一、微信小程序项目介绍

1.项目名称

微信小程序——《最优保姆》

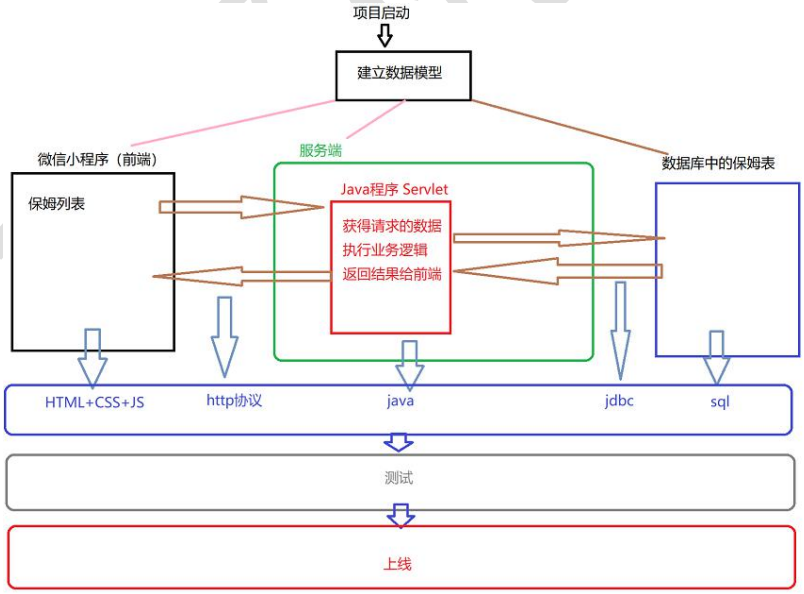
当今世界，大家都很忙，没有时间打理自己的生活，此时就需要有一个统一的，提供家政服务的平台。因此，我们的《最优保姆》产品就问世了！

2.功能

- 保姆列表
- 保姆详情
- 预约
- 我的（订单中心）

3.架构图

在浏览器中：http://localhost:8080/zybm/bmlist ==》得到一个 json 字符串
在微信小程序中：http://localhost:8080/zybm/bmlist ==》 得到一个 json 字符串



4.高保真图

二、项目实施规划

1.根据项目功能提取数据实体及数据项
2.找出实体关系，绘制 E-R 图
3.创建数据表
4.分析小程序与服务器的数据接口
5.开发小程序
6.开发服务端（web 项目）
    a.建库建表
    b.JDBC
    c.Servlet（接受小程序请求并响应）
7.测试
8.部署并上线

三、建立数据模型

1.实体

1）保姆
2）用户
3）订单

2.提取数据项

1）保姆：
    编号、姓名、年龄、性别、工作年限、薪水、教育背景、家乡、工作内容

2）用户：

编号、姓名、性别、地址、年龄...

3）订单:

编号、用户编号、保姆编号、订单时间、订单状态、备注

3.E-R 图　entity　relationship

4.分析小程序与服务器的数据接口(不是 interface)

a)保姆列表

请求 url	http://localhost:8080/zybm/bmlist
请求方式	get
请求参数	firstIndex:0,count:10
返回结果	JSON 字符串 [{保姆 id,bmname,bmsalary...},{}...]

b)保姆详情

请求 url	http://localhost:8080/zybm/bminfo
请求方式	get
请求参数	bmid:1001
返回结果	JSON 字符串 {保姆 id,bmname,bmsalary...}

c)订单列表

请求 url	http://localhost:8080/zybm/orderlist
请求方式	get
请求参数	firstIndex:0,count:10
返回结果	JSON 字符串 [{订单编号,用户编号,保姆编号...},{}...]

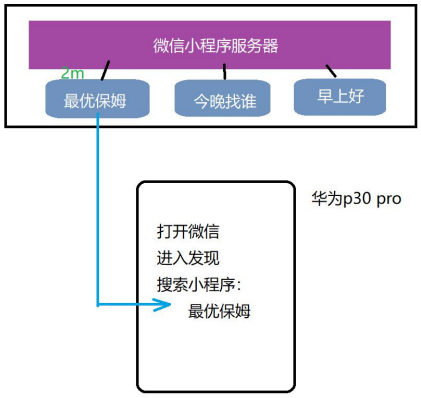
d)订单详情

请求 url	http://localhost:8080/zybm/orderinfo
请求方式	get
请求参数	orderid:1
返回结果	JSON 字符串 {订单编号,用户编号,保姆编号...}

四、小程序介绍

1.什么是小程序

微信小程序是微信 app 提出的一个小功能，可以快速展示信息，以及实现一些小功能的程序。



2.优点

1）快、小
2）方便、功能强大

3.应用场景

1）商业宣传
2）信息展示...

五、如何开发一个自己的小程序

1.在微信平台注册一个小程序

地址：https://mp.weixin.qq.com

- 注册程序（QQ 邮箱，实名认证的微信）
- 登录到微信小程序的管理界面（QQ 邮箱，微信）
- 设置小程序基本信息（名称，简称，图标，简介，分类）
- 获取小程序的 AppID

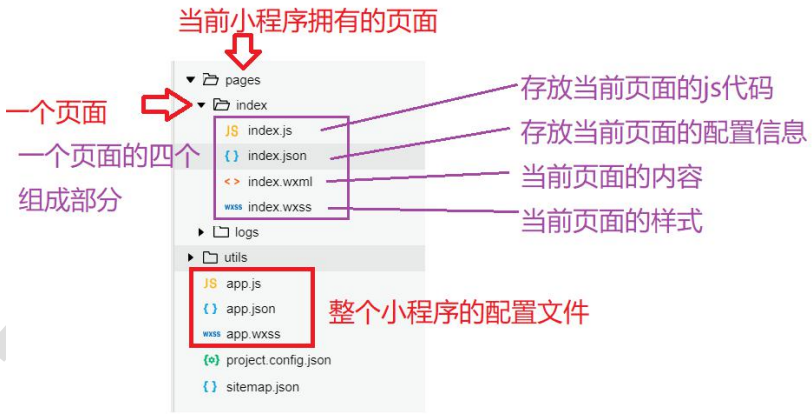
2.下载微信 web 开发者工具

3.进行小程序的开发

一定要对官方的文档很熟练

https://developers.weixin.qq.com/miniprogram/dev/framework/

1) 小程序的基本结构



2）app.json

4.小程序本地测试

5.小程序发布

六、最优保姆的开发

1.搭建整体框架

1）创建页面

2）在目录中新建 page

3）在 app.json 中设置窗体（window）信息

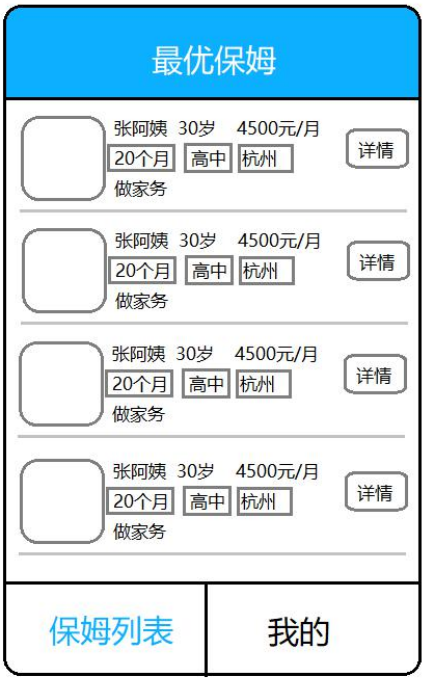
```
"window": {
  "backgroundTextStyle": "dark",
  "navigationBarBackgroundColor": "#3399ff",
  "navigationBarTitleText": "最优保姆",
  "navigationBarTextStyle": "white",
  "enablePullDownRefresh": true
},
```

