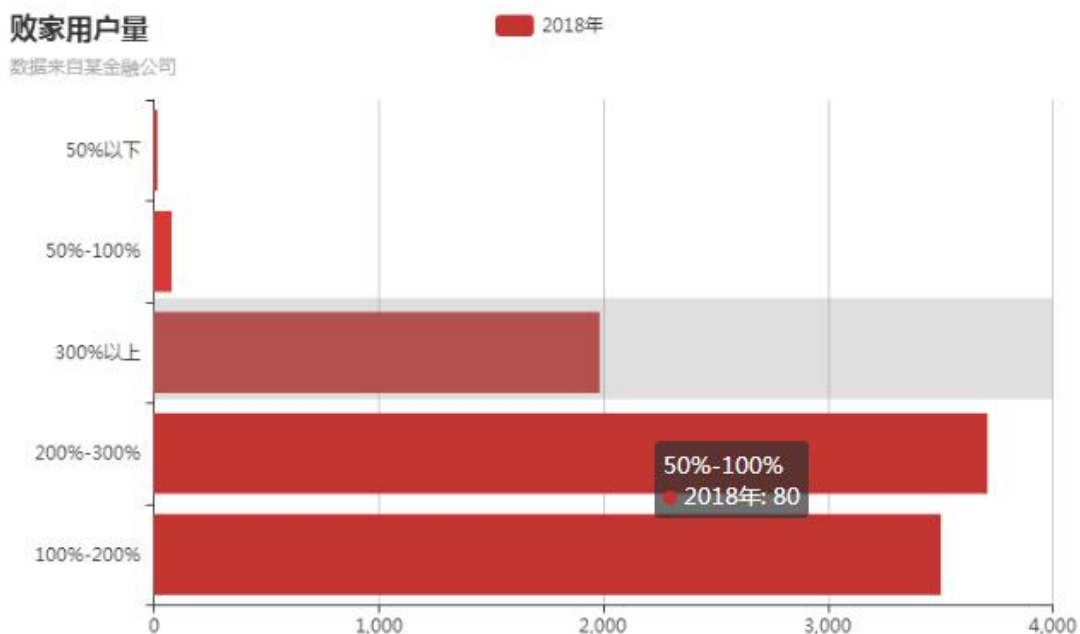


### 2. 不同银行，信用卡持有用户占比

# 大讲台----课程资料

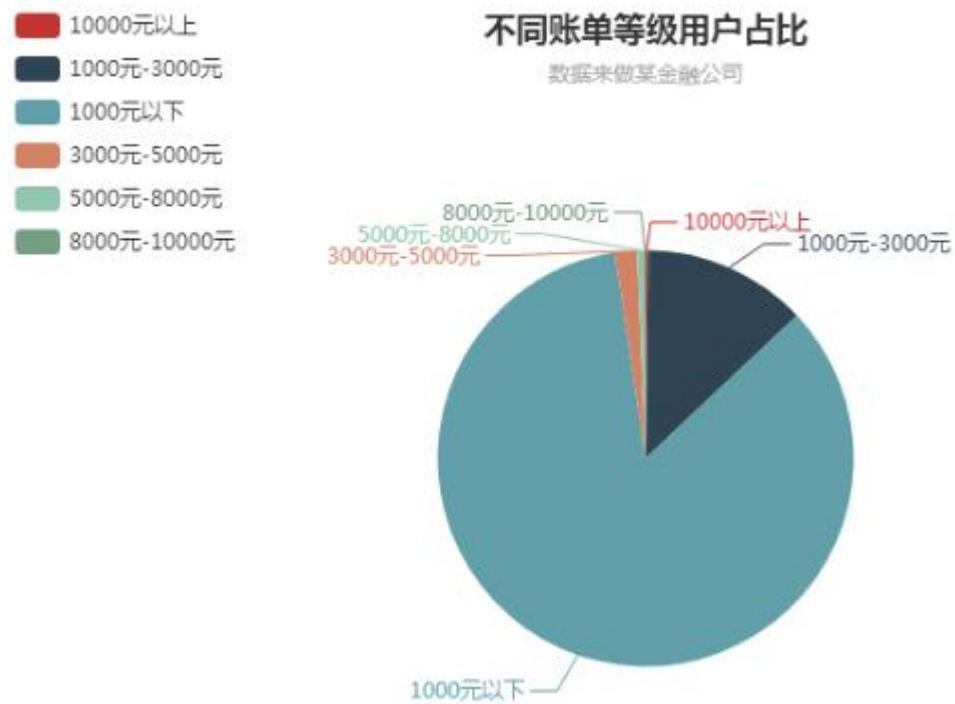


## 3. 败家指数：消费超额、消费透支用户数量



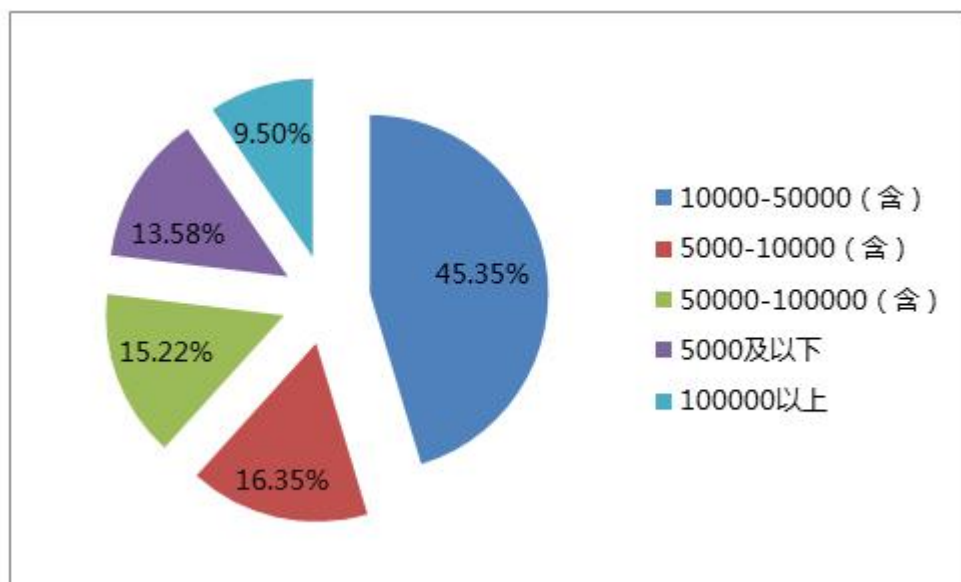
## 4. 不同信用卡账单总金额等级，用户占比

## 大讲台----课程资料



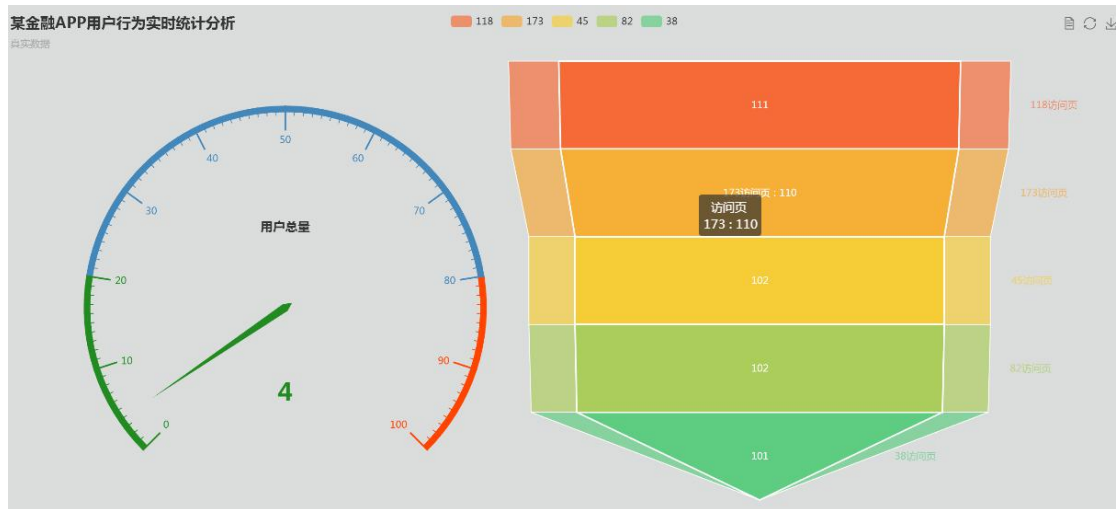
### 1.3 用户信用卡管理行为分析

不同信用卡总额度范围，用户占比

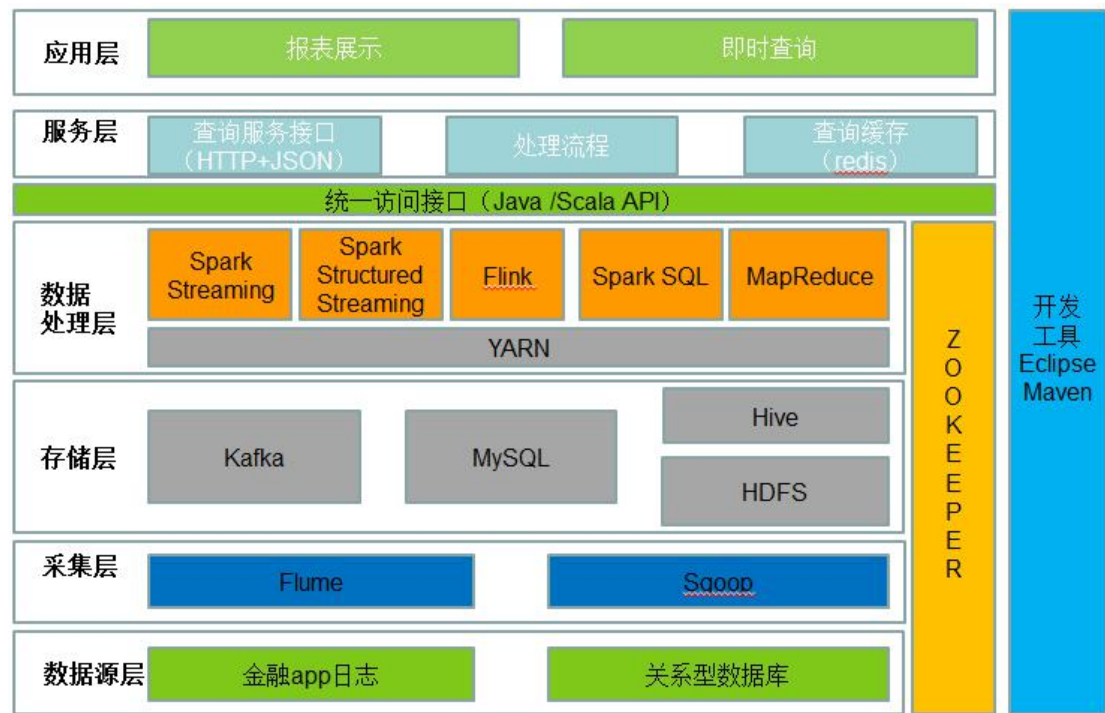


# 大讲台----课程资料

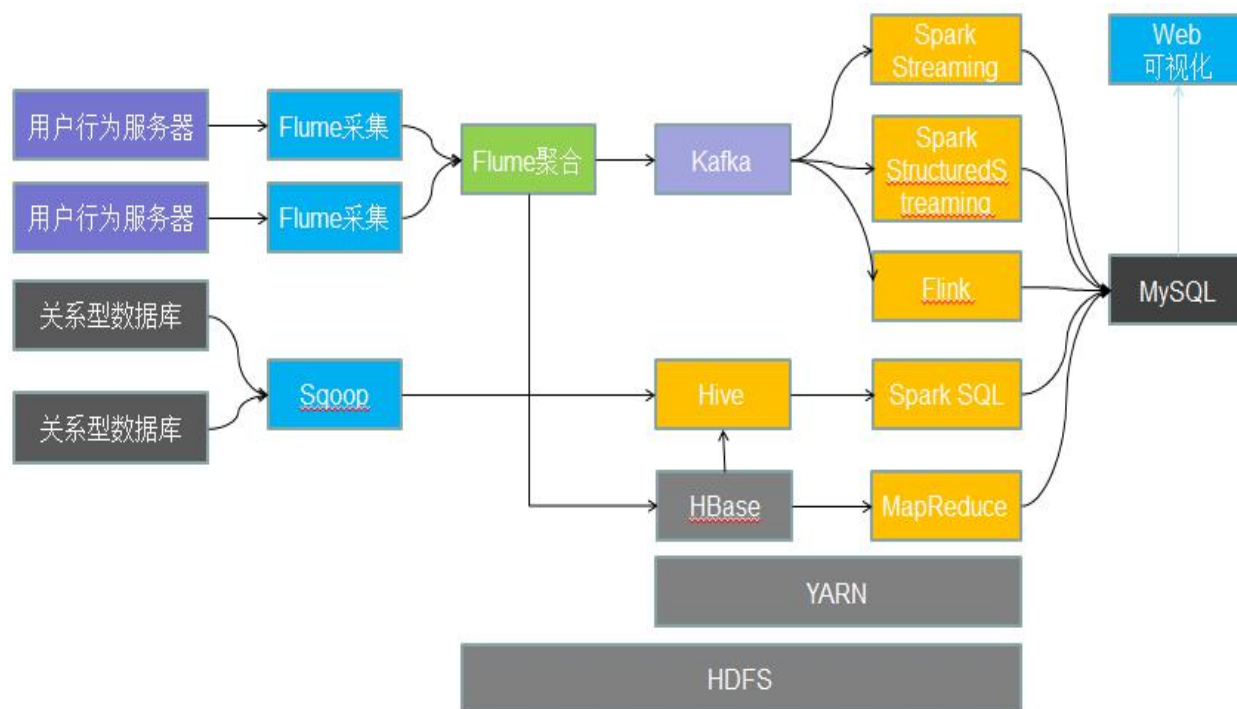
## 1.4 金融 app 页面访问 topn 及 uv



## 2.系统架构设计



## 3.数据流程设计



## 4.大数据平台集群规划

	CDH01	CDH02	CDH03
Zookeeper	QuorumPeerMain	QuorumPeerMain	QuorumPeerMain
HDFS	NN,DN	NN,DN	DN
YARN	RM,NM	RM,NM	NM
Hive	Hive		
HBase	HMaster,RS	HMaster,RS	RS
Flume	Flume 聚合	Flume 采集	Flume 采集
Kafka	Kafka	Kafka	Kafka
Spark	spark		
Mysql	Mysql		
Sqoop	Sqoop		
Flink	Flink		
Kylin	Kylin		

## (二) 大数据项目离线分析

### 1.启动集群相关服务

#### 1.1 启动 Zookeeper 集群

```
runRemoteCmd.sh "/home/hadoop/app/zookeeper/bin/zkServer.sh start" all
```

#### 1.2 启动 HDFS 集群

```
sbin/start-dfs.sh
```

#### 1.3 启动 YARN 集群

```
sbin/start-yarn.sh
```

#### 1.4 启动 hive 元数据库 mysql 服务

查看 mysql 服务状态

```
[root@cdh01 ~]# service mysqld status
```

启动 mysql 服务

```
[root@cdh01 ~]# service mysqld start
```

#### 1.5mysql 授权访问

查看 mysql 数据库 user 用户信息

```
select host ,user,password from mysql.user;
```

创建用户和密码

```
create user 'root' identified by 'root';
```

允许 root 用户在一个特定的 IP 进行远程登录，并具有所有库任何操作权限.

```
grant all on *.* to 'root'@'cdh01' identified by 'root';
```

```
flush privileges;
```

允许 root 用户在任何地方进行远程登录，并具有所有库任何操作权限.

```
grant all on *.* to 'root'@'%' identified by 'root';
```

```
flush privileges;
```



### 1.6window 下配置 mysql 远程访问（如果是 Linux 下，则已关闭防火墙）

防火墙设置

1.打开控制面板-》系统和安全-》检查防火墙状态

2.点击“高级设置”

3.点击入站规则，点击新建规则

4. 选择端口, 下一步

5.选择 TCP ,填 端口号 3306 按下一步

6. 选择允许所有连接，下一步

7 全选., 确定

### 2.MySQL 业务数据导入 Hive 数据仓库

#### 2.1 用户基础表 user\_info 导入 Hive

```
bin/sqoop import \  
--connect  
'jdbc:mysql://192.168.1.101/test?useUnicode=true&characterEncoding=utf-8' \  
--username root \  
--password root \  
--table user_info \  
--fields-terminated-by '\t' \  
-m 3 \  
--hive-import;
```

#### 2.2 银行流水记录表 bank\_detail 导入 Hive

```
bin/sqoop import \  
--connect  
'jdbc:mysql://192.168.1.101/test?useUnicode=true&characterEncoding=utf-8' \  
--username root \  

```

## 大讲台----课程资料

```
--password root \  
--table bank_detail \  
--fields-terminated-by '\t' \  
-m 3 \  
--hive-import;
```