4）在 app.json 中设置 tabBar 底部导航栏的信息

```
"tabBar": {
  "color": "#000000",
  "selectedColor": "#3399ff",
  "list": [
    {
      "pagePath": "pages/index/index",
      "text": "保姆列表"
    },
    {
      "pagePath": "pages/self/self",
      "text": "我的"
    }
  ]
}
```

2.创建保姆列表页

预期效果：



1）列表页的布局

wxml:

app.json | index.wxml | index.

```
<view id='single'>
  <view id='d1'>1</view>
  <view id='d2'>2</view>
  <view id='d3'>3</view>
</view>
```

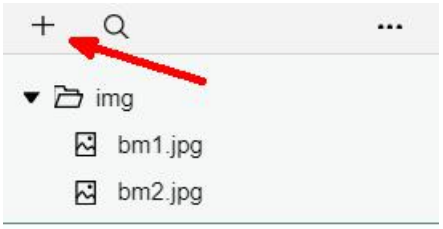
wxss:

```
#single{
  width: 750rpx;
  height: 150rpx;
  background-color: pink;
}
#d1{
  width: 150rpx;
  height: 150rpx;
  background-color: red;
  float:left;
}
#d2{
  width: 450rpx;
  height: 150rpx;
  background-color: green;
  float:left;
}
#d3{
  width: 150rpx;
  height: 150rpx;
  background-color: blue;
  float:left;
}
```

2)创建列表页中的内容

（1）保姆头像

- 新建 img 文件，再存入图片



- wxml:

```
<view id='d1'>
  <image class='bm_img' src='../../img/bm1.jpg'></image>
</view>
```

-wxss:

```
#d1{
  width: 150rpx;
  height: 150rpx;
  float:left;
  padding:1px;
  box-sizing: border-box;
}
.bm_img{
  width: 146rpx;
  height: 146rpx;
  border:1rpx solid #132639;
  box-sizing: border-box;
  border-radius: 10rpx;
}
```

（2）保姆信息

- wxml：目前数据是写死的，以后数据来自于后端提供。

```
<view id='d2'>
  <view id='il1'>
    <text class="bmname_style">张阿姨</text>
    <text class="age_style">20岁</text>
    <text class="salary_style">4500元/月</text>
  </view>
  <view id='il2'>
    <text class="jobtime_style">20个月</text>
    <text class="edu_style">高中</text>
    <text class="home_style">杭州</text>
  </view>
  <view id='il3'>
    <text class="job_style">做家务</text>
  </view>
</view>
```

-wxss:

```
#il1{
  padding-top: 0rpx;
}
#il1 text{
  margin-left:25rpx;
  font-size: 35rpx;
  color: #132639;
}

#il2 text{
  font-size: 30rpx;
  border: 1rpx solid #132639;
  margin-left: 25rpx;
  padding:3rpx 8rpx;
}
#il2{
  margin-top:3rpx;
}
#il3{
  font-size:35rpx;
  color:#4d0026;
  padding-left:25rpx;
  padding-top:5rpx;
}
```

（3） “详情”按钮

- wxml:目前用 text，也可以用 button

```
<view id='d3'>
    <text class="infobtn_style">详情</text>
</view>
```

-wxss:

```
#d3{
  width: 150rpx;
  height: 150rpx;
  float:left;
  padding-top:50rpx;

}


.infobtn_style{
  border: 1rpx solid #132639;
  padding: 10rpx 20rpx;
  border-radius: 10rpx;
}
```

目前的效果：

3）实现页面的数据绑定　　（双向数据绑定）

差值表达式：　{{变量}}

先在 index.js 文件中的 data：{设置键值对}.

```
Page({
  data: {
    "bminfo":{
      "bmname":"李阿姨",
      "age":23,
      "salary":5500,
      "job_time":26,
      "edu":"本科",
      "home":"杭州",
      "job":"做家务，家教"
    }
  }
})
```

再在页面中使用插值表达式来获取 data 中的数据：

```
<view id='single'>
  <view id='d1'>
    <image class='bm_img' src='../../img/bm1.jpg'></image>
  </view>
  <view id='d2'>
    <view id='il1'>
      <text class="bmname_style">{{bminfo.bmname}}</text>
      <text class="age_style">{{bminfo.age}}岁</text>
      <text class="salary_style">{{bminfo.salary}}元/月</text>
```

4）页面的多条数据的循环展示

多个对象，是封装在数组内，存进 data 中。因此，前端页面需要通过遍历该数组，遍历显示多个对象。

index.js:

```
Page({
  data: {
    "bminfos":[
      {
      "bmname":"李阿姨",
      "age":23,
      "salary":5500,
      "job_time":26,
      "edu":"本科",
      "home":"杭州",
      "job":"做家务，家教"
      },
      {
        "bmname": "张阿姨",
        "age": 25,
        "salary": 7500,
        "job_time": 20,
        "edu": "硕士",
        "home": "武汉",
```

index.wxml:

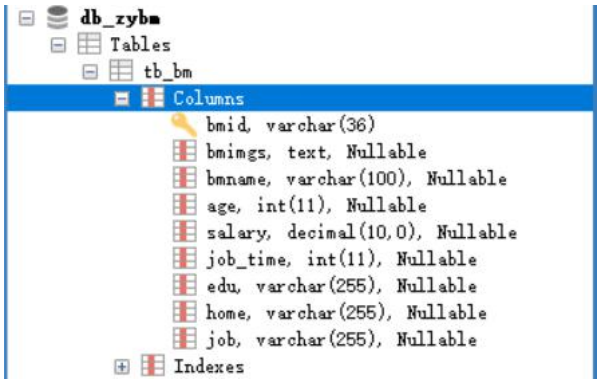
注意，wx:for 的写法：在之前的单个数据展示框外套上循环的 view

在循环中的具体的数据的获取：item　表示一个数组中的一个元素（对象）

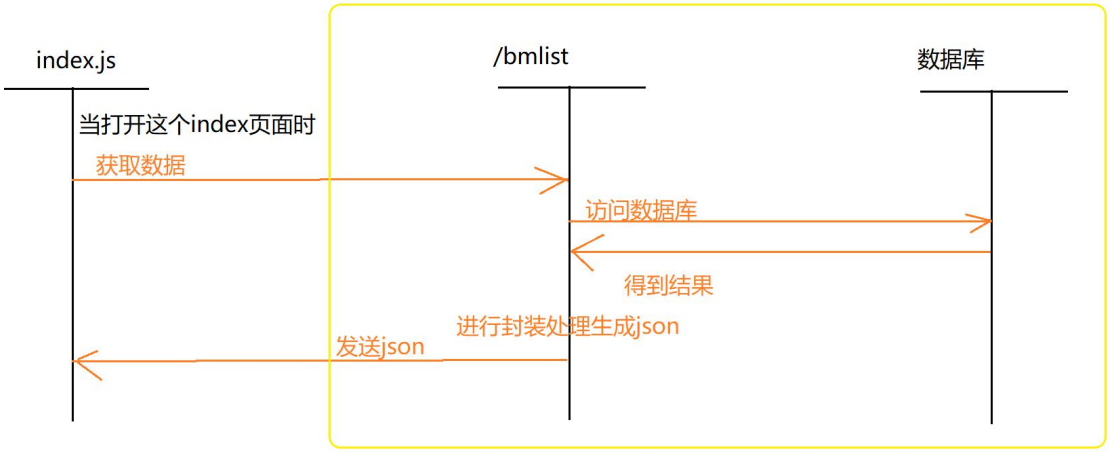
```
<view wx:for="{{bminfos}}">
  <view class='single' >
    <view class='d1'>
      <image class='bm_img' src='../../img/bm1.jpg'></image>
    </view>
    <view class='d2'>
      <view class='il1'>
        <text class="bmname_style">{{item.bmname}}</text>
        <text class="age_style">{{item.age}}岁</text>
        <text class="salary_style">{{item.salary}}元/月</text>
      </view>
      <view class='il2'>
        <text class="jobtime_style">{{item.job_time}}个月</text>
        <text class="edu_style">{{item.edu}}</text>
        <text class="home_style">{{item.home}}</text>
      </view>
      <view class='il3'>
        <text class="job_style">{{item.job}}</text>
```

3.从后端获取数据

1）建库建表



2）时序图



3）具体的后端的开发

注意：java.math.BigDecimal: 用于描述一个固定精度的小数的类

```
protected void doGet(HttpServletRequest request, HttpServletResponse response)
                throws ServletException, IOException {
            // 1.去数据库查询所有的保姆
            String json = null;
            Connection conn = null;
            PreparedStatement pstmt = null;
            ResultSet rs = null;
```

```
try {
	conn = MyJDBCUtil.getConnection();

	String sql = "select * from tb_bm";
	pstmt = conn.prepareStatement(sql);
	rs = pstmt.executeQuery();
	// 2.封装成装着多个BaoMu对象的集合
	List<BaoMu> bms = new ArrayList<BaoMu>();
	while (rs.next()) {
		BaoMu bm = new BaoMu();
		// 封装对象
		bm.setBmid(rs.getString("bmid"));
		bm.setBmname(rs.getString("bmname"));
		bm.setAge(rs.getInt("age"));
		bm.setBmimgs(rs.getString("bmimgs"));
		bm.setEdu(rs.getString("edu"));
		bm.setHome(rs.getString("home"));
		bm.setJob(rs.getString("job"));
		bm.setJob_time(rs.getInt("job_time"));
		bm.setSalary(rs.getBigDecimal("salary"));
		// 将对象添加到集合中
		bms.add(bm);

	}
	// 3.转换成json字符串
	Gson gson = new Gson();
	json = gson.toJson(bms);

} catch (SQLException e) {
	// TODO Auto-generated catch block
	e.printStackTrace();
} finally {
	try {
		MyJDBCUtil.releaseResource(conn, pstmt, rs);
	} catch (SQLException e) {
		// TODO Auto-generated catch block
		e.printStackTrace();
	}

}
// 4.响应给前端

response.setCharacterEncoding("utf-8");
response.setContentType("application/json");
```

```
        response.getWriter().write(json);
    }
```

注意：把数据库中的图片字符串变成字符串数组

```
//把图片的字符串编程字符串数组
//../../img/bm3.jpg,../../img/bm4.jpg
String imgs = rs.getString("bmimgs");
String[] bmimgs = imgs.split(",");
bm.setBmimgs(bmimgs);
```

4.微信小程序如何访问后端 java 程序并获得数据

1）index.js 中 page()中的生命周期函数。

这个函数，是当页面加载时调用的。因此可以在这个函数内发送请求来获取后端提供的 json 数据。

onLoad:function(){}

2）怎样发送请求

```
wx:request({
    url:"请求的具体的资源路径",
    data:{
        //要传递的参数的键值对
        "username":"chenpengfei",
        "pwd":"123"
    },
    success(res){
        //请求成功后，将会调用此函数。
        //res 中封装了返回的数据对象
    }
});
```

```
/**
```

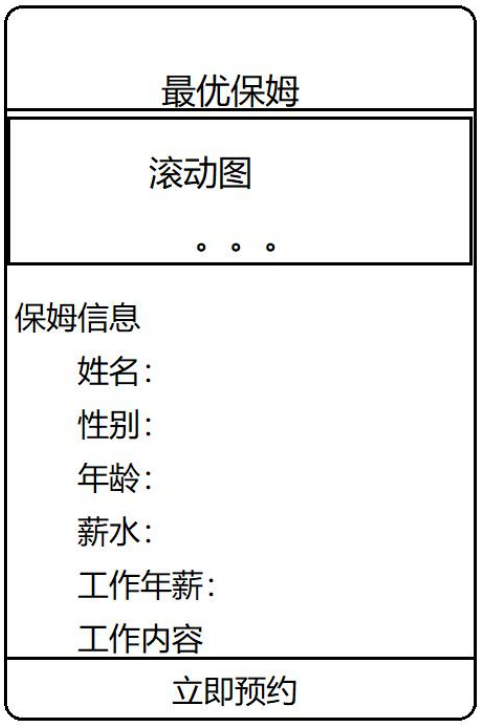
```
 * 当页面加载时就会调用此函数
 */
onLoad: function (options) {
  //options 里存放其他页面跳转至此页面时传递来的值。
  //发送请求，请求服务端的数据
  wx.request({
    url:"http://localhost:8080/zybm/bmlist",
    //当请求成功后调用 getInfos 函数，this 是服务器返回的数据
    success:this.getInfos.bind(this)
  });
},
//将服务器返回的数据（res.data）通过 this.setData 来存放到 data 中。
getInfos:function(res){
  this.setData({"bminfos":res.data});
}
```