### 2.3 用户浏览行为表 **browse\_history** 导入 Hive

```
bin/sqoop import \  
--connect  
'jdbc:mysql://192.168.1.101/test?useUnicode=true&characterEncoding=utf-8' \  
--username root \  
--password root \  
--table browse_history \  
--fields-terminated-by '\t' \  
-m 3 \  
--hive-import;
```

### 2.4 信用卡账单记录表 **bill\_detail** 导入 Hive

```
bin/sqoop import \  
--connect  
'jdbc:mysql://192.168.1.101/test?useUnicode=true&characterEncoding=utf-8' \  
--username root \  
--password root \  
--table bill_detail \  
--fields-terminated-by '\t' \  
-m 3 \  
--hive-import;
```

### 2.5 放款时间信息表 **loan\_time** 导入 Hive

```
bin/sqoop import \  
--connect  
'jdbc:mysql://192.168.1.101/test?useUnicode=true&characterEncoding=utf-8' \  
--username root \  
--password root \  
--table loan_time \  
--fields-terminated-by '\t' \  
-m 3 \  
--hive-import;
```

## 大讲台----课程资料

### 2.6 顾客逾期还款表 overdue 导入 Hive

```
bin/sqoop import \  
--connect  
'jdbc:mysql://192.168.1.101/test?useUnicode=true&characterEncoding=utf-8' \  
--username root \  
--password root \  
--table overdue \  
--fields-terminated-by '\t' \  
-m 3 \  
--hive-import;
```

### 2.7 开发 Sqoop shell 脚本

1.开发数据库连接公共配置文件

```
vi sq_db.config
```

```
CONNECTURL=jdbc:mysql://192.168.1.101/test?useUnicode=true&characterEncodin  
g=utf-8  
USERNAME=root  
PASSWORD=root  
MAPNUM=3
```

2.开发文件目录路径公共配置文件

```
vi common.sh  
#!/bin/sh  
#切换到当前目录的父目录  
home=$(cd `dirname $0`; cd ../ pwd)  
bin_home=$home/bin  
conf_home=$home/conf  
logs_home=$home/logs  
data_home=$home/data  
lib_home=$home/lib  
#sqoop 的根目录  
sqoop_home=/home/hadoop/app/sqoop
```

3.开发 sqoop shell 脚本

```
vi sq_import_overdue.sh  
#!/bin/sh  
source ~/.bashrc  
source $conf_home/sq_db.config  
source $conf_home/common.config  
#表名称
```

## 大讲台----课程资料

```
v_tableName=overdue
$sqoop_home/bin/sqoop import \
--connect $CONNECTURL \
--username $USERNAME \
--password $PASSWORD \
--table $v_tableName \
--fields-terminated-by '\t' \
-m $MAPNUM \
--hive-import;
```

### 4.测试运行 sqoop shell 脚本

对 sq\_import\_overdue.sh 脚本授予可执行权限：  
chmod u+x sq\_import\_overdue.sh

执行 sq\_import\_overdue.sh 脚本  
bin/sq\_import\_overdue.sh

## 3.Hive 大数据项目离线分析

### 3.1 信用卡持卡用户特征分析

#### 1. 统计 80 后、90 后等不同年龄段的用户，持有信用卡占比

从导入的数据表来看，只有 bill\_detail 表（信用卡账单记录）跟信用卡用户相关

（1）统计出所有持有信用卡用户，生成中间表 middle\_bill\_user

```
bin/hive -e 'create table middle_bill_user as select u.* from (select uid from bill_detail group by uid) tbd inner join user_info u on tbd.uid=u.uid'
```

（2）统计 80 后，90 后等不同年龄段，持有信用卡用户量

```
bin/hive -e ' create table result1
```

```
row format delimited
```

```
fields terminated by ','
```

```
STORED AS TEXTFile
```

```
as select mbu.period,count(mbu.uid) as num from (select u.uid,case when u.birthdate>='1990' and u.birthdate<='1999' then '90 后' when u.birthdate>='1980' and u.birthdate<='1989' then '80 后' when u.birthdate>='1970' and u.birthdate<='1979' then '70 后' when u.birthdate>='1960' and u.birthdate<='1969' then '60 后' when u.birthdate>='1950' and u.birthdate<='1959' then '50 后' else '其他' end as period from middle_bill_user u) mbu group by mbu.period;'
```

（3）可以导出外部 mysql，统计不同年龄段，持有信用卡占比

## 大讲台----课程资料

```
select period,num/tmp.sum_num from result1,(select sum(num) as sum_num from result1) as tmp;
```

### 2.统计男性、女性，信用卡持卡占比

(1) 统计男性、女性，信用卡持卡用户数量

```
bin/hive -e 'create table result2
row format delimited
fields terminated by ','
STORED AS TEXTFile
as select sex,count(uid) as num from middle_bill_user group by sex;'
```

(2) 可以导出外部 mysql，统计男性、女性或者未知性别持有信用卡占比

```
select sex,num/tmp.sum_num from result2,(select sum(num) as sum_num from result2) as tmp;
```

### 3.统计不同省份，信用卡持卡用户占比

(1) 统计不同省份，信用卡持卡用户数量

```
bin/hive -e 'create table result3
row format delimited
fields terminated by ','
STORED AS TEXTFile
as select province,count(uid) as num from middle_bill_user group by province;'
```

(2) 可以导出外部 mysql，统计不同省份信用卡持卡用户占比

```
select province,num/tmp.sum_num from result3,(select sum(num) as sum_num from result3) as tmp;
```

### 4.统计信用卡持卡用户，年收入不同等级占比

(1) 统计信用卡持卡用户，年收入不同等级，用户数量统计

```
bin/hive -e 'create table result4
row format delimited
fields terminated by ','
STORED AS TEXTFile
as select bds.salarylevel,count(bds.uid) as num from
(select bd.uid,
case when bd.amount>=30000 then '30000 以上'
when bd.amount>=20000 and bd.amount<30000 then '20000-30000'
```

## 大讲台----课程资料

```
when bd.amount>=10000 and bd.amount<20000 then '10000-20000'  
when bd.amount>=5000 and bd.amount<10000 then '5000-10000'  
when bd.amount>=1000 and bd.amount<5000 then '1000-5000'  
else '1000 以下' end as salarylevel  
from  
(select b.uid,sum(b.tradeaccount) as amount from bank_detail b LEFT JOIN  
middle_bill_user u on u.uid=b.uid and b.tradetype=0 group by b.uid) bd) bds group  
by bds.salarylevel;
```

(2) 可以导出外部 mysql, 统计不同收入等级占比

```
select salarylevel,num/tmp.sum_num from result4,(select sum(num) as sum_num  
from result4) as tmp;
```

### 3.2 信用卡用户消费行为分析

#### 1. 持有银行信用卡不同张数的用户占比

(1) 统计持有银行信用卡不同张数的用户数量

```
bin/hive -e 'create table result5  
row format delimited  
fields terminated by ','  
STORED AS TEXTFile  
as select bdm.banknum,count(bdm.uid) as cardnum from  
(select bd.uid,  
case when bd.num>=5 then '持卡 5 张及以上'  
when bd.num=4 then '持卡 4 张'  
when bd.num=3 then '持卡 3 张'  
when bd.num=2 then '持卡 2 张'  
else '持卡 1 张' end as banknum  
from (select uid,count(distinct bankid) as num from bill_detail group by uid) bd) bdm  
group by bdm.banknum;'
```

(2) 可以导出外部 mysql, 统计持有信用卡不同张数, 分别占比

```
select banknum,cardnum/tmp.sum_num from result5,(select sum(cardnum) as  
sum_num from result5) as tmp;
```

#### 2. 统计不同银行, 信用卡持卡用户占比

(1) 统计不同银行, 信用卡持卡用户数量

```
bin/hive -e 'create table result6  
row format delimited  
fields terminated by ','  
STORED AS TEXTFile  
as select bd.bankid,count(bd.uid) as usernum from
```

## 大讲台----课程资料

(select distinct uid, bankid from bill\_detail ) bd group by bd.bankid;

(2) 可以导出外部 mysql, 统计不同银行, 信用卡持卡用户占比  
select bankid,usernum/tmp.sum\_num from result6,(select sum(usernum) as sum\_num from result6) as tmp;

### 3.败家指数: 消费金额/收入金额占比

(1) 统计消费金额/收入金额不同占比范围的用户数量

```
bin/hive -e 'create table result7
row format delimited
fields terminated by ','
STORED AS TEXTFile
as select bdt.consumeportion,count(bdt.uid) as num from
(select bdt.uid,
case when bdt.proportion>=3.0 then '300%以上'
when bdt.proportion>=2.0 and bdt.proportion<3.0 then '200%-300%'
when bdt.proportion>=1.0 and bdt.proportion<2.0 then '100%-200%'
when bdt.proportion>=0.5 and bdt.proportion<1.0 then '50%-100%'
else '50%以下' end as consumeportion from
(select bd.uid,max(case when bd.tradetype=1 then bd.amount else 0 end)/max(case
when bd.tradetype=0 then bd.amount else 0 end) as proportion from (select
uid,tradetype,sum(tradeaccount) as amount from bank_detail group by uid,tradetype)
bd group by bd.uid) bdt) bdt group by bdt.consumeportion;'
```

(2) 可以导出外部 mysql, 统计消费金额/收入金额比例不同范围的占比  
select consumeportion,num/tmp.sum\_num from result7,(select sum(num) as sum\_num from result7) as tmp;

### 4. 统计信用卡账单总金额不同范围的用户比例

(1) 统计信用卡账单总金额不同范围的用户数量

```
bin/hive -e 'create table result8
row format delimited
fields terminated by ','
STORED AS TEXTFile
as select bdt.amountlevel,count(bdt.uid) as num
from
(select bd.uid,
case when bd.amount>=10000 then '10000 元以上'
when bd.amount>=8000 and bd.amount<10000 then '8000 元-10000 元'
when bd.amount>=5000 and bd.amount<8000 then '5000 元-8000 元'
when bd.amount>=3000 and bd.amount<5000 then '3000 元-5000 元'
when bd.amount>=1000 and bd.amount<3000 then '1000 元-3000 元'
else '1000 元以下' end as amountlevel
```

## 大讲台----课程资料

```
from (select uid,sum(currentBillAmount) as amount from bill_detail group by uid)
bd) bdt group by bdt.amountlevel;'
```

(2) 可以导出外部 mysql, 统计信用卡账单总金额不同范围的用户占比

```
select amountlevel,num/tmp.sum_num from result8,(select sum(num) as sum_num
from result8) as tmp;
```

### 3.3 用户信用卡管理行为分析

#### 1. 统计用户信用卡总额度不同范围用户占比

(1) 统计用户信用卡总额度不同范围用户数量

```
bin/hive -e 'create table result9
row format delimited
fields terminated by ','
STORED AS TEXTFile
as select bdh.amountlevel,count(bdh.uid) as num
from
(select bdt.uid,
case when bdt.amount>=10000 then '10000 以上'
when bdt.amount>=5000 and bdt.amount<10000 then '5000-10000'
when bdt.amount>=1000 and bdt.amount<5000 then '1000-5000'
else '1000 以下' end as amountlevel
from
(select bd.uid,sum(bd.creditlimit) as amount from bill_detail bd group by uid) bdt)
bdh group by bdh.amountlevel;'
```

(2) 导出到外部 mysql, 统计用户信用卡总额度不同范围用户占比

```
select amountlevel,num/tmp.sum_num from result9,(select sum(num) as sum_num
from result9) as tmp;
```

#### 2. 用户信用卡逾期还款时间占比

(1) 用户信用卡逾期还款不同时间范围, 用户数量

```
bin/hive -e 'create table result10
row format delimited
fields terminated by ','
STORED AS TEXTFile
as select sampleLabel,count(uid) as num from overdue group by sampleLabel;'
```

(2) 导出外部 mysql, 统计用户信用卡逾期还款不同时间范围用户占比

```
select sampleLabel,num/tmp.sum_num from result10,(select sum(num) as sum_num
from result10) as tmp;
```



### 3.4 开发 Hive shell 脚本

#### 1. 开发文件目录路径公共配置文件

```
vi common.sh
#!/bin/sh
#切换到当前目录的父目录
home=$(cd `dirname $0`; cd ../ pwd)
bin_home=$home/bin
conf_home=$home/conf
logs_home=$home/logs
data_home=$home/data
lib_home=$home/lib
#hive 的根目录
hive_home=/home/hadoop/app/hive
```

#### 2.开发 hive shell 脚本

```
vi hive_create_result10.sh
#!/bin/sh
source ~/.bashrc
source $conf_home/common.config
echo "HIVE 数据处理开始..."
$hive_home/bin/hive -e 'create table result10
row format delimited
fields terminated by ','
STORED AS TEXTFile
as select sampleLabel,count(uid) as num from overdue group by sampleLabel;'
```

#### 3.测试运行 hive shell 脚本

给 hive shell 脚本授权:

```
chmod u+x hive_create_result10.sh
```

执行 hive shell 脚本:

```
bin/ hive_create_result10.sh
```

## 4.MySQL 业务数据汇总建模

### 1.创建不同年龄段，所持有信用卡用户占比表

```
DROP TABLE IF EXISTS `result1`;
```

```
CREATE TABLE `result1` (  
  `rid` int(11) NOT NULL AUTO_INCREMENT,  
  `period` varchar(10) NOT NULL,  
  `num` double DEFAULT NULL,  
  PRIMARY KEY (`rid`)  
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

### 2.创建男女，所持有信用卡用户占比表

```
DROP TABLE IF EXISTS `result2`;
```

```
CREATE TABLE `result2` (  
  `rid` int(10) NOT NULL AUTO_INCREMENT,  
  `sex` varchar(3) NOT NULL,  
  `num` double DEFAULT NULL,  
  PRIMARY KEY (`rid`)  
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

### 3.创建不同省份，所持有信用卡用户占比表

```
DROP TABLE IF EXISTS `result3`;
```

```
CREATE TABLE `result3` (  
  `rid` int(11) NOT NULL AUTO_INCREMENT,  
  `province` varchar(10) NOT NULL,  
  `num` double DEFAULT NULL,  
  PRIMARY KEY (`rid`)  
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

### 4.创建年收入不同范围，所持有信用卡用户占比表

```
DROP TABLE IF EXISTS `result4`;
```

```
CREATE TABLE `result4` (  
  `rid` int(11) NOT NULL AUTO_INCREMENT,
```

## 大讲台----课程资料

```
`salarylevel` varchar(10) NOT NULL,  
`num` double DEFAULT NULL,  
PRIMARY KEY (`rid`)  
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

### 5.创建信用卡持卡不同张数，对应用户占比表

```
DROP TABLE IF EXISTS `result5`;
```

```
CREATE TABLE `result5` (  
  `rid` int(11) NOT NULL AUTO_INCREMENT,  
  `banknum` varchar(20) NOT NULL,  
  `cardnum` double DEFAULT NULL,  
  PRIMARY KEY (`rid`)  
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

### 6.创建不同银行，所持信用卡用户占比表

```
DROP TABLE IF EXISTS `result6`;
```

```
CREATE TABLE `result6` (  
  `rid` int(11) NOT NULL AUTO_INCREMENT,  
  `bankid` varchar(10) NOT NULL,  
  `usernum` double DEFAULT NULL,  
  PRIMARY KEY (`rid`)  
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

### 7.创建消费与收入比例范围，所对应用户占比表

```
DROP TABLE IF EXISTS `result7`;
```

```
CREATE TABLE `result7` (  
  `rid` int(11) NOT NULL AUTO_INCREMENT,  
  `consumeportion` varchar(10) NOT NULL,  
  `num` double DEFAULT NULL,  
  PRIMARY KEY (`rid`)  
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

### 8.创建信用卡账单总金额不同范围，所对应用户占比表

```
DROP TABLE IF EXISTS `result8`;
```

## 大讲台----课程资料

```
CREATE TABLE `result8` (  
  `rid` int(11) NOT NULL AUTO_INCREMENT,  
  `amountlevel` varchar(20) NOT NULL,  
  `num` double DEFAULT NULL,  
  PRIMARY KEY (`rid`)  
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

### 9.创建信用卡总额度不同范围，所对应用户占比表

```
DROP TABLE IF EXISTS `result9`;
```

```
CREATE TABLE `result9` (  
  `rid` int(11) NOT NULL AUTO_INCREMENT,  
  `amountlevel` varchar(20) NOT NULL,  
  `num` double DEFAULT NULL,  
  PRIMARY KEY (`rid`)  
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

### 10.创建信用卡逾期不同还款时间范围，所对应用户占比表

```
DROP TABLE IF EXISTS `result10`;
```

```
CREATE TABLE `result10` (  
  `rid` int(11) NOT NULL AUTO_INCREMENT,  
  `sampleLabel` varchar(3) NOT NULL,  
  `num` double DEFAULT NULL,  
  PRIMARY KEY (`rid`)  
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