3)发送请求后碰到合法域名的问题

当我们在开发环境下使用的域名是不固定的，微信为了防止上线出现这种情况，因此规定得使用固定的合法域名。可以通过“详情”里的设置取消这种限制。或者使用内网穿透工具 natapp(natapp.cn)，来购买域名。

5.创建详情页

原型图：



wxml:

```
<!--滚动图-->
<view id='topimg'>
  <swiper indicator-dots="{{indicatorDots}}"
    autoplay="{{autoplay}}" interval="{{interval}}" duration="{{duration}}">
    <block wx:for="{{bminfo.bmimgs}}">
      <swiper-item>
        <image id='bmimg' src="{{item}}" class="slide-image"/>
      </swiper-item>
    </block>
  </swiper>

</view>
<view id="content">
  <view>
    <text>姓名：</text><text>{{bminfo.bmname}}</text>
  </view>
   <view>
    <text>年龄：</text><text>{{bminfo.age}}</text>
  </view>
   <view>
    <text>薪水：</text><text>{{bminfo.salary}}</text>
  </view>
   <view>
```

```
    <text>教育程度：</text><text>{{bminfo.edu}}</text>
  </view>
   <view>
    <text>工作内容：</text><text>{{bminfo.job}}</text>
  </view>
</view>
<view id='bottombtn'>
  <button>立即预约</button>
</view>
```

wxss:

```
#topimg{
  width: 750rpx;
  height: 300rpx;
  background-color: pink;
}

#bmimg{
  width: 750rpx;
  height: 300rpx;
}
#content{
  margin-top: 30rpx;
}

#bottombtn{
  width: 750rpx;
  position: fixed;
  bottom: 0px;
}
```

6.路由

在保姆列表页点击保姆的详情，进入详情页。

```
<view class='d3'>
        <text class="infobtn_style" bindtap='gotodetail'>详情</text>
    </view>
```

bindtap==》绑定了点击事件。

index.js:

```
/**
   * 前往详情页
   */
  gotodetail:function(){
    wx.navigateTo({
      url: '../detail/detail',
    })
  }
})
```

常用的路由方式：

1）navigateTo

不能跳转到 tabBar 页面，也就是说跳转到在 tabBar（底部导航条）中注册的页面是无效的。

可以通过 navigateBack 返回上一层或多层的某个页面。

2）redirectTo

关闭当前页面，之前的页面就不存在了，返回不回去。不能跳转到 tabbar 页面。

7.路由时传递参数

1）在页面中绑定参数

```
<text class="infobtn_style" bindtap='gotodetail' data-bmid='{{item.bmid}}'>详情
</text>
```

2）在函数中获得页面中绑定的参数

就可以通过 data.bmid 的方式访问到页面中绑定的值：

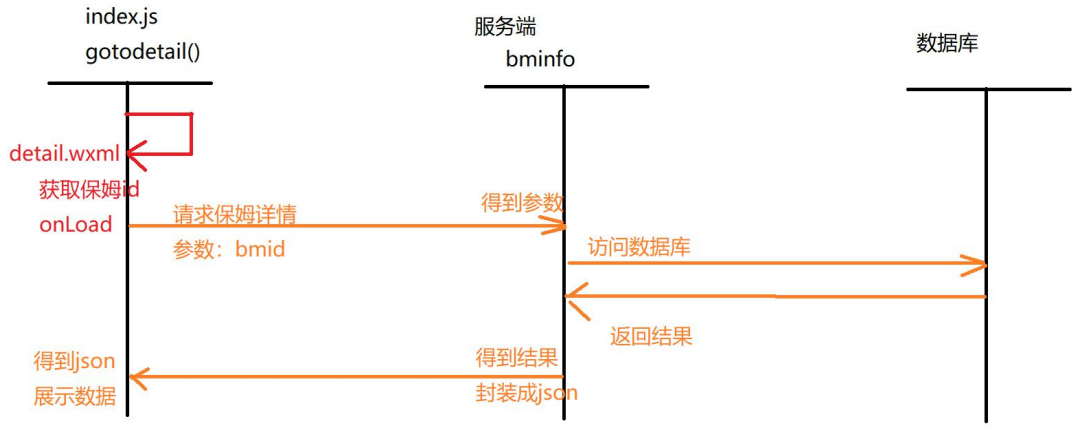
```
/**
 * 前往详情页
 */
gotodetail:function(e){
  var data = e.currentTarget.dataset;
```

3）发送请求

```
/**
 * 前往详情页
 */
gotodetail:function(e){
  var data = e.currentTarget.dataset;
  wx.navigateTo({
    url: '../detail/detail?bmid='+data.bmid
  })
}
```

8.详情页面获取后端数据

1）时序图



2）如何在 detail.js 的 onload 中获得 bmid

```
onLoad: function (options) {
  //1.获取bmid
  var bmid = options.bmid;
},
```

此时的 options 就是之前路由时传递来的参数。

index.js 页面中的:

```
/**
 * 前往详情页
 */
gotodetail:function(e){
  var data = e.currentTarget.dataset;
  wx.navigateTo({
    url: '../detail/detail?bmid='+data.bmid
  })
}
```

3）detail.js 中发送请求并得到服务端的返回的 json 数据，并设置在页面的 data 中

```
/**
   * 生命周期函数--监听页面加载
   */
  onLoad: function (options) {
    //1.获取 bmid
    var id = options.bmid;
```

```
    wx.request({
      url: 'http://localhost:8080/zybm/bminfo',
      data:{
        bmid:id
      },
      success:this.setbminfo.bind(this)
    })
  },
  setbminfo:function(res){
    this.setData({bminfo:res.data})
  },
```

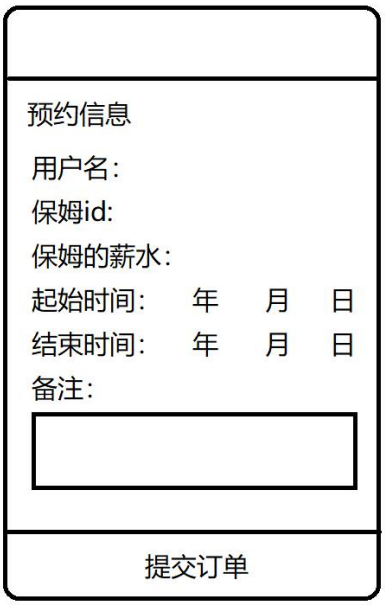
4）后端程序的实现

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

        //1.获取bmid
        String bmid = request.getParameter("bmid");
        String json =  null;

        Connection conn = null;

        PreparedStatement pstmt = null;
        ResultSet rs = null;

        //2.去数据库查询数据
        try {
            conn = MyJDBCUtil.getConnection();

            String sql = "select * from tb_bm where bmid=?";
            pstmt = conn.prepareStatement(sql);
            pstmt.setString(1, bmid);
            rs = pstmt.executeQuery();

            BaoMu bm = new BaoMu();
            if(rs.next()) {
                bm.setAge(rs.getInt("age"));
                bm.setBmid(rs.getString("bmid"));
                String imgs = rs.getString("bmimgs");
                bm.setBmimgs(imgs.split(","));
                bm.setBmname(rs.getString("bmname"));
```

```
                bm.setEdu(rs.getString("edu"));
                bm.setHome(rs.getString("home"));
                bm.setJob(rs.getString("job"));
                bm.setJob_time(rs.getInt("job_time"));
                bm.setSalary(rs.getBigDecimal("salary"));
            }

            Gson gson = new Gson();
            json = gson.toJson(bm);
        } catch (SQLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } finally {
            try {
                MyJDBCUtil.releaseResource(conn, pstmt, rs);
            } catch (SQLException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }

        // 把json响应给客户端
        response.setCharacterEncoding("utf-8");
        response.setContentType("application/json");
        response.getWriter().write(json);
    }
```


9.搭建预约页面

1）高保真图



2）如何获得当前微信用户的信息

微信开发了一个获得当前微信用户信息的接口： open-data

type 的合法值

值	说明	最低版本
groupName	拉取群名称	1.4.0
userNickName	用户昵称	1.9.90
userAvatarUrl	用户头像	1.9.90
userGender	用户性别	1.9.90
userCity	用户所在城市	1.9.90
userProvince	用户所在省份	1.9.90
userCountry	用户所在国家	1.9.90
userLanguage	用户的语言	1.9.90

```
<view>
  <open-data type="userAvatarUrl"></open-data>
  <open-data type="userNickName"></open-data>
  <open-data type="userGender"></open-data>
  <open-data type="userCity"></open-data>
</view>
```

3）预约页面的实现：wxml

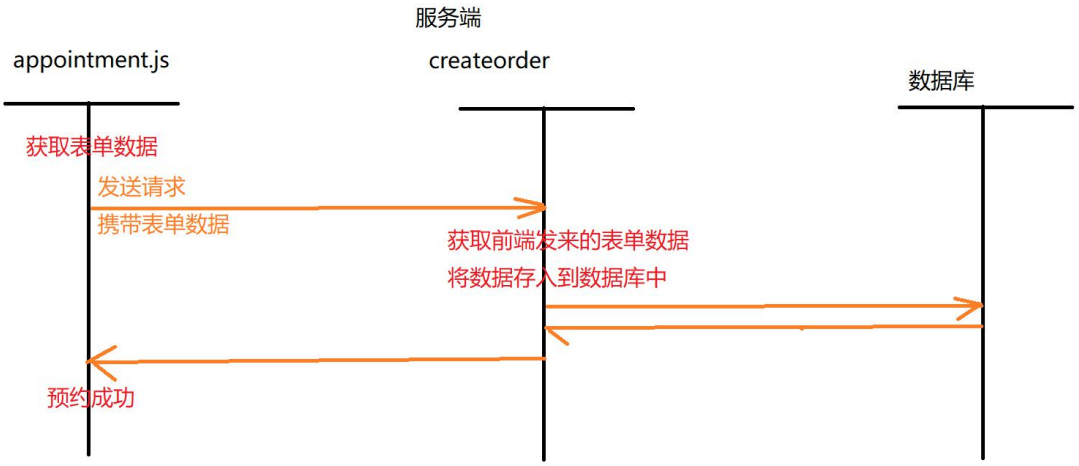
```
<!--pages/appointment/appointment.wxml-->
<view>
  <view>预约信息</view>
  <view>用户名：<open-data type="userNickName"></open-data></view>
  <view>保姆 id：<text></text></view>
  <view>保姆薪水：<text></text></view>
  <view class='timeStyle'>起始时间:</view>
  <view class='yearStyle'><input class="timeInputStyle" type='text'
value='2019'></input><text class="timeTextStyle">年</text></view>
  <view class='monthStyle'><input class="timeInputStyle" type='text'
value='5'></input><text class="timeTextStyle">月</text></view>
  <view class='dayStyle'><input class="timeInputStyle" type='text'
value='30'></input><text class="timeTextStyle">日</text></view>


  <view class='timeStyle'>结束时间:</view>
  <view class='yearStyle'><input class="timeInputStyle" type='text'
value='2019'></input><text class="timeTextStyle">年</text></view>
  <view class='monthStyle'><input class="timeInputStyle" type='text'
value='6'></input><text class="timeTextStyle">月</text></view>
  <view class='dayStyle'><input class="timeInputStyle" type='text'
value='30'></input><text class="timeTextStyle">日</text></view>
  <view>备注：</view>
  <view><input id="markInputStyle" type='text'></input></view>

  <view id='btnStyle'><button>提交订单</button></view>

</view>
```

4）预约页面的 wxss

```
.timeStyle{
  width: 185rpx;
  float:left;
}
.yearStyle{
  width: 185rpx;
  float:left;
}
.monthStyle{
  width: 185rpx;
  float:left;
}
.dayStyle{
  width: 185rpx;
  float:left;
}
.timeInputStyle{
  width: 100rpx;
  border: 1rpx solid lightgray;
  float:left;
}
.timeTextStyle{
  margin-left: 10rpx;
  width: 30rpx;
  float:left;
}

#markInputStyle{
  height:300rpx;
  border: 1rpx solid lightgray;
}

#btnStyle{
  width:100%;
  position: fixed;
  bottom: 0rpx;
}
```

10.实现预约页面的功能

1）时序图



2）在 js 中如何获取 wxml 页面中的表单数据

（1）在 wxml 中设置 form 表单标签，form 表单中的 input 就是具体的表单数据，因此 input 元素需要 name 属性，并且有修改 button 的属性为 submit

```
<form bindsubmit='createorder'>
<input class="timeInputStyle" type='text' value='2019' name="startYear"></input>
<button form-type='submit'>提交订单</button>
```

```
<view>
  <form bindsubmit='createorder'>
    <view>预约信息</view>
    <view>用户名：<open-data type="userNickName"></open-data></view>
    <view>保姆 id：<text></text></view>
    <view>保姆薪水：<text></text></view>
    <view class='timeStyle'>起始时间:</view>
    <view class='yearStyle'><input class="timeInputStyle" type='text' value='2019'
name="startYear"></input><text class="timeTextStyle">年</text></view>
    <view class='monthStyle'><input class="timeInputStyle" type='text' value='5'
name="startMonth"></input><text class="timeTextStyle">月</text></view>
```

```
    <view class='dayStyle'><input class="timeInputStyle" type='text' value='30'
name="startDay"></input><text class="timeTextStyle">日</text></view>