## 5.Hive 业务汇总数据导出到 MySQL 数据库

### 5.1Sqoop 将 Hive 中的 result1 导入 mysql 数据库

```
bin/sqoop export \  
--connect  
'jdbc:mysql://192.168.1.101/test?useUnicode=true&characterEncoding=utf-8' \  
--username root \  
--password root \  
--table result1 \  
--columns period,num \  
--export-dir '/user/hive/warehouse/result1' \  

```

## 大讲台----课程资料

```
--fields-terminated-by '\t' \  
-m 1;
```

### 5.2 Sqoop 将 Hive 中的 result2 导入 mysql 数据库

```
bin/sqoop export \  
--connect  
'jdbc:mysql://192.168.1.101/test?useUnicode=true&characterEncoding=utf-8' \  
--username root \  
--password root \  
--table result2 \  
--columns sex,num \  
--export-dir '/user/hive/warehouse/result2' \  
--fields-terminated-by ',' \  
-m 1;
```

### 5.3 Sqoop 将 Hive 中的 result3 导入 mysql 数据库

```
bin/sqoop export \  
--connect  
'jdbc:mysql://192.168.1.101/test?useUnicode=true&characterEncoding=utf-8' \  
--username root \  
--password root \  
--table result3 \  
--columns province,num \  
--export-dir '/user/hive/warehouse/result3' \  
--fields-terminated-by ',' \  
-m 1;
```

### 5.4 Sqoop 将 Hive 中的 result4 导入 mysql 数据库

```
bin/sqoop export \  
--connect  
'jdbc:mysql://192.168.1.101/test?useUnicode=true&characterEncoding=utf-8' \  
--username root \  
--password root \  
--table result4 \  
--columns salarylevel,num \  
--export-dir '/user/hive/warehouse/result4' \  
--fields-terminated-by ',' \  
-m 1;
```

## 大讲台----课程资料

### 5.5 Sqoop 将 Hive 中的 result5 导入 mysql 数据库

```
bin/sqoop export \  
--connect  
'jdbc:mysql://192.168.1.101/test?useUnicode=true&characterEncoding=utf-8' \  
--username root \  
--password root \  
--table result5 \  
--columns banknum,cardnum \  
--export-dir '/user/hive/warehouse/result5' \  
--fields-terminated-by ',' \  
-m 1;
```

### 5.6 Sqoop 将 Hive 中的 result6 导入 mysql 数据库

```
bin/sqoop export \  
--connect  
'jdbc:mysql://192.168.1.101/test?useUnicode=true&characterEncoding=utf-8' \  
--username root \  
--password root \  
--table result6 \  
--columns bankid,usernum \  
--export-dir '/user/hive/warehouse/result6' \  
--fields-terminated-by ',' \  
-m 1;
```

### 5.7 Sqoop 将 Hive 中的 result7 导入 mysql 数据库

```
bin/sqoop export \  
--connect  
'jdbc:mysql://192.168.1.101/test?useUnicode=true&characterEncoding=utf-8' \  
--username root \  
--password root \  
--table result7 \  
--columns consumeportion,num \  
--export-dir '/user/hive/warehouse/result7' \  
--fields-terminated-by ',' \  
-m 1;
```

### 5.8 Sqoop 将 Hive 中的 result8 导入 mysql 数据库

```
bin/sqoop export \  
--connect  
'jdbc:mysql://192.168.1.101/test?useUnicode=true&characterEncoding=utf-8' \  
--username root \  
--password root \  
--table result8 \  
--columns amountlevel,num \  
--export-dir '/user/hive/warehouse/result8' \  
--fields-terminated-by ',' \  
-m 1;
```

### 5.9 Sqoop 将 Hive 中的 result9 导入 mysql 数据库

```
bin/sqoop export \  
--connect  
'jdbc:mysql://192.168.1.101/test?useUnicode=true&characterEncoding=utf-8' \  
--username root \  
--password root \  
--table result9 \  
--columns amountlevel,num \  
--export-dir '/user/hive/warehouse/result9' \  
--fields-terminated-by ',' \  
-m 1;
```

### 5.10 Sqoop 将 Hive 中的 result10 导入 mysql 数据库

```
bin/sqoop export \  
--connect  
'jdbc:mysql://192.168.1.101/test?useUnicode=true&characterEncoding=utf-8' \  
--username root \  
--password root \  
--table result10 \  
--columns sampleLabel,num \  
--export-dir '/user/hive/warehouse/result10' \  
--fields-terminated-by ',' \  
-m 1;
```

## (三) 大数据项目实时分析

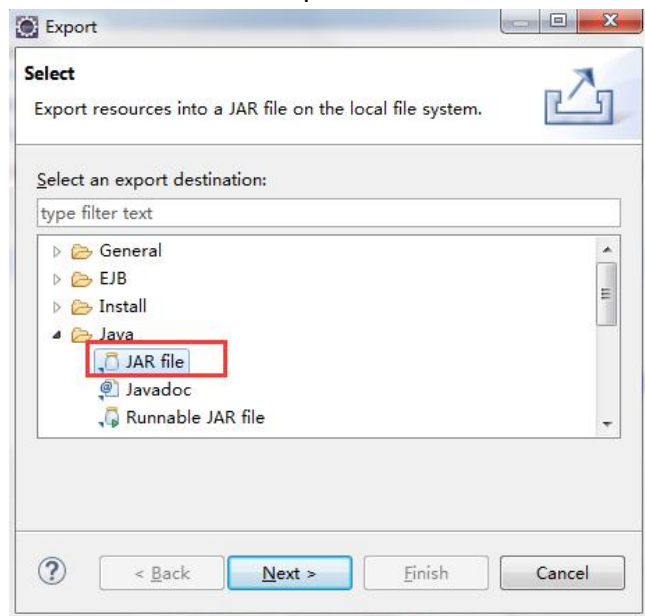
### 1.应用服务器模拟程序开发

#### 1.1 使用 Java 开发应用程序模拟 app 日志数据

```
public static void readData(String inputFile,String outputFile)
{
    FileInputStream fis = null;
    InputStreamReader isr = null;
    BufferedReader br = null;
    String tmp =null;
    try {
        fis = new FileInputStream(inputFile);
        isr = new InputStreamReader(fis, "UTF-8");
        br = new BufferedReader(isr);
        //计数器
        int counter = 1;
        //按行读取数据
        while((tmp=br.readLine())!=null){
            System.out.println("第"+counter+"行: "+tmp);
            //数据写入文件
            writeData(outputFile, tmp);
            counter++;
            //方便观察效果，控制数据产生速度
            Thread.sleep(300);
        }
    }
}
```

#### 1.2Eclipse 对应用程序进行打包

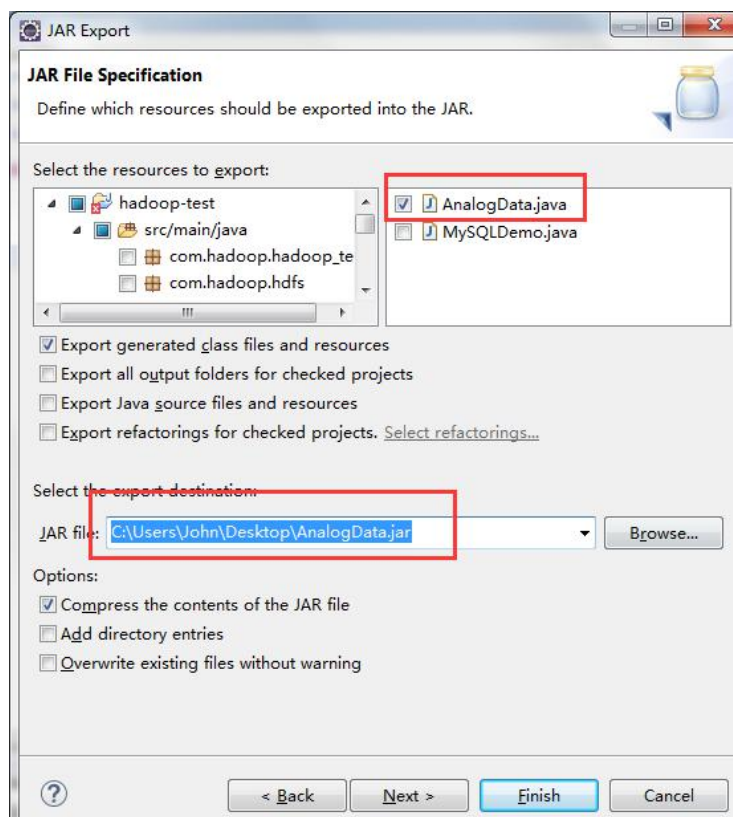
##### 1.右键项目名称—>Export—>JAR file





# 大讲台----课程资料

## 2.JAR Export



## 1.3 分别在 cdh02、cdh03 节点创建脚本目录并上传应用程序包

### 1.创建 shell 目录

```
mkdir shell
```

### 2.创建 shell 子目录

```
[hadoop@cdh02 ~]$ mkdir shell
```

```
[hadoop@cdh02 shell]$ mkdir bin logs conf lib data
```

```
[hadoop@cdh02 lib]$ ls
```

```
AnalogData.jar
```

## 1.4 启动应用程序模拟产生 app 日志数据

### 1.将金融 app 浏览数据 browse\_history.txt 上传至 data 目录

```
[hadoop@cdh02 data]$ ls
```

```
browse_history.txt
```

### 2.编写应用程序 shell 执行脚本

```
vi applog.sh
```

```
#!/bin/bash
```

```
echo "start applog.sh....."
```

## 大讲台----课程资料

#第一个参数是原日志文件，第二个参数是日志生成输出文件

```
java -cp /home/hadoop/shell/lib/AnalogData.jar com.hadoop.test.AnalogData  
/home/hadoop/shell/data/browse_history.txt /home/hadoop/shell/data/applogs.log
```

3.为 shell 脚本添加可执行权限

```
chmod u+x applog.sh
```

4.测试应用程序模拟产生日志（cat /dev/null > applogs.log 清空日志）

查看生产的日志：

```
tail -f applogs.log
```

执行脚本产生日志：

```
./ applog.sh
```

```
[hadoop@cdh02 bin]$ ./applog.sh  
start applog.sh.....  
第1行: 34801,5926003545,173,1  
第2行: 34801,5926003545,164,4  
第3行: 34801,5926003545,38,7  
第4行: 34801,5926003545,45,1  
第5行: 34801,5926003545,110,7  
第6行: 34801,5926003545,118,1  
第7行: 34801,5926003545,50,6  
第8行: 34801,5926003545,139,4  
第9行: 34801,5926003545,82,1  
第10行: 34801,5926003545,101,1  
第11行: 34801,5902423089,173,1  
第12行: 34801,5902423089,82,1
```

## 2.Flume 集群实时采集 app 日志

### 2.1.在 cdh02、cdh03 节点编写 flume 采集日志配置文件

```
vi exec-memory-avro.properties
```

```
agent.sources = execSource
```

```
agent.channels = memoryChannel
```

```
agent.sinks = avroSink
```

```
# For each one of the sources, the type is defined
```

```
agent.sources.execSource.type = exec
```

```
agent.sources.execSource.command = tail -F /home/hadoop/shell/data/applogs.log
```

```
agent.sources.execSource.channels = memoryChannel
```

```
# Each channel's type is defined.
```

```
agent.channels.memoryChannel.type = memory
```

## 大讲台----课程资料

```
agent.channels.memoryChannel.capacity = 100
```

```
# Each sink's type must be defined
agent.sinks.avroSink.type = avro
agent.sinks.avroSink.channel = memoryChannel
agent.sinks.avroSink.hostname = cdh01
agent.sinks.avroSink.port = 1234
```

### 2.2.在 cdh01 节点编写 flume 聚合日志配置文件

```
vi avro-memory-logger.properties
agent.sources = avroSource
agent.channels = memoryChannel
agent.sinks = loggerSink

# For each one of the sources, the type is defined
agent.sources.avroSource.type = avro
agent.sources.avroSource.channels = memoryChannel
agent.sources.avroSource.bind = 0.0.0.0
agent.sources.avroSource.port = 1234

# Each channel's type is defined.
agent.channels.memoryChannel.type = memory
agent.channels.memoryChannel.capacity = 100

# Each sink's type must be defined
agent.sinks.loggerSink.type = logger
agent.sinks.loggerSink.channel = memoryChannel
```

### 2.3.Flume 集群采集日志测试运行

1.在 cdh01 节点启动 flume 聚合服务

```
bin/flume-ng agent -n agent -c conf -f conf/avro-memory-logger.properties
-Dflume.root.logger=INFO,console
```

2.在 cdh02、cdh03 节点启动 flume 采集 applogs 日志数据

```
bin/flume-ng agent -n agent -c conf -f conf/exec-memory-avro.properties
-Dflume.root.logger=INFO,console
```

3.在 cdh02、cdh03 节点启动应用程序脚本模拟产生数据

## 大讲台----课程资料

```
[hadoop@cdh02 bin]$ ./applog.sh
```

### 2.4 flume shell 脚本开发

1.开发启动和关停 flume 数据采集脚本

1) 共用配置脚本

```
vi common.sh
```

```
#!/bin/sh
env=$1
#切换到当前目录的父目录
home=$(cd `dirname $0`; cd ../ pwd)
bin_home=$home/bin
conf_home=$home/conf
logs_home=$home/logs
data_home=$home/data
lib_home=$home/lib
#flume 的根目录
flume_home=/home/hadoop/app/flume
```

2) 启动脚本

```
vi start-flume23-applogs.sh
```

```
#!/bin/sh
```

```
home=$(cd `dirname $0`; cd ../ pwd)
```

```
. ${home}/bin/common.sh
```

```
${flume_home}/bin/flume-ng agent \
--conf ${conf_home} \
-f ${conf_home}/exec-memory-avro.properties -n agent \
>> ${logs_home}/applogs.log 2>&1 &
```

```
echo $! > ${logs_home}/applogs.pid
```

3) 关闭脚本

```
vi stop-flume23-applogs.sh
```

```
#!/bin/sh
```

```
home=$(cd `dirname $0`; cd ../ pwd)
```

```
. ${home}/bin/common.sh
```

```
pid=`cat ${logs_home}/applogs.pid | head -1`
```

## 大讲台----课程资料

```
kill ${pid}
```

### 2.开发启动和关闭 flume 数据聚合脚本

#### 1) 启动脚本

```
vi start-flume1-applogs.sh
```

```
#!/bin/sh
```

```
home=$(cd `dirname $0`; cd ../ pwd)
```

```
. ${home}/bin/common.sh
```

```
${flume_home}/bin/flume-ng agent \  
--conf ${conf_home} \  
-f ${conf_home}/avro-memory-logger.properties -n agent \  
-Dflume.root.logger=INFO,console \  
>> ${logs_home}/applogs.log 2>&1 &
```

```
echo $! > ${logs_home}/applogs.pid
```

#### 2) 关闭脚本

```
vi stop-flume1-applogs.sh
```

```
#!/bin/sh
```

```
home=$(cd `dirname $0`; cd ../ pwd)
```

```
. ${home}/bin/common.sh
```

```
pid=`cat ${logs_home}/applogs.pid | head -1`
```

```
kill ${pid}
```

## 3.Kafka 数据管道

### 3.1 创建 Kafka topic

#### 1.启动 Zookeeper

```
runRemoteCmd.sh "/home/hadoop/app/zookeeper/bin/zkServer.sh start" all
```

#### 2.启动 kafka 集群

```
bin/kafka-server-start.sh config/server.properties
```

#### 3.查看 kafka topic 列表

## 大讲台----课程资料

```
bin/kafka-topics.sh --zookeeper localhost:2181/kafka1.0 --list
```

### 4.创建 kafka topic

```
bin/kafka-topics.sh --zookeeper localhost:2181/kafka1.0 --create --topic applogs  
--replication-factor 3 --partitions 3
```

## 3.2 在 cdh01 节点编写配置文件将 flume 聚合日志发送 Kafka, 并上传至 shell/conf 目录下

```
vi avro-memory-kafka.properties  
agent.sources = avroSource  
agent.channels = memoryChannel  
agent.sinks = kafkaSink  
  
# For each one of the sources, the type is defined  
agent.sources.avroSource.type = avro  
agent.sources.avroSource.channels = memoryChannel  
agent.sources.avroSource.bind = 0.0.0.0  
agent.sources.avroSource.port = 1234  
  
# Each channel's type is defined.  
agent.channels.memoryChannel.type = memory  
agent.channels.memoryChannel.capacity = 100  
agent.channels.memoryChannel.transactionCapacity = 100  
agent.channels.memoryChannel.keep-alive = 5  
  
# Each sink's type must be defined  
agent.sinks.kafkaSink.type = org.apache.flume.sink.kafka.KafkaSink  
agent.sinks.kafkaSink.topic = applogs  
agent.sinks.kafkaSink.brokerList = cdh01:9092,cdh02:9092,cdh03:9092  
agent.sinks.kafkaSink.requiredAcks = 1  
agent.sinks.kafkaSink.kafka.producer.type = sync  
agent.sinks.kafkaSink.batchSize = 1  
agent.sinks.kafkaSink.channel = memoryChannel
```

## 3.3 flume1 节点 shell 脚本开发

### 1.编写 start-flume1-kafka-applogs.sh 脚本

```
vi start-flume1-kafka-applogs.sh  
#!/bin/sh
```

```
home=$(cd `dirname $0`; cd ..; pwd)
```

## 大讲台----课程资料

```
. ${home}/bin/common.sh
```

```
${flume_home}/bin/flume-ng agent \  
--conf ${conf_home} \  
-f ${conf_home}/avro-memory-kafka.properties -n agent \  
-Dflume.root.logger=INFO,console \  
>> ${logs_home}/applogs.log 2>&1 &
```

```
echo $! > ${logs_home}/applogs.pid
```

2.授予 shell 脚本可执行权限

```
chmod u+x start-flume1-kafka-applogs.sh
```

### 3.4 测试运行

1.打开 kafka 消费者消费数据

```
bin/kafka-console-consumer.sh --zookeeper localhost:2181/kafka1.0 --topic applogs
```

2.启动 flume1 聚合节点服务

2.启动 flume1 数据聚合节点服务

```
./start-flume1-kafka-applogs.sh
```

3.启动 flume23 数据采集节点服务

```
bin/start-flume23-applogs.sh
```

4.启动应用程序模拟产生数据

```
bin/applog.sh
```

5.观察是否能正常消费 kafka 数据

## 4.Spark Streaming 大数据项目实时计算

用户浏览金融 app 行为数据格式为：用户 id,时间戳,浏览页面数据,浏览子行为编号

```
34724,5926003545,172,1  
34724,5926003545,163,4  
34724,5926003545,38,7  
67215,5932800403,163,4  
67215,5932800403,138,4  
67215,5932800403,109,7
```

### 4.1 业务表建模

#### 1.业务表 1

业务建表:

```
DROP TABLE IF EXISTS `appCounts`;
```

```
CREATE TABLE `appCounts` (  
  `aid` int(11) NOT NULL AUTO_INCREMENT,  
  `appname` varchar(20) NOT NULL,  
  `count` int(11) NOT NULL,  
  PRIMARY KEY (`aid`)  
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

统计 app 每个页面用户浏览排行:

```
select appname,count from appCounts order by count desc limit 10
```

#### 2.业务表 2

业务建表:

```
DROP TABLE IF EXISTS `userCounts`;
```

```
CREATE TABLE `userCounts` (  
  `aid` int(11) NOT NULL AUTO_INCREMENT,  
  `uid` varchar(20) NOT NULL,  
  PRIMARY KEY (`aid`)  
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

统计访问金融 app uv:

```
select count(1) from userCounts
```

### 4.2 Spark Streaming 业务代码开发

#### 1.通过 maven 引入相关依赖

```
<dependency>  
  <groupId>org.apache.spark</groupId>  
  <artifactId>spark-streaming_2.11</artifactId>  
  <version>2.3.0</version>  
</dependency>  
<dependency>  
  <groupId>mysql</groupId>
```



## 大讲台----课程资料

```
<artifactId>mysql-connector-java</artifactId>
<version>5.1.29</version>
</dependency>
<dependency>
  <groupId>org.apache.spark</groupId>
  <artifactId>spark-streaming-kafka-0-10_2.11</artifactId>
  <version>2.3.0</version>
</dependency>
```

### 2. nc+spark Streaming 集成开发

- 1) 业务代码开发（手动实现）
- 2) 测试运行

在 cdh01 节点启动 nc 服务

nc -lk 9999

本地启动 spark Streaming 应用，通过 nc 输入测试数据，调试整个运行流程

### 3.nc+spark Streaming+mysql 集成开发

- 1) 业务代码开发（手动实现）
- 2) 测试运行

在 cdh01 节点启动 nc 服务

nc -lk 9999

本地启动 spark Streaming 应用，通过 nc 输入测试数据，调试整个运行流程

### 4.Flume+Kafka+Spark Streaming+MySQL 集成开发

- 1) 业务代码开发（手动实现）

官网参考地址：

<http://spark.apache.org/docs/2.3.0/streaming-kafka-0-10-integration.html>

- 2) 测试运行

a) 启动 Spark Streaming 应用程序

b) 启动 flume 聚合脚本

./start-flume1-kafka-applogs.sh

c) 启动 flume 采集脚本

bin/start-flume23-applogs.sh

d) 启动应用程序模拟产生日志脚本

bin/applog.sh

e) 观察数据库结果

## 5. Spark SQL 大数据项目离线计算

### 5.1 业务建模

#### 1. 业务表 1

业务建表:

```
DROP TABLE IF EXISTS `appCounts`;
```

```
CREATE TABLE `appCounts` (  
  `aid` int(11) NOT NULL AUTO_INCREMENT,  
  `appname` varchar(20) NOT NULL,  
  `count` int(11) NOT NULL,  
  PRIMARY KEY (`aid`)  
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

统计 app 每个页面用户浏览排行:

```
select appname,count from appCounts order by count desc limit 10
```

#### 2. 业务表 2

业务建表:

```
DROP TABLE IF EXISTS `userCounts`;
```

```
CREATE TABLE `userCounts` (  
  `aid` int(11) NOT NULL AUTO_INCREMENT,  
  `uid` varchar(20) NOT NULL,  
  PRIMARY KEY (`aid`)  
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

统计访问金融 app uv:

```
select count(1) from userCounts
```

### 5.2 Spark SQL 业务代码开发

#### 1. 通过 maven 引入相关依赖

```
<dependency>  
  <groupId>org.apache.spark</groupId>  
  <artifactId>spark-sql_2.11</artifactId>  
  <version>2.3.0</version>
```

</dependency>

2. Spark SQL 与 mysql 集成开发

## 6. Spark Structured Streaming 大数据项目实时计算

### 6.1 业务建模

1. 业务表 1

业务建表:

```
DROP TABLE IF EXISTS `appCounts`;
```

```
CREATE TABLE `appCounts` (  
  `aid` int(11) NOT NULL AUTO_INCREMENT,  
  `appname` varchar(20) NOT NULL,  
  `count` int(11) NOT NULL,  
  PRIMARY KEY (`aid`)  
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

统计 app 每个页面用户浏览排行:

```
select appname, count from appCounts order by count desc limit 10
```

2. 业务表 2

业务建表:

```
DROP TABLE IF EXISTS `userCounts`;
```

```
CREATE TABLE `userCounts` (  
  `aid` int(11) NOT NULL AUTO_INCREMENT,  
  `uid` varchar(20) NOT NULL,  
  PRIMARY KEY (`aid`)  
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

统计访问金融 app uv:

```
select count(1) from userCounts
```

### 6.2 Spark Structured Streaming 业务代码开发

1. 通过 maven 引入相关依赖

## 大讲台----课程资料

```
<dependency>
  <groupId>org.apache.spark</groupId>
  <artifactId>spark-sql_2.11</artifactId>
  <version>2.3.0</version>
</dependency>
<dependency>
  <groupId>org.apache.spark</groupId>
  <artifactId>spark-sql-kafka-0-10_2.11</artifactId>
  <version>2.3.0</version>
</dependency>
```

### 2.nc+Spark Structured Streaming 集成开发

### 3.nc+Spark Structured Streaming+mysql 集成开发

Spark Structured 将数据写入 mysql:

<https://databricks.com/blog/2017/04/04/real-time-end-to-end-integration-with-apache-kafka-in-apache-sparks-structured-streaming.html>

### 4.flume+Kafka+Spark Structured Streaming+MySQL

#### 1) 业务代码开发（手动实现）

#### 2) 测试运行

##### a) 启动 Spark Structured Streaming 应用程序

##### b) 启动 flume 聚合脚本

./start-flume1-kafka-applogs.sh

##### c) 启动 flume 采集脚本

bin/start-flume23-applogs.sh

##### d) 启动应用程序模拟产生日志脚本

bin/applog.sh

##### e) 观察数据库结果

## （四）大数据项目可视化

### 1.Eclipse 与 tomcat 集成

#### 1.下载 tomcat

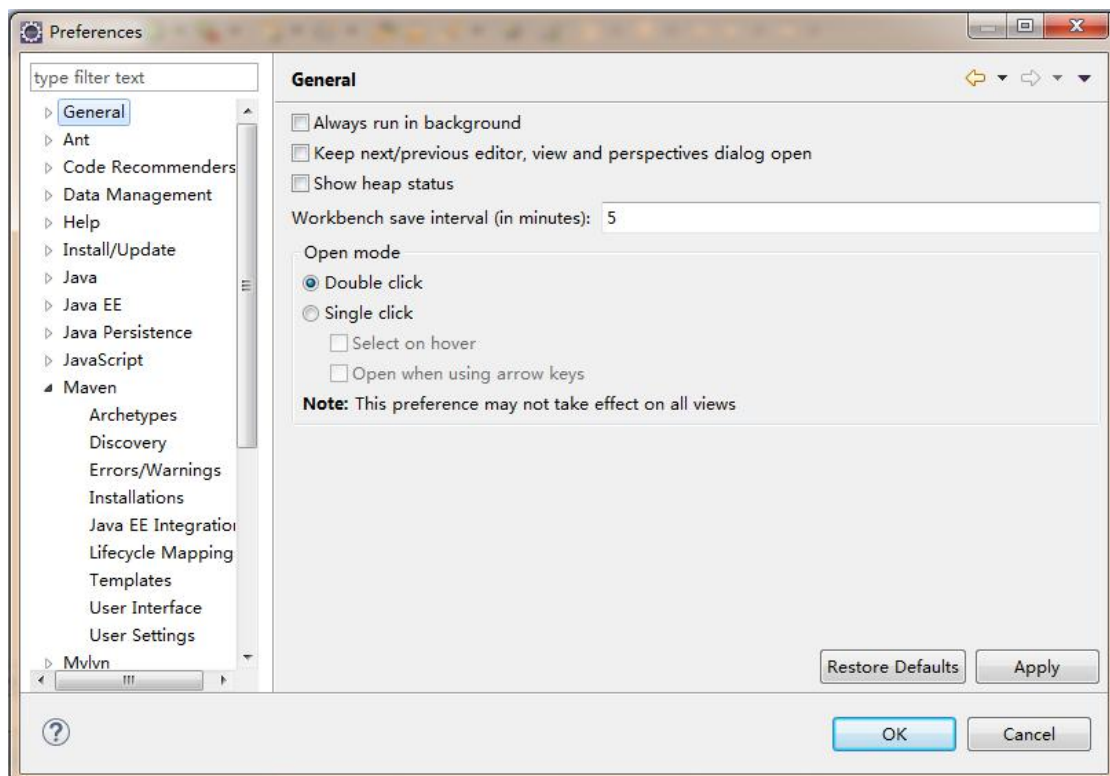
到官网下载 tomcat: <https://tomcat.apache.org/download-70.cgi>

#### 2.解压 tomcat

从官网下载 tomcat 安装包，解压到本地某个目录即可。

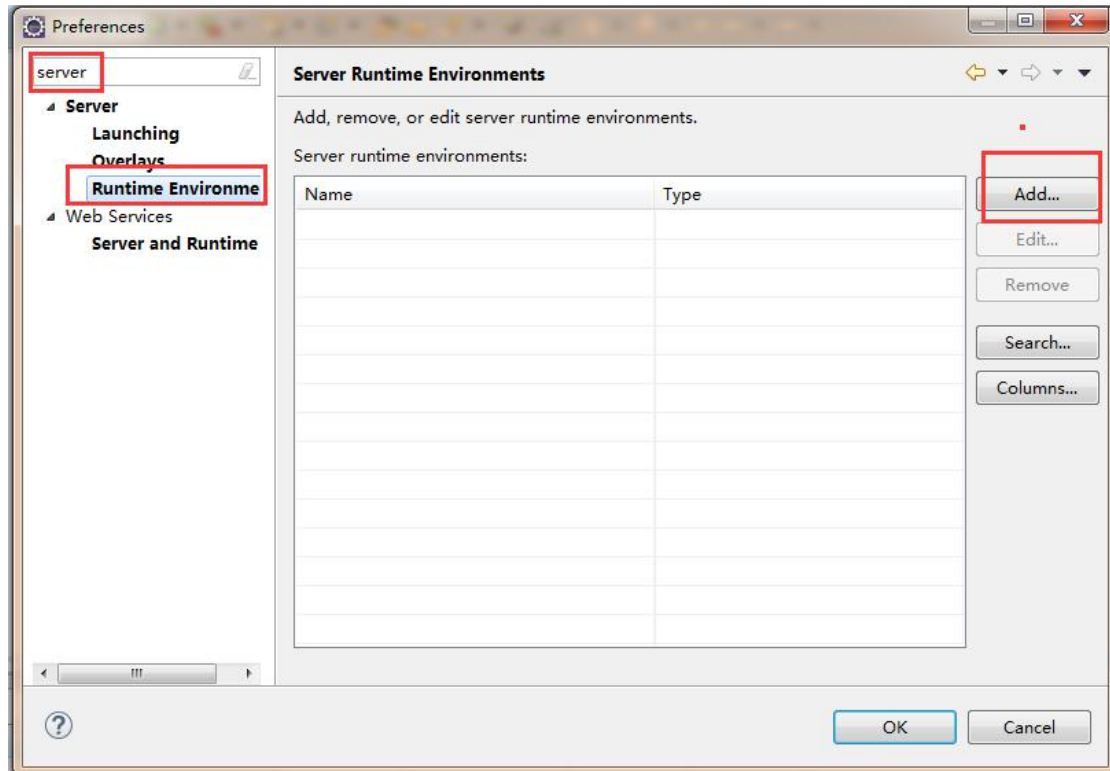
### 3.Eclipse 配置 tomcat

3.1window -> preferences 打开 eclipse 首选项面板



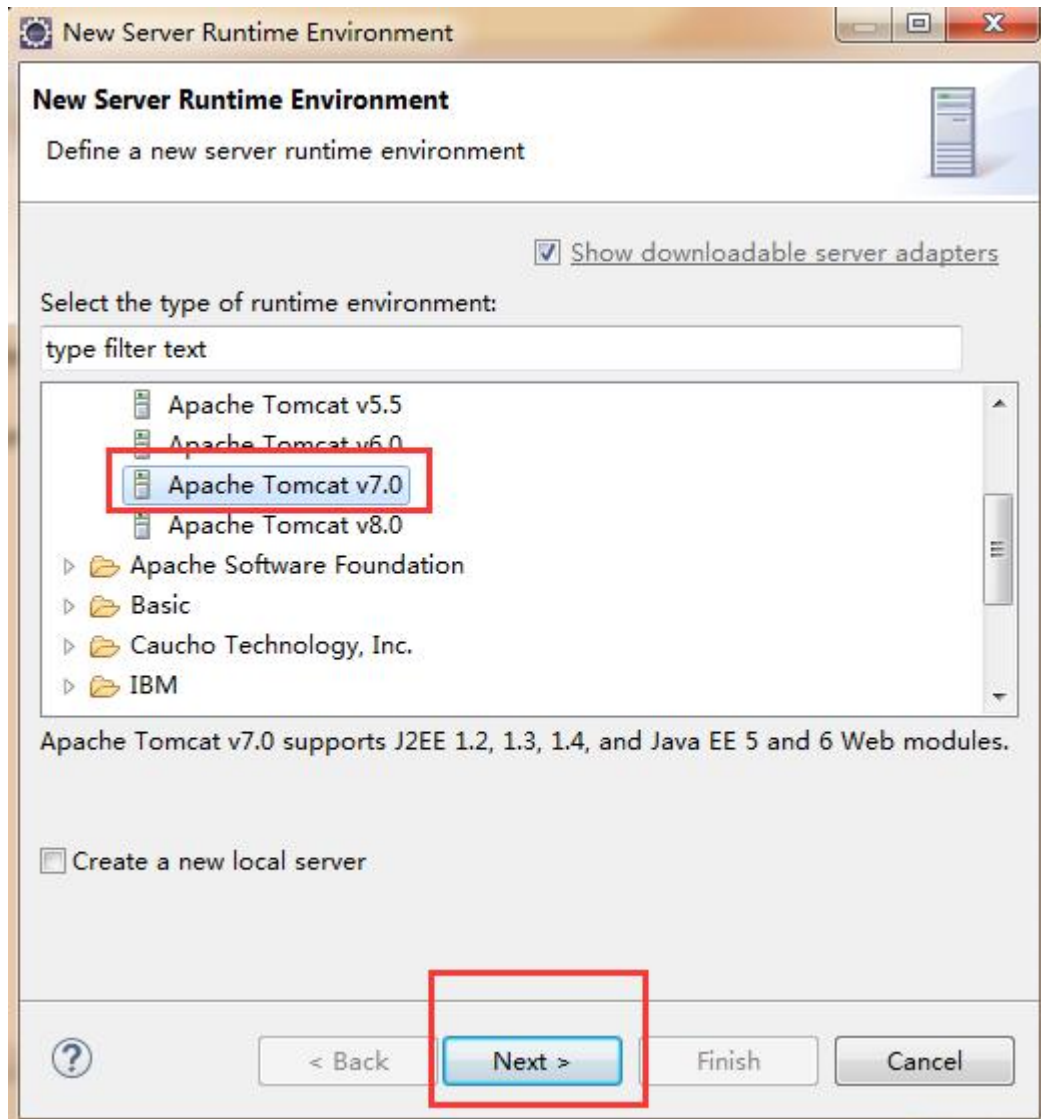
3.2 在“首选项”窗口输入“server”搜索定位到“Runtime Environments”，点击“add”