    <view class='timeStyle'>结束时间:</view>
    <view class='yearStyle'><input class="timeInputStyle" type='text' value='2019'
name="endYear"></input><text class="timeTextStyle">年</text></view>
    <view class='monthStyle'><input class="timeInputStyle" type='text' value='6'
name="endMonth"></input><text class="timeTextStyle">月</text></view>
    <view class='dayStyle'><input class="timeInputStyle" type='text' value='30'
name="endDay"></input><text class="timeTextStyle">日</text></view>
    <view>备注：</view>
    <view><input id="markInputStyle" type='text'></input></view>

    <view id='btnStyle'><button form-type='submit'>提交订单</button></view>
  </form>
</view>
```

（2）js 中获得表单数据：

e.detail.value

```
createorder:function(e){
    var d = e.detail.value;
    //如何获得表单数据
    console.log(d);
  }
```

3）用户名如何作为表单数据

wxml 中的用户名是通过<open-data type="userNickName"></open-data>这个方式得到。但是这个方式并不会将此用户名数据作为表单数据。因此我们得通过 js 的方式来获得用户名。

（1）将 wxml 中的用户名的获取方式修改成如下：

```
<view>用户名：<input type="text" value="{{user.nickName}}"
name="username"></input></view>
```

（2）在 js 中提供 data

当页面加载时，通过 wx.getUserInfo 来得到用户信息。

```
/**
   * 生命周期函数--监听页面加载
```

```
  */
  onLoad: function (options) {
    //调用内置的函数 getUserInfo 来得到当前微信用户的信息，这些信息封装在 res 里
    wx.getUserInfo({
      //经获得的用户信息传递给 setuserinfo 方法
      success:this.setuserinfo.bind(this)
    })
  },
  setuserinfo:function(res){
    this.setData({user:res.userInfo})
  },
```

我们发现，要想获得用户信息。得先经过用户的授权。因此，我们在来到此预约页面之前，先进行用户的授权：

```
<view id='bottombtn'>
  <button open-type='getUserInfo'
bindgetuserinfo='gotoappointment'>立即预约</button>
</view>
```

其中 open-type='getUserInfo' 指的就是当点击此按钮时进行用户授权。

bindgetuserinfo='gotoappointment' 当授权完成后调用'gotoappointment'函数

4）如何在预约页面获得保姆 id 和保姆薪水

通过保姆信息页面，点击立即预约时获得保姆 id 和保姆薪水

在执行'gotoappointment'函数时就能获得绑定的两个 data-***的键值对

```
<view id='bottombtn'>
  <button open-type='getUserInfo' bindgetuserinfo='gotoappointment'
data-bmid='{{bminfo.bmid}}' data-salary='{{bminfo.salary}}'>立即预约
</button>
</view>
```

js: 通过 e 参数获取 wxml 中当前按钮中的 data-***属性的键值对 s

```
gotoappointment:function(e){
    //获取 wxml 中当前按钮中的 data-***属性的键值对 s
    var d = e.currentTarget.dataset;
   // d.bmid;
    //d.salary
```

```
    wx.navigateTo({
      url: '../appointment/appointment?bmid='+d.bmid+'&salary='+d.salary
    })
  }
```

在 appointment.js 中获取之前页面发送来的参数
options 里就有之前页面发送来的键值对

```
onLoad: function (options) {
    //options 对象中就有前面转过来的键值对
    // url: '../appointment/appointment?bmid='+d.bmid+'&salary='+d.salary
    this.setData({
      bmid:options.bmid,
      salary:options.salary
    })


    //调用内置的函数 getUserInfo 来得到当前微信用户的信息，这些信息封装在 res 里
    wx.getUserInfo({
      //经获得的用户信息传递给 setuserinfo 方法
      success:this.setuserinfo.bind(this)
    })
  },
```

再修改 appointment.wxml

```
<view>保姆 id：<input type='text' value='{{bmid}}' name='bmid'></input></view>
<view>保姆薪水：<input type='text' value='{{salary}}' name='salary'></input></view>
```

5）提交订单访问后端接口实现订单的生成

appointment.js:

```
  createorder:function(e){
    //获得表单数据
    var d = e.detail.value;

    wx.request({
      url: 'http://localhost:8080/zybm/createorder',
      data:d,
      success:this.getresult.bind(this)
```

```
    })
  },
  getresult:function(res){
    //显示一个提示框:
    wx.showToast({
      title: res.data.msg,//提示框内容
      duration:3000//持续时间
    });
    //设置定时器: 4 秒后回到列表页面
    setTimeout(function(){
      wx.navigateBack({
        delta:2
      })
    },4000)
  }
```

后端代码:

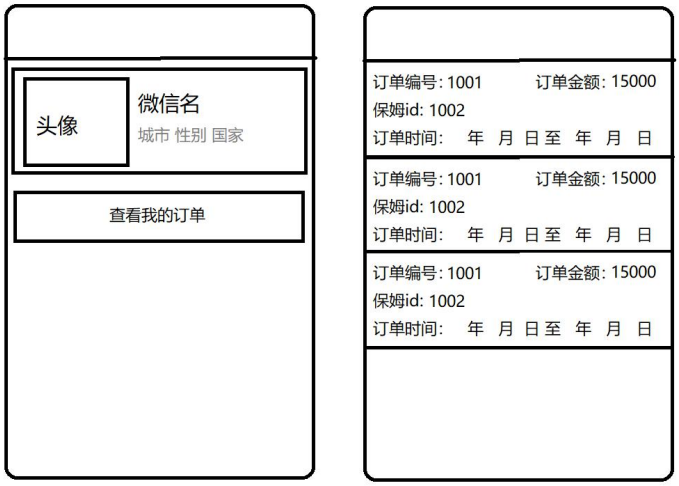
```
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException,
IOException {

        request.setCharacterEncoding("utf-8");

        String bmid = request.getParameter("bmid");
        String username = request.getParameter("username");
        String salary = request.getParameter("salary");
        String startYear = request.getParameter("startYear");
        String startMonth = request.getParameter("startMonth");
        String startDay = request.getParameter("startDay");
        String endYear = request.getParameter("endYear");
        String endMonth = request.getParameter("endMonth");
        String endDay = request.getParameter("endDay");
        String mark = request.getParameter("mark");

        //jdbc
        String sql = "insert into tb_order values(?,?,?,?,?,?)";
        Connection conn = null;
        PreparedStatement pstmt = null;

        try {
            conn = MyJDBCUtil.getConnection();
            pstmt = conn.prepareStatement(sql);
```

```
            pstmt.setString(1, UUID.randomUUID().toString());
            pstmt.setString(2, bmid);
            pstmt.setString(3,username);

            //如何计算月的数量
            int count = getMonths(startYear,startMonth,startDay,endYear,endMonth,endDay);