## 大讲台----课程资料



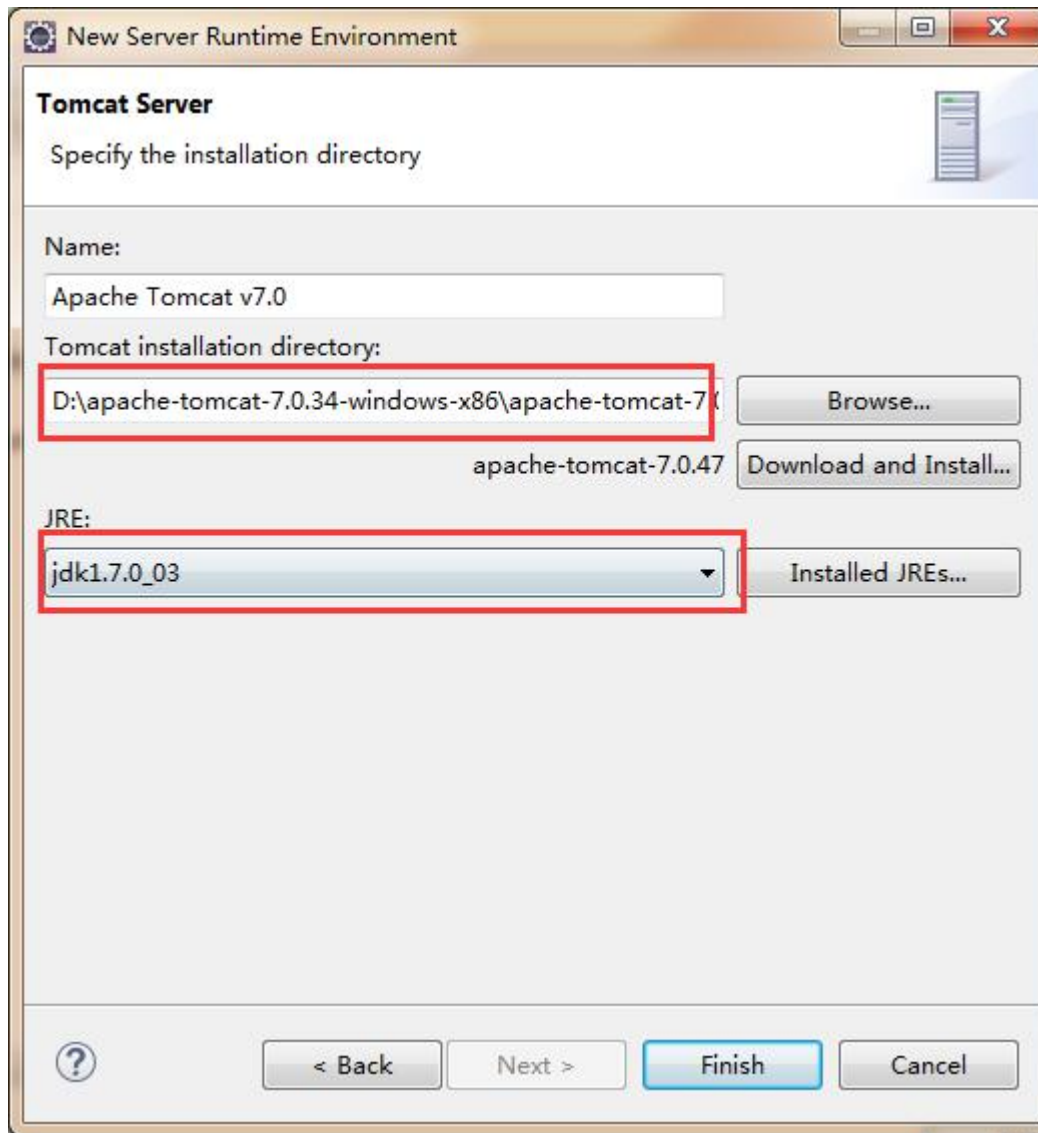
3.3 选择对应的 tomcat 版本，点击 "next"

## 大讲台----课程资料



3.4 设置被添加的 tomcat 的安装目录.设置 JRE，默认使用是 eclipse 自带的 JRE，可以改成独立安装 JDK 中的 JRE，点击 “Finish”

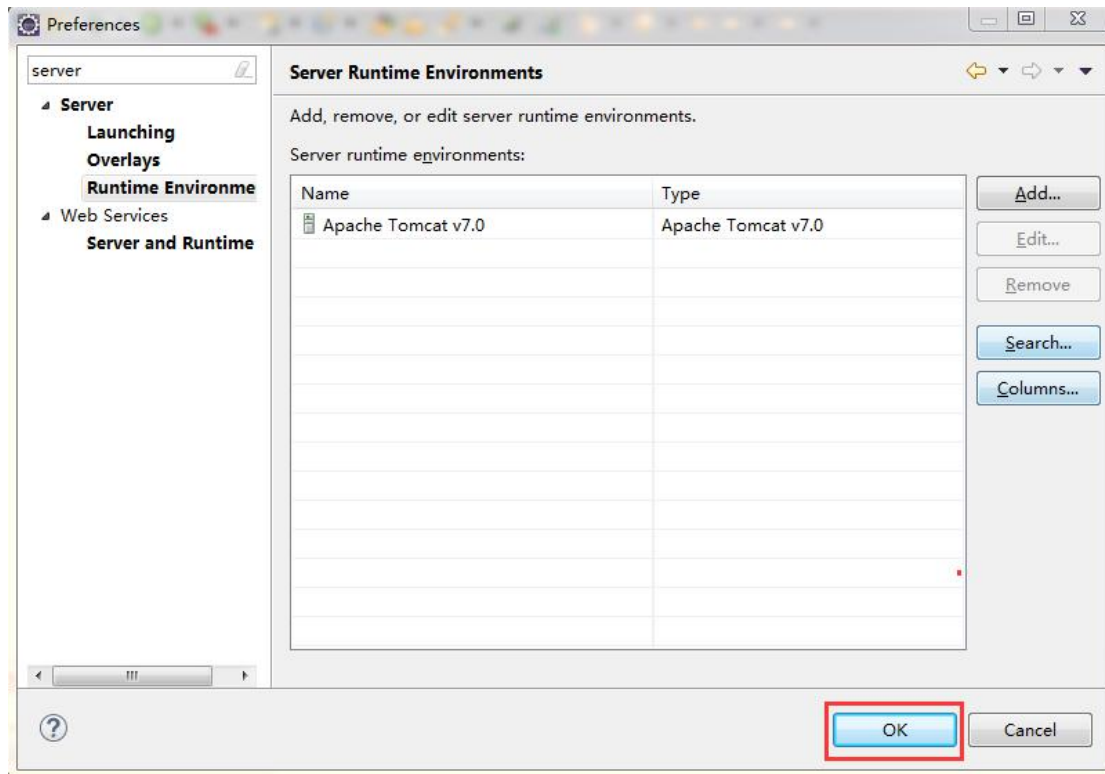
## 大讲台----课程资料



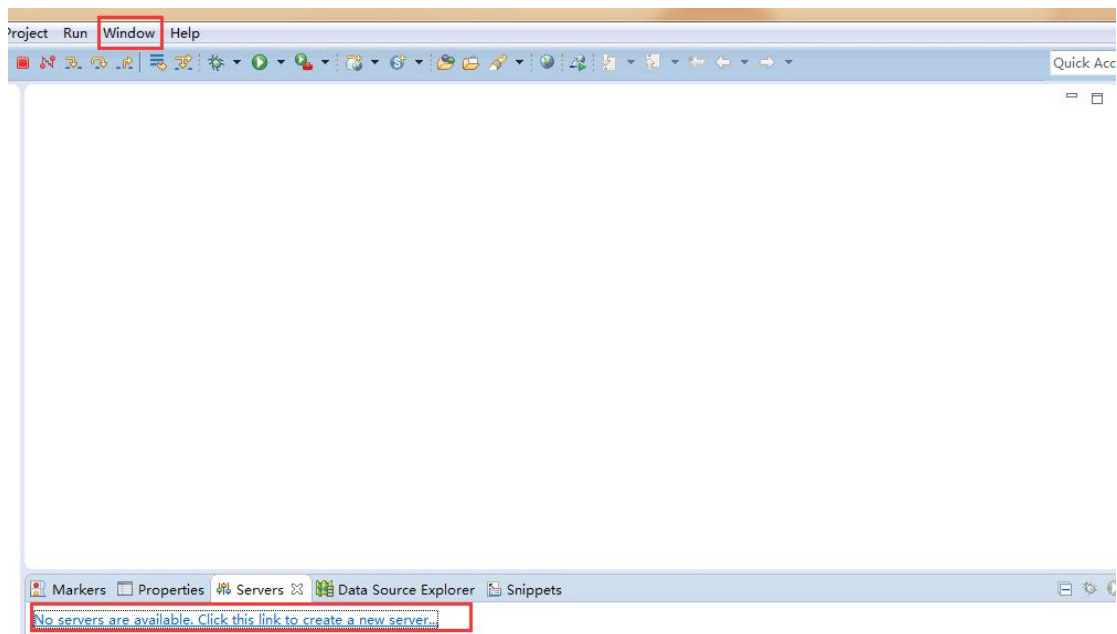
3.5 点击 "ok"



## 大讲台----课程资料

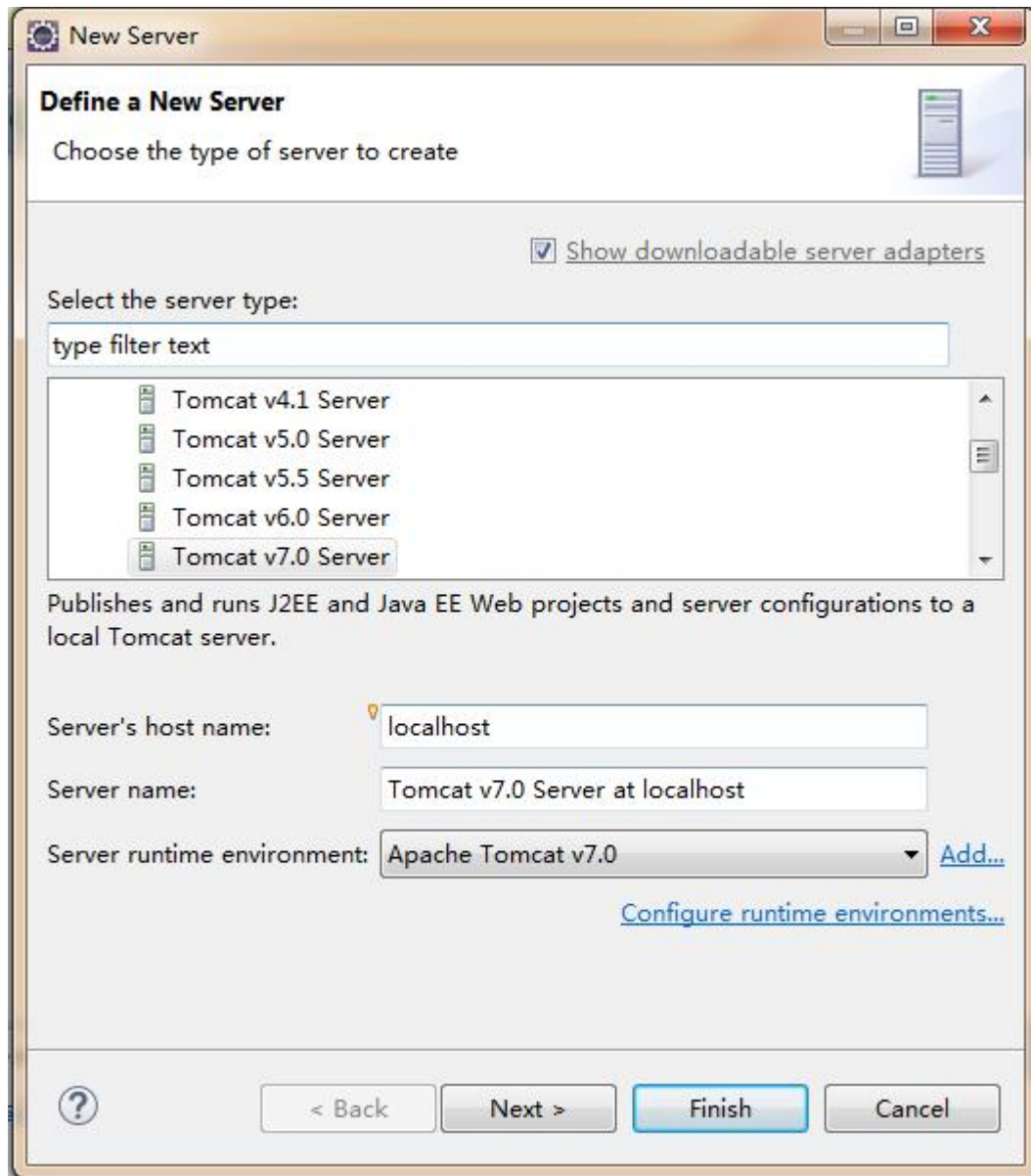


3.6 Window -> show view -> Server 打开“Server” 面板配置 Server。如果没有“Server”，选择“other”，在搜索框输入 Server 定位选项，再双击打开 Server 面板。出现下图所示。



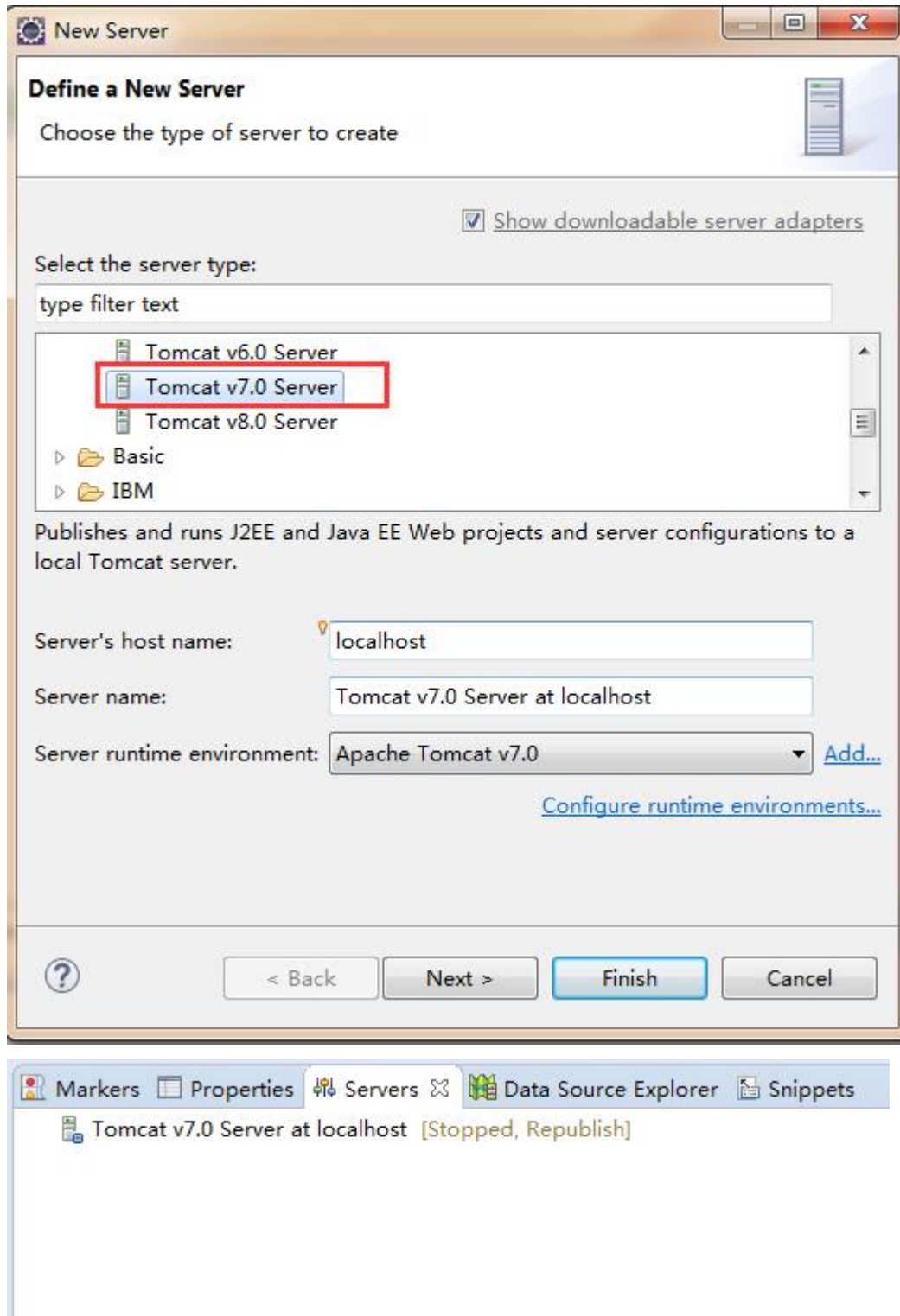
3.7 点击面板中 “click this to create a new server” 超链接，或者在面板内部 右键 “new” 打开新增 Server 实例对话框。

## 大讲台----课程资料



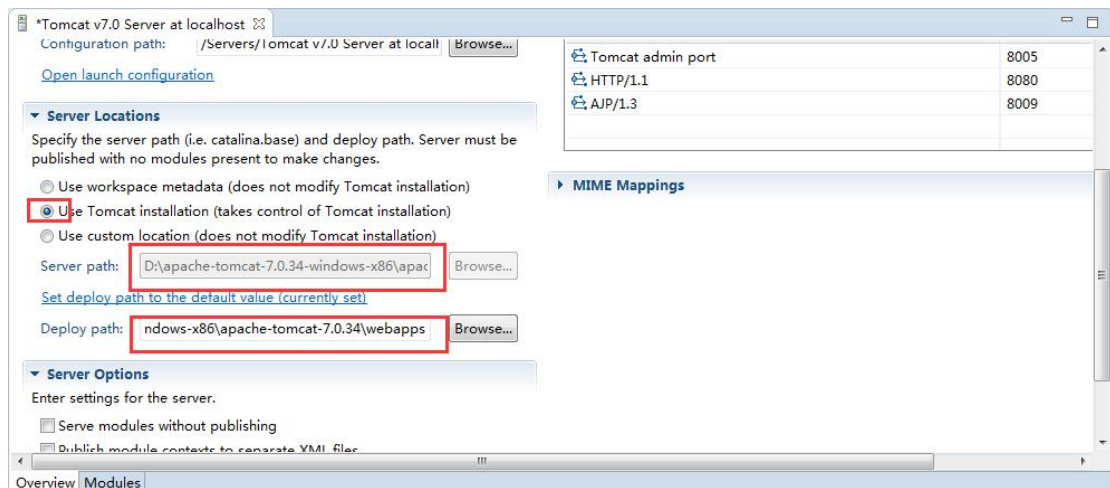
3.8 在新增 Server 实例对话框中选择正确的 Server 版本，点击“Finish”完成 Server 实例创建，此时可以在 Server 面板可以看到一个 tomcat Server 实例。

## 大讲台----课程资料



3.9 双击 tomcat 实例，打开实例配置界面，修改 Server Path 路径。

## 大讲台----课程资料

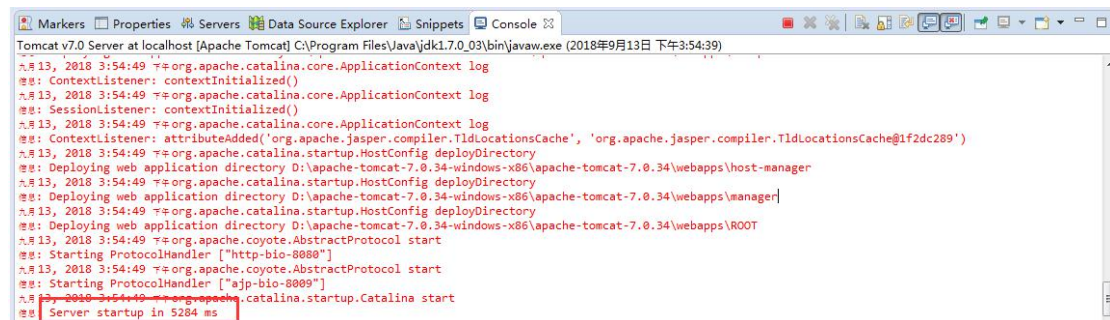


修改完成后，保存，关闭界面即可。

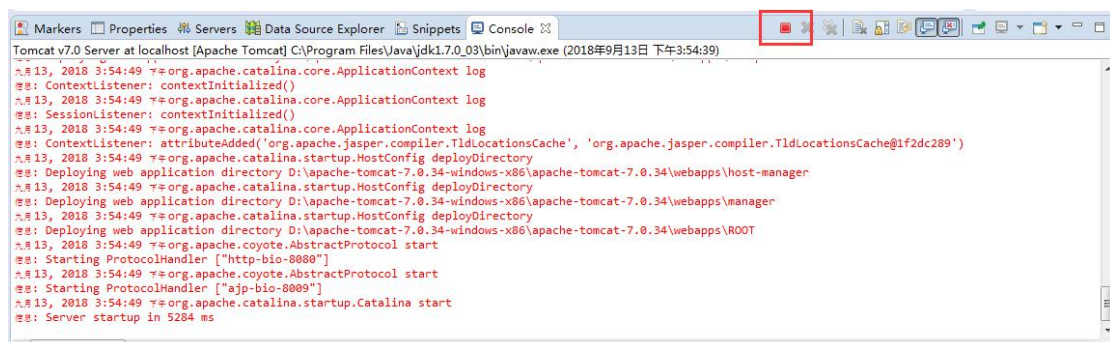
### 3.10 选择 tomcat 实例，右键选项 start 启动。



### 显示 Server 已经启动



可以通过红色按钮关闭 Server。



### 3.11 在浏览器中输入：http://localhost:8080/

# 大讲台----课程资料

http://localhost:8080/ 右侧与男友现身街头


谷歌 Links Links for Apache Tomcat/7.0.34

Home Documentation Configuration Examples Wiki Mailing Lists Find Help

## Apache Tomcat/7.0.34

The Apache Software Foundation  
http://www.apache.org/

If you're seeing this, you've successfully installed Tomcat. Congratulations!

 Recommended Reading:

- [Security Considerations HOW-TO](#)
- [Manager Application HOW-TO](#)
- [Clustering/Session Replication HOW-TO](#)

Server Status  
Manager App  
Host Manager

### Developer Quick Start

- [Tomcat Setup](#)
- [Realms & AAA](#)
- [Examples](#)
- [Servlet Specifications](#)
- [First Web Application](#)
- [JDBC Data Sources](#)
- [Tomcat Versions](#)

#### Managing Tomcat

For security, access to the `manager.webapp` is restricted. Users are defined in:

```
$CATALINA_HOME/conf/tomcat-users.xml
```

In Tomcat 7.0 access to the manager application is split between different users.  
[Read more...](#)

#### Release Notes

#### Documentation

[Tomcat 7.0 Documentation](#)  
[Tomcat 7.0 Configuration](#)  
[Tomcat Wiki](#)

Find additional important configuration information in:

```
$CATALINA_HOME/RUNNING.txt
```

Developers may be interested in:

#### Getting Help

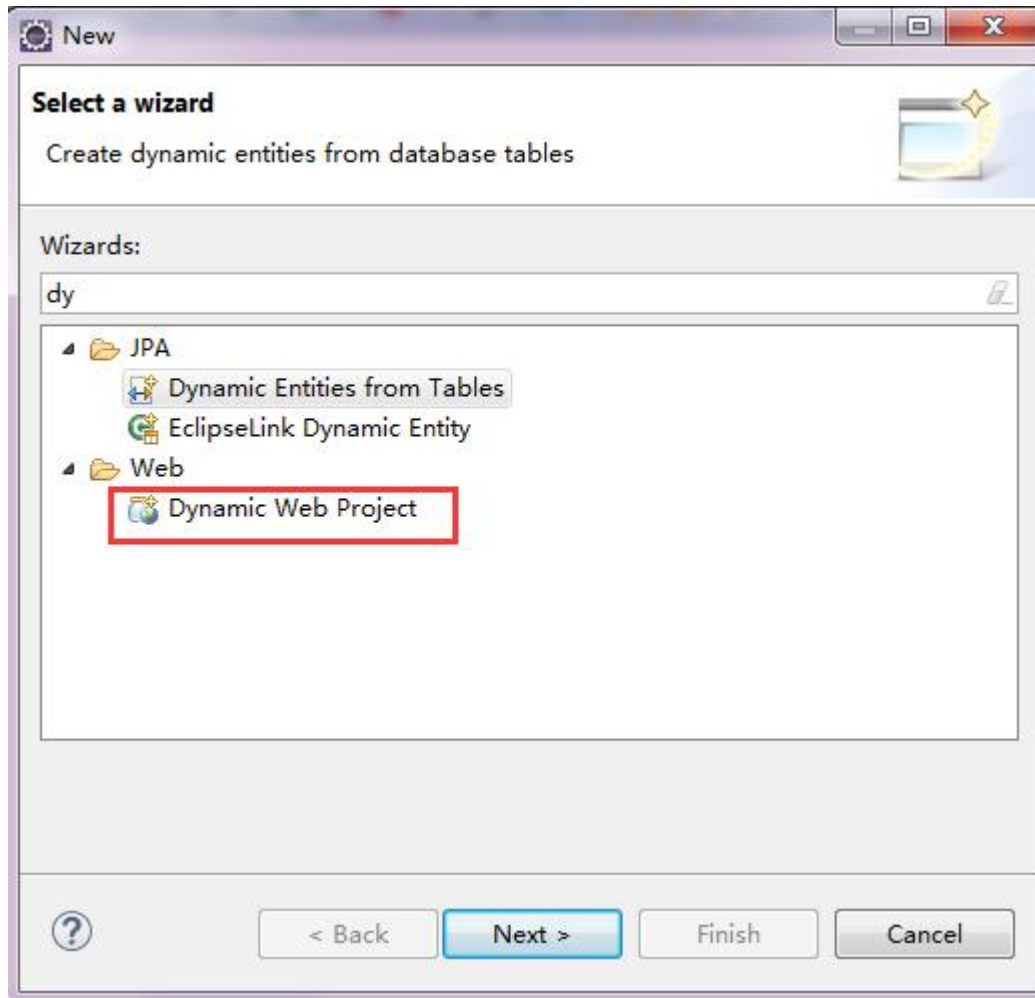
##### FAQ and Mailing Lists

The following mailing lists are available:

- [announce@tomcat.apache.org](mailto:announce@tomcat.apache.org)  
Important announcements, releases, security vulnerability notifications. (Low volume).
- [users@tomcat.apache.org](mailto:users@tomcat.apache.org)  
User support and discussion
- [facils-user@tomcat.apache.org](mailto:facils-user@tomcat.apache.org)

## 2. Eclipse 构建 Web 项目

1.创建 **dynamic web project**，只需要命名为 **newsWeb**，其余默认设置即可。





## 大讲台----课程资料

**New Dynamic Web Project**

**Dynamic Web Project**

Create a standalone Dynamic Web project or add it to a new or existing Enterprise Application.

**Project name:** newsWeb

**Project location**

☒ Use default location

Location: C:\Users\John\workspace\newsWeb Browse...

**Target runtime**

Apache Tomcat v7.0 New Runtime...

**Dynamic web module version**

3.0

**Configuration**

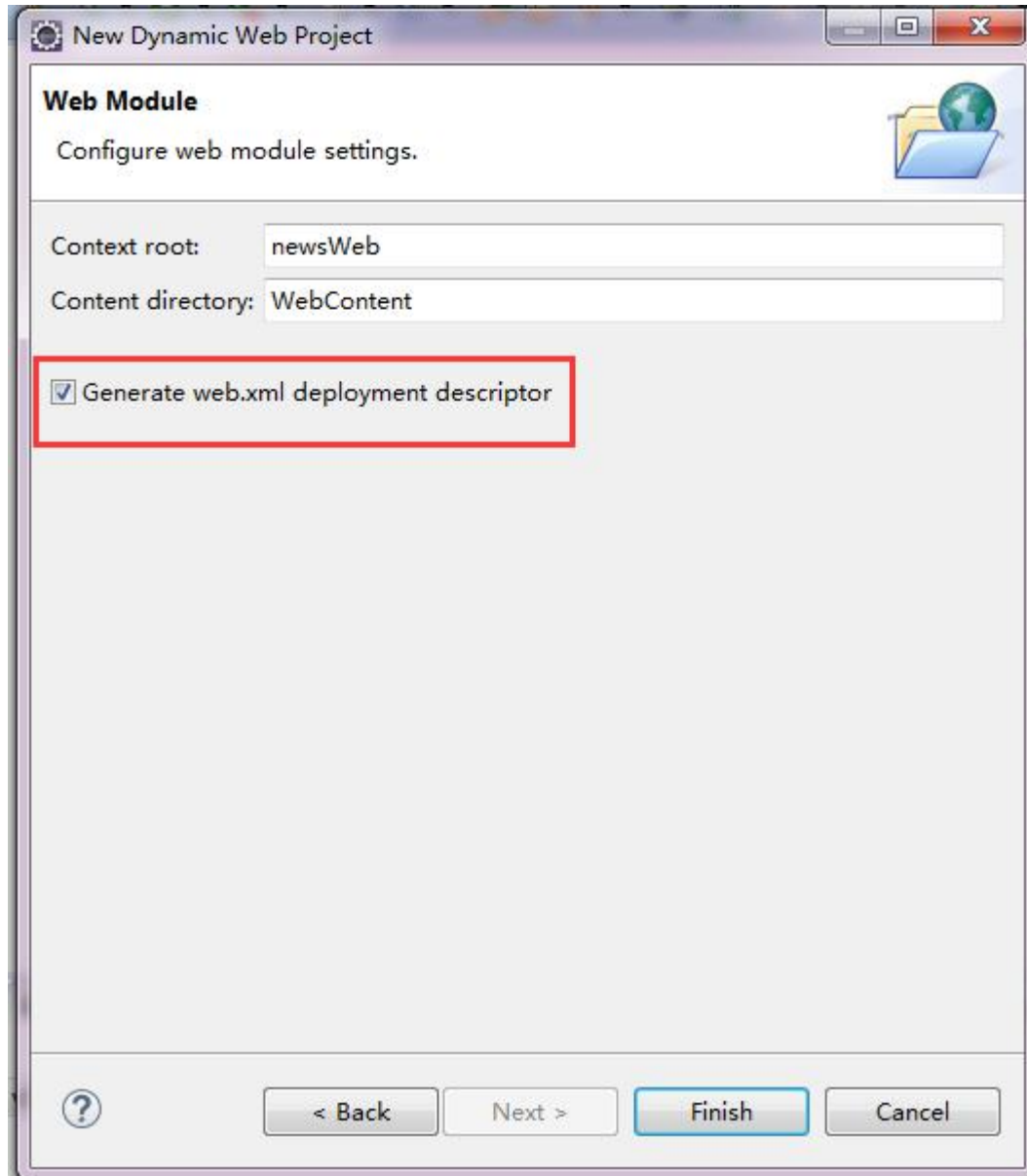
Default Configuration for Apache Tomcat v7.0 Modify...

A good starting point for working with Apache Tomcat v7.0 runtime. Additional facets can later be installed to add new functionality to the project.