            BigDecimal amount = new BigDecimal(Double.parseDouble(salary)*count);
            pstmt.setInt(4, count);
            pstmt.setBigDecimal(5, amount);
            pstmt.setString(6, mark);



            int rows = pstmt.executeUpdate();
            //用来封装结果的对象
            ResultDto rd = new ResultDto();
            if(rows>0) {
                //插入成功
                rd.setResult(true);
                rd.setMsg("约成功啦！");
            }else {
                //插入失败
                rd.setResult(false);
                rd.setMsg("提交失败");
            }
            //向前端发送json {result: true,msg:"成功"}
            Gson gson = new Gson();
            String json = gson.toJson(rd);
            response.setCharacterEncoding("utf-8");
            response.setContentType("application/json");
            response.getWriter().write(json);


        } catch (SQLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } finally {
            try {
                MyJDBCUtil.releaseResource(conn, pstmt, null);
            } catch (SQLException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }
```

```
    }

    /**
     * 根据时间获得月的总数
     * @param startYear
     * @param startMonth
     * @param startDay
     * @param endYear
     * @param endMonth
     * @param endDay
     * @return
     */
    private int getMonths(String startYear, String startMonth, String startDay, String endYear, String
endMonth,
            String endDay) {

        int count = 0;
        //把string=>int
        int sy = Integer.parseInt(startYear);
        int sm = Integer.parseInt(startMonth);
        int sd = Integer.parseInt(startDay);
        int ey = Integer.parseInt(endYear);
        int em = Integer.parseInt(endMonth);
        int ed = Integer.parseInt(endDay);

        if(ey-sy==0) {
            count = em-sm;
            if(ed-sd>0) {
                count+=1;
            }
        }else {
            count=(ey-sy)*12;
            count += em-sm;
            if(ed-sd>0) {
                count+=1;
            }
        }
        return count;
    }
```

11."个人中心"页面的搭建

1）高保真图



2）开发个人中心页面

wxml:

```
<view id='main'>
  <view id='v1'>
    <view id='i1'>
      <open-data id='userimg' type='userAvatarUrl'></open-data>
    </view>
    <view id='i2'>

      <view>
        <open-data type='userNickName'></open-data>

      </view>
       <view id='ii2'>
        <open-data type='userGender'></open-data>-
        <open-data type='userCity'></open-data>-
        <open-data type='userCountry'></open-data>
      </view>

    </view>
```

```
  </view>
  <view id='v2'>
    <button id='v2btn'>查看我的订单</button>

  </view>
</view>
```

wxss:

```
#main{
  background-color: #f0f0f5;
  position: fixed;
  top: 0rpx;
  left:0rpx;
  right: 0rpx;
  bottom: 0rpx;
}
#v1{
  background-color: white;
  width: 734rpx;
  height: 220rpx;
  margin:5rpx 8rpx 0rpx 8rpx;
  border-radius: 10rpx;
  box-shadow: darkgray 0rpx 6rpx 10rpx 0rpx;
}


#i1{
  float:left;
  width: 180rpx;
  height: 160rpx;
  margin: 30rpx 0rpx 30rpx 30rpx;

}

#i2{
  width:470rpx;
  margin: 60rpx 0rpx 50rpx 30rpx;
  float:left;
  color:#333;
}
#ii2{
```

```
  margin-top:20rpx;
  font-size: 30rpx;
  color:#999;
}

#v2{
  background-color: white;
  width: 734rpx;
  height: 150rpx;
  margin:15rpx 8rpx 0rpx 8rpx;
  border-radius: 10rpx;

}
#v2btn{
  height: 150rpx;
  padding-top:30rpx;
}
```

3）开发订单页面

wxml:

```
<view class='single'>
  <view class='i1'>
    <view class='ii1'>
      <text class='text1'>订单编号:</text>
      <text class='text1'>1001</text>
    </view>
    <view class='ii2'>
      <text class='text2'>订单金额:</text>
      <text class='text2'>7500.0</text>
    </view>
  </view>
  <view class='i2'>
    <text class='text1'>保姆编号:</text>
    <text class='text1'>1001</text>
  </view>
```

```
  <view class='i3'>
    <text class='text1'>订单时间:</text>
    <text class='text1'>30</text>
    <text class='text1'>个月</text>
  </view>
</view>
<view class='single'>
  <view class='i1'>
    <view class='ii1'>
      <text class='text1'>订单编号:</text>
      <text class='text1'>1001</text>
    </view>
    <view class='ii2'>
      <text class='text2'>订单金额:</text>
      <text class='text2'>7500.0</text>
    </view>
  </view>
  <view class='i2'>
    <text class='text1'>保姆编号:</text>
    <text class='text1'>1001</text>
  </view>
  <view class='i3'>
    <text class='text1'>订单时间:</text>
    <text class='text1'>30</text>
    <text class='text1'>个月</text>
  </view>
</view>
```

wxss:

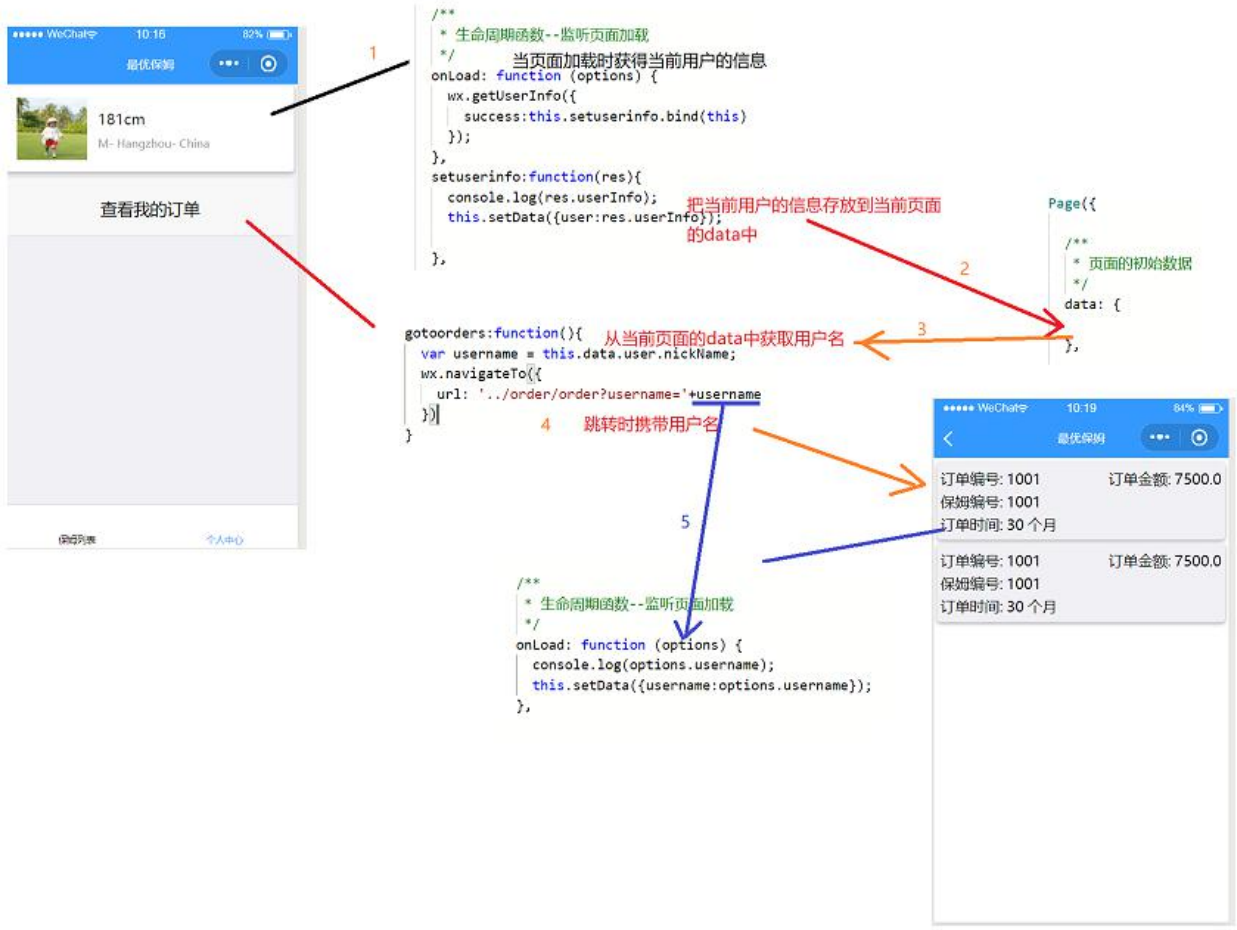
```
.single{
  background-color: #f0f0f5;
  width: 734rpx;
  height: 200rpx;
  margin:10rpx 8rpx 0rpx 8rpx;
  border-radius: 10rpx;
  box-shadow: darkgray 0rpx 6rpx 10rpx 0rpx;
 padding-top: 10rpx;
 box-sizing: border-box;
}
.ii1{
  float: left;
```

```
}
.ii2{
  float:right;

}
.text1{
  margin-right: 10rpx;
}
.text2{
  margin-right: 10rpx;
}
.i1{
  margin-top:10rpx;
  margin-left:10rpx;
}
.i2{
  margin-left:10rpx;
  clear: both;
  padding-top:10rpx;
}
.i3{

  padding-top:10rpx;
  padding-left:10rpx;
}
```

4）个人中心跳转到订单页面

要解决的问题：

如何在跳转时携带用户名



self.wxml

```
<view id='v2'>
    <button id='v2btn' bindtap='gotoorders' >查看我的订单</button>

  </view>
```

self.js

```
/**
   * 生命周期函数--监听页面加载
   */
  onLoad: function (options) {
    wx.getUserInfo({
      success:this.setuserinfo.bind(this)
    });
  },
  setuserinfo:function(res){
    console.log(res.userInfo);
    this.setData({user:res.userInfo});
```

```
  },
gotoorders:function(){
    var username = this.data.user.nickName;
    wx.navigateTo({
      url: '../order/order?username='+username
    })
  }
```

order.js:

```
onLoad: function (options) {
    console.log(options.username);
    this.setData({username:options.username});
  },
```

5）小程序发送请求，获得服务端数据并展示

order.js

```
/**
   * 生命周期函数--监听页面加载
   */
  onLoad: function (options) {
    //获得此页面中的用户名
    var un = options.username;
    //发送请求。获得当前用户的所有的订单
    wx.request({
      url: 'http://localhost:8080/zybm/orderlist',
      data:{
        username:un
      },
      success:this.setorderlist.bind(this)
    })
  },
  setorderlist:function(res){
    this.setData({orderlist:res.data});

  },
```

order.wxml:

```
<!--pages/order/order.wxml-->
```

```
<view wx:for='{{orderlist}}'>
  <view class='single'>
    <view class='i1'>
      <view class='ii1'>
        <text class='text1'>订单编号:</text>
        <text class='text1'>{{item.oid}}</text>
      </view>

    </view>

    <view class='i2'>
      <text class='text1'>保姆编号:</text>
      <text class='text1'>{{item.bmid}}</text>
    </view>
    <view class='i3'>
      <text class='text1'>订单时间:</text>
      <text class='text1'>{{item.ordertime}}</text>
      <text class='text1'>个月</text>
    </view>
    <view class='i3'>
       <text class='text1'>订单金额:</text>
       <text class='text1'>{{item.amount}}</text>
      </view>
  </view>
</view>
```

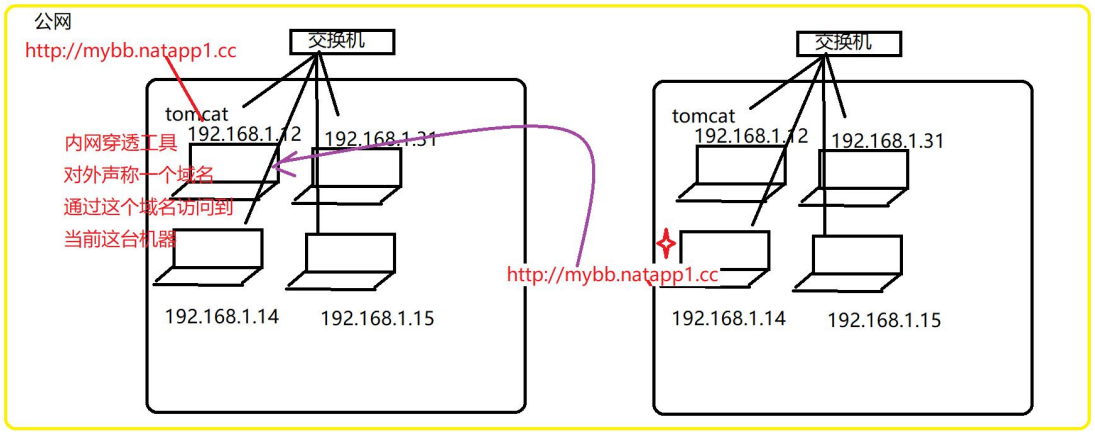
后端的 orderlist:

```
protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        request.setCharacterEncoding("utf-8");
        //获得客户端发送来的用户名
        String username = request.getParameter("username");
        Connection conn = null;
        PreparedStatement pstmt = null;
        ResultSet rs = null;
        try {
            conn = MyJDBCUtil.getConnection();
            String sql = "select * from tb_order where usernickname=?";
            pstmt = conn.prepareStatement(sql);
            pstmt.setString(1, username);
            rs = pstmt.executeQuery();
            List<BmOrder> list = new ArrayList<>();
            while(rs.next()) {
```

```
                //封装对象
                BmOrder bo = new BmOrder();
                bo.setOid(rs.getString("oid"));
                bo.setBmid(rs.getString("bmid"));
                bo.setUsernickname(rs.getString("usernickname"));
                bo.setMark(rs.getString("mark"));
                bo.setAmount(rs.getBigDecimal