**EAR membership**

? < Back Next > Finish Cancel

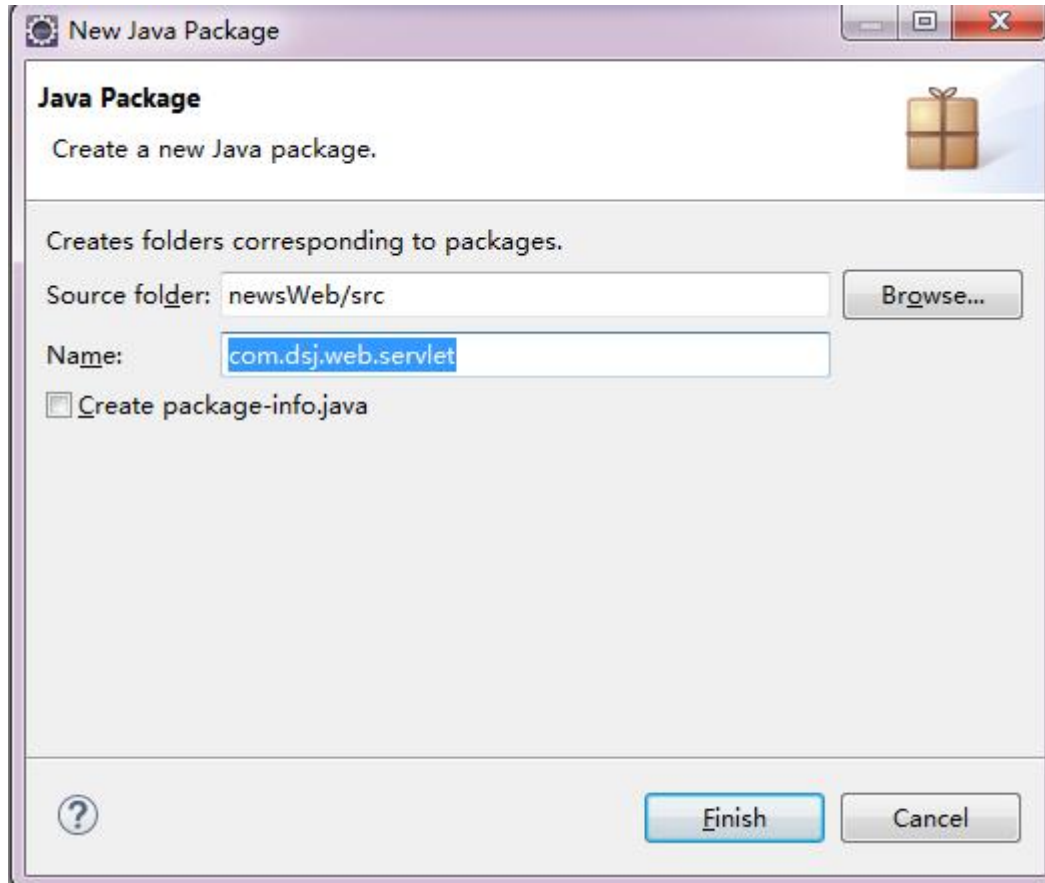
2. next-next-勾选 **generate web.xml deployment description**, 该选项会在目录下生成一个默认的 **xml** 文件。



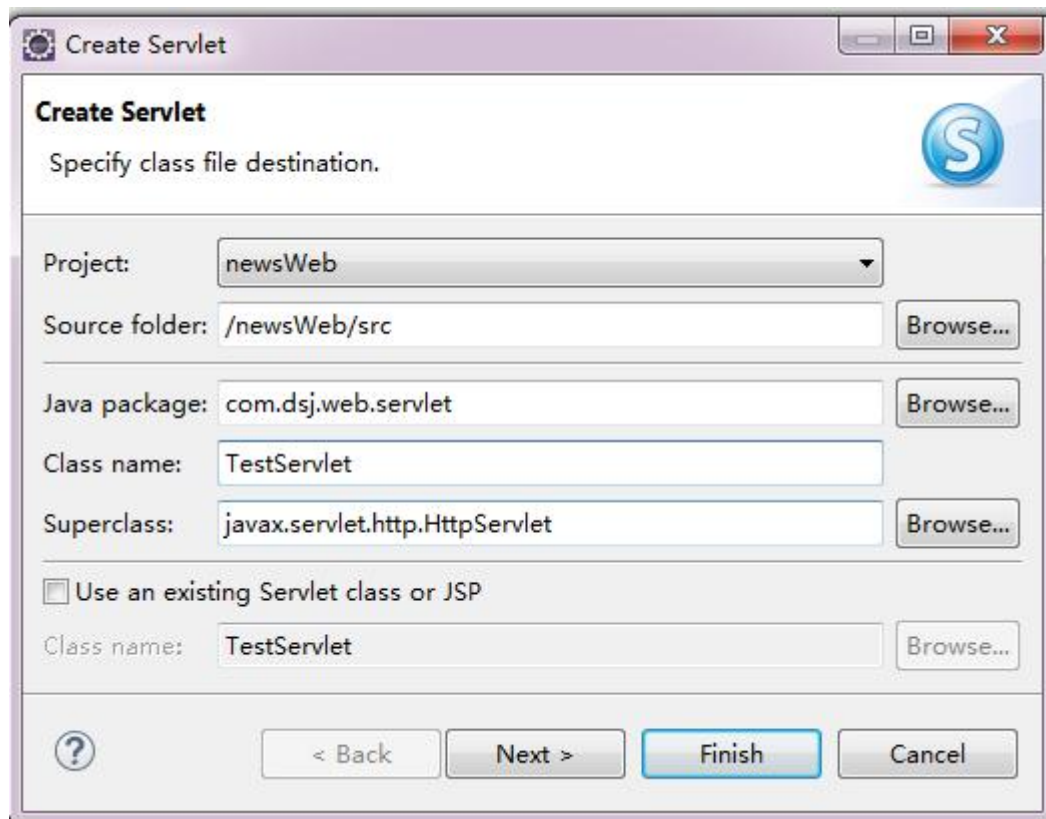


## 大讲台----课程资料

3. 在 java resource 的 src 下建立命名为 `com.ds.j.web.servlet` 的 package，创建名为 `TestServlet` 的类。



## 4.TestServlet 类继承自 HttpServlet。



#注解

@WebServlet("/TestServlet")

```
public class TestServlet extends HttpServlet {  
    private static final long serialVersionUID = 1L;
```

```
    /**  
     * @see HttpServlet#HttpServlet()  
     */  
    public TestServlet() {  
        super();  
        // TODO Auto-generated constructor stub  
    }
```

```
    /**  
     * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse  
response)  
     */  
    protected void doGet(HttpServletRequest request, HttpServletResponse  
response) throws ServletException, IOException {  
        // TODO Auto-generated method stub  
        this.doPost(request, response);  
    }
```

## 大讲台----课程资料

```
}

/**
 * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse
 response)
 */
protected void doPost(HttpServletRequest request, HttpServletResponse
 response) throws ServletException, IOException {
    // TODO Auto-generated method stub
    response.setContentType("text/html;charset=utf-8");
    PrintWriter pw = response.getWriter();
    pw.write("hello world!");
    pw.flush();
    pw.close();
}

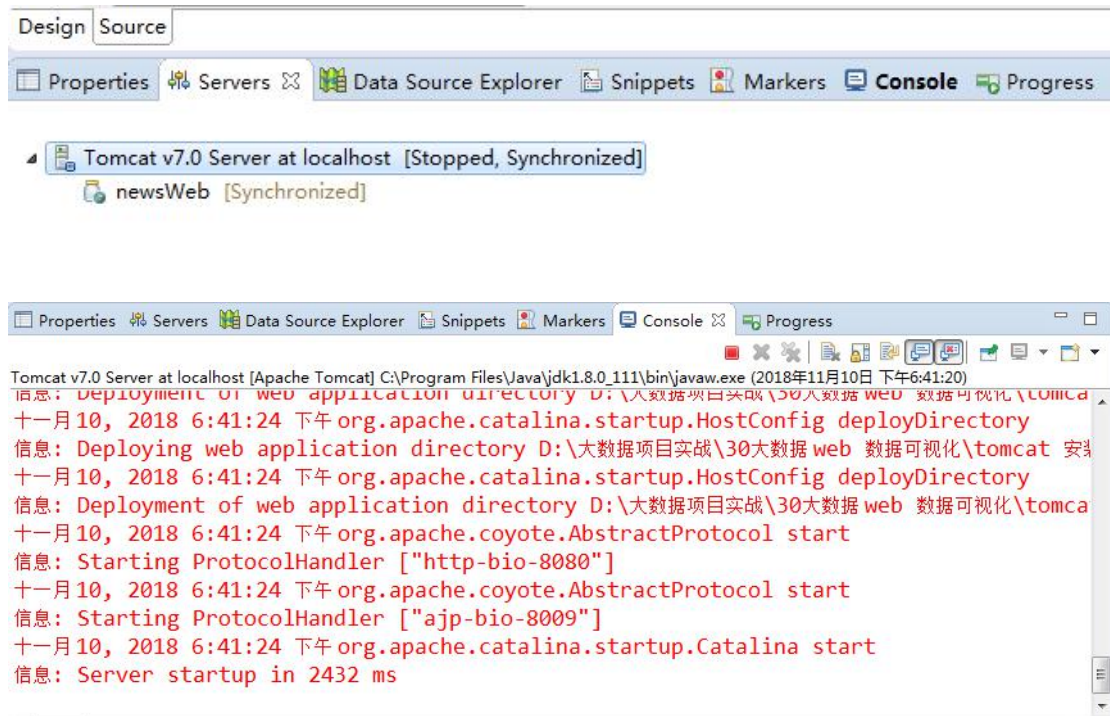
}
```

5. 修改 **web.xml** 的配置，需要自己添加的 **<servlet><servlet-mapping>**，其中 **servlet-name** 为 **servlet** 的命名，对应写的类，**<servlet>**和**<servlet-mapping>**成对出现，定义访问的路径。（可以省略，已经使用注解的方式）

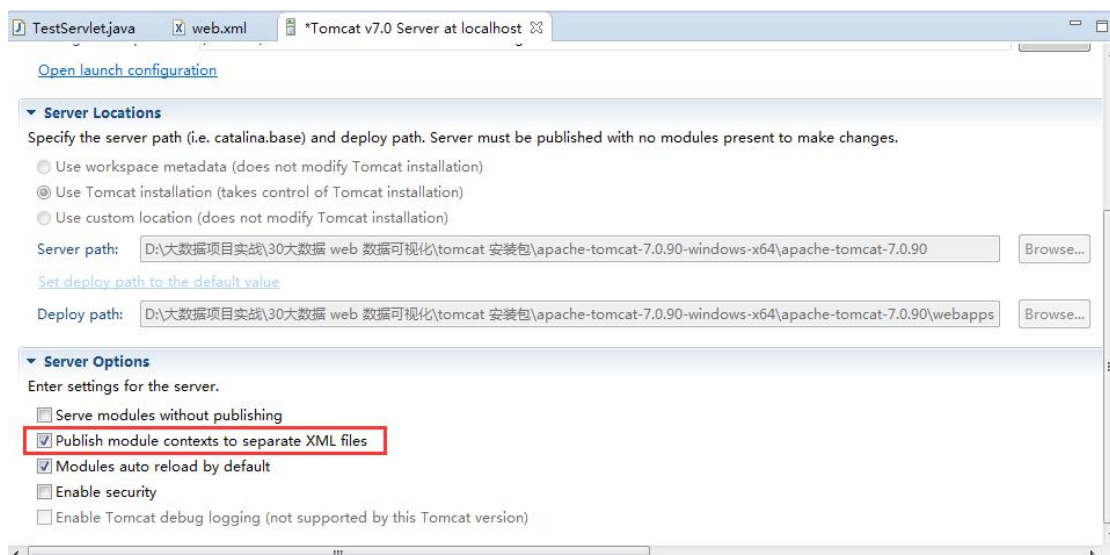
```
<servlet>
    <servlet-name>TestServlet</servlet-name>
    <servlet-class>com.dsj.web.servlet.TestServlet</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>TestServlet</servlet-name>
    <url-pattern>/TestServlet</url-pattern>
</servlet-mapping>
```

## 大讲台----课程资料

6.将项目 **newsWeb** 加载到 **tomcat server**。可以直接把项目拖动到 **tomcat** 的图标上（右键项目名，选择 **Run AS--->Run on Server**，然后点击 **Finish** ），启动。



警告：[SetPropertiesRule]{Server/Service/Engine/Host/Context} Setting property 'source' to 'org.eclipse.jst.jee.server:newsWeb' did not find a matching property.



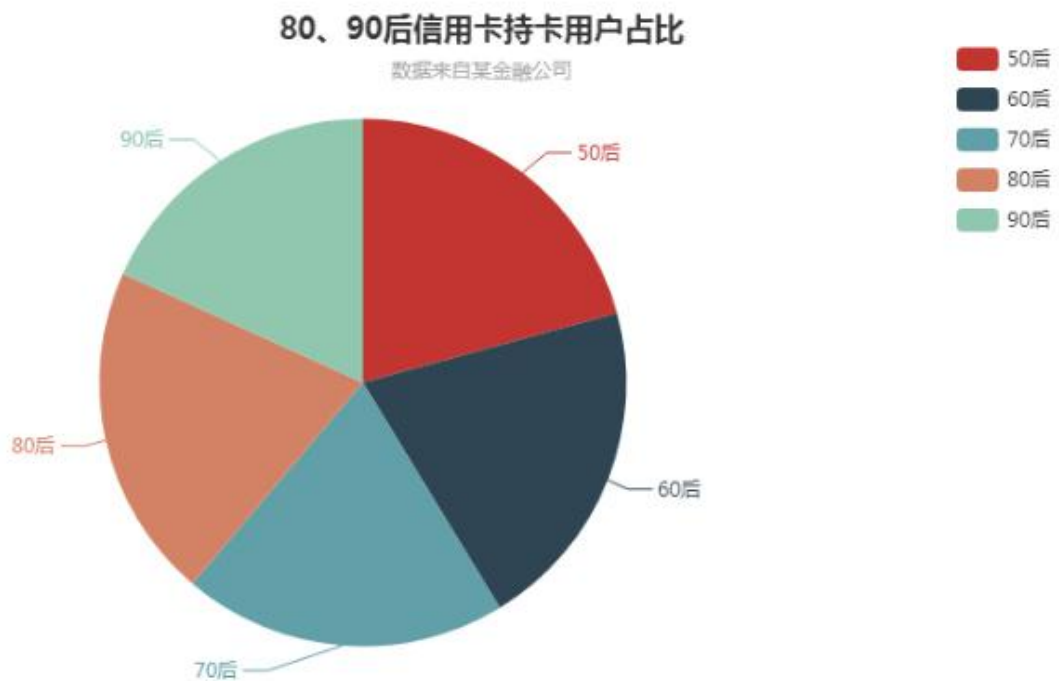
## 7.访问 web 项目

地址: <http://localhost:8080/appWeb/TestServlet>

## 3.离线项目可视化

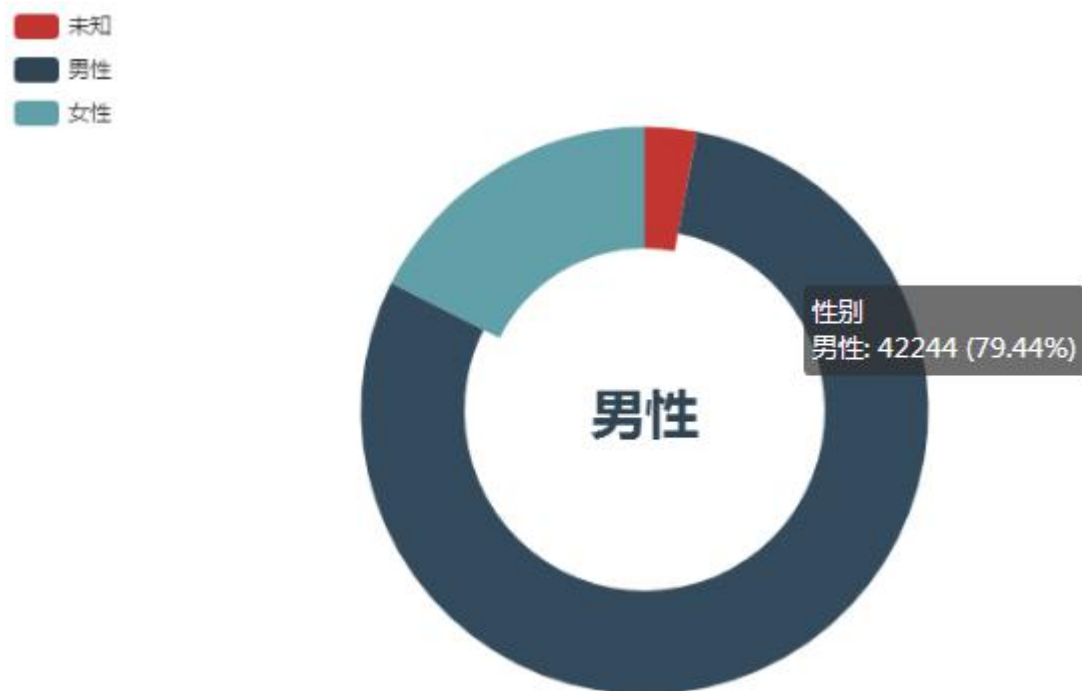
<http://localhost:8080/appWeb/jsp/credit.jsp>

### 1.180 后、90 后不同年龄段，持有信用卡用户占比



## 大讲台----课程资料

### 2.男、女不同性别，持有信用卡用户占比

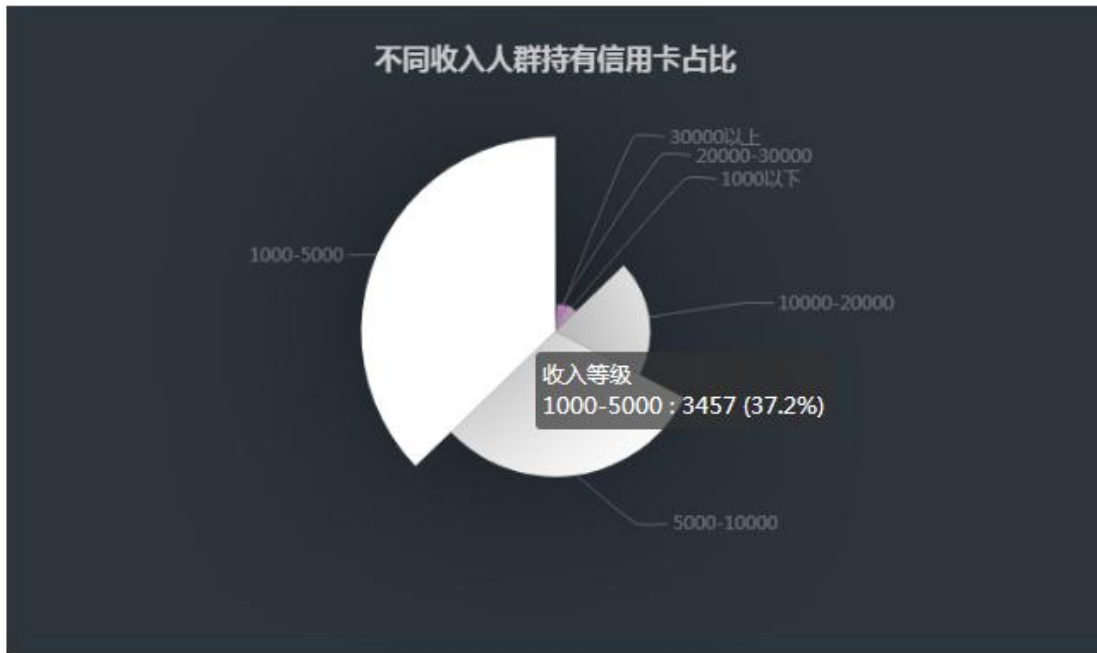


### 3.不同省份，持有信用卡用户占比

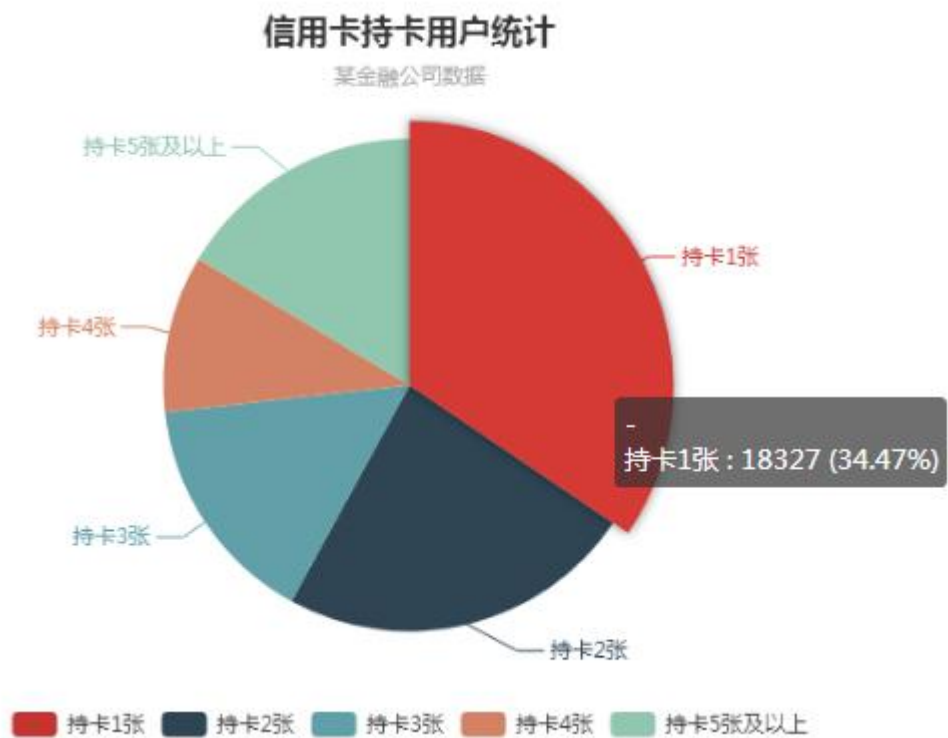


## 大讲台----课程资料

### 4.不同收入等级，持有信用卡用户占比

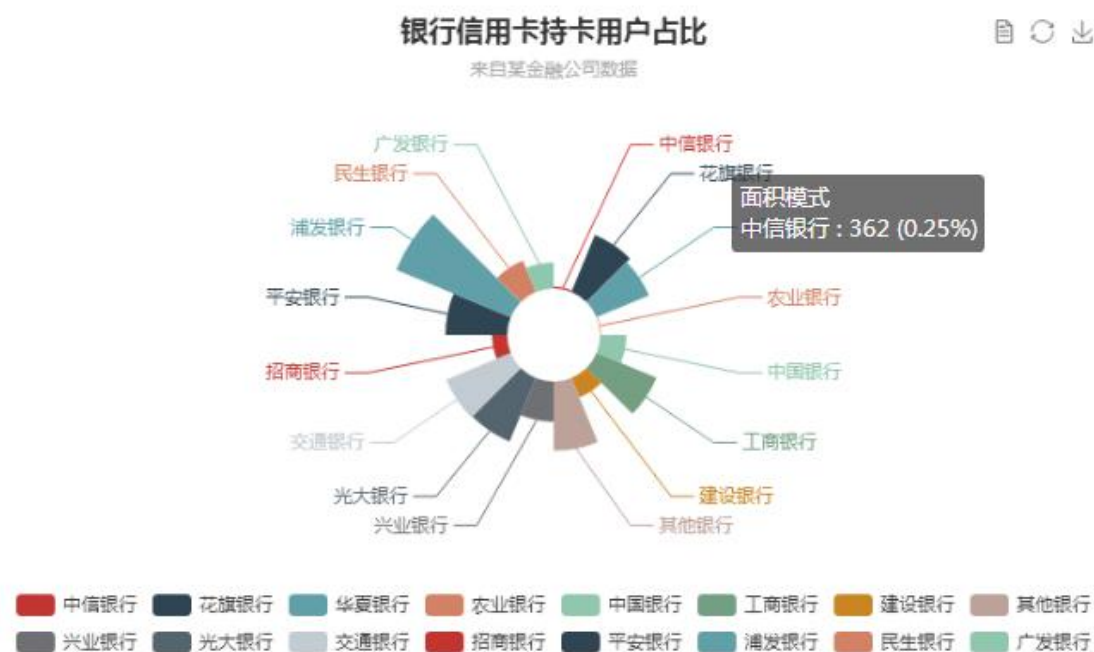


### 5.持有多张信用卡的用户占比

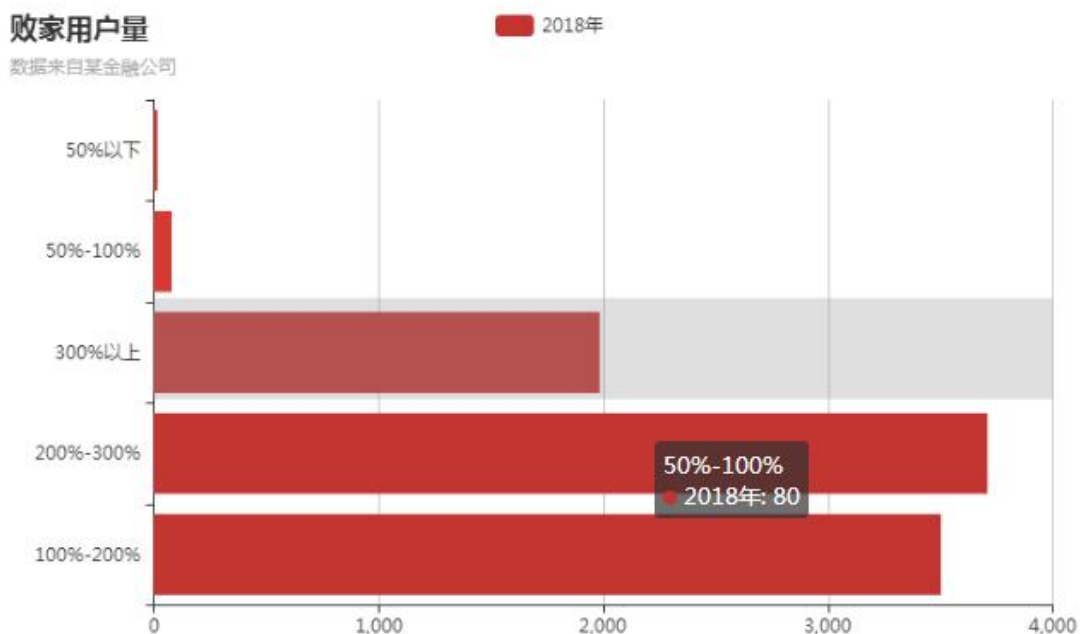


## 大讲台----课程资料

### 6.不同银行，信用卡持有用户占比



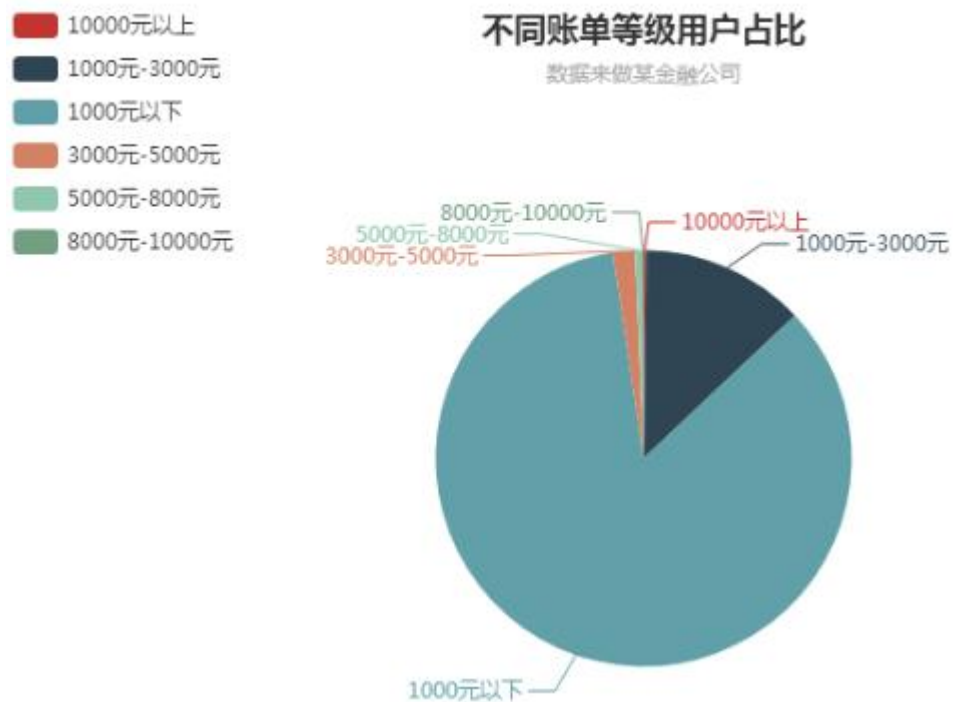
### 7.败家指数：消费超额、消费透支用户占比



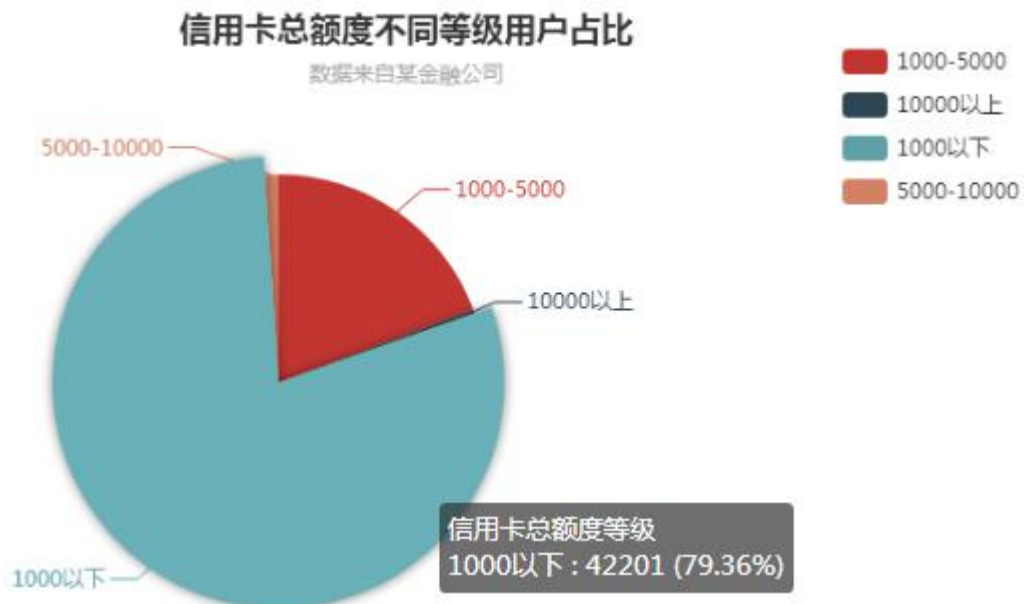


## 大讲台----课程资料

### 8.不同信用卡账单总金额等级，用户占比



### 9.不同信用卡总额度范围，用户占比

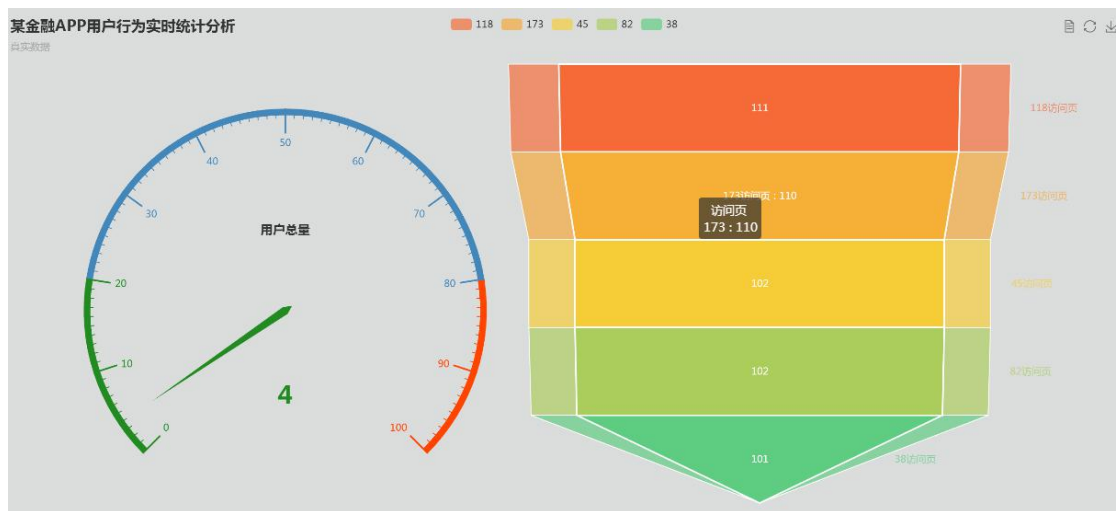


# 大讲台----课程资料

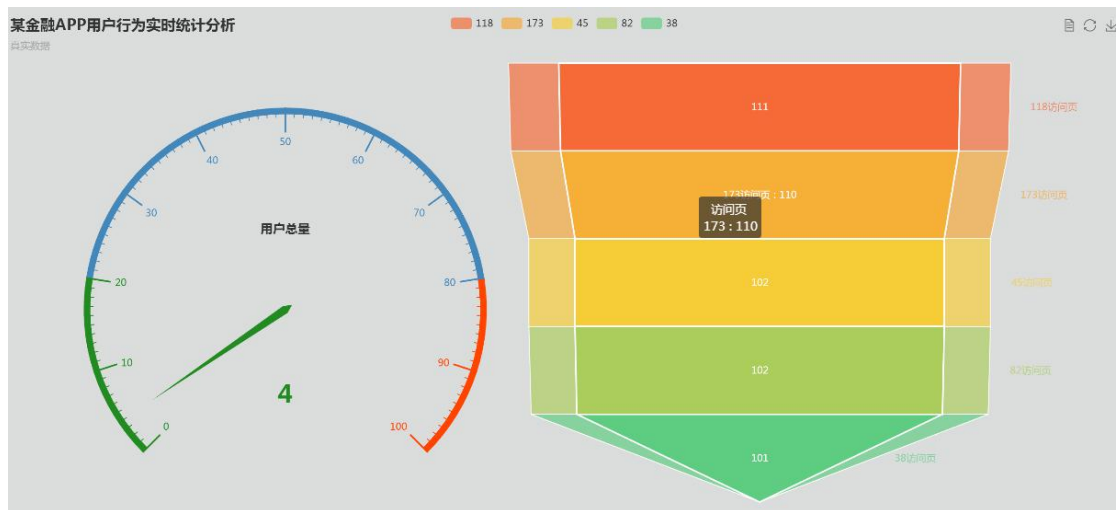
## 4.实时项目可视化

<http://localhost:8080/appWeb/>

### 1.金融 app 页面访问 topn



### 2.金融 app 访问用户 uv



大讲台----课程资料

**QQ 学习交流群:732021751**

**技术咨询微信/QQ:84985152**

**课程咨询助教 QQ : 484166349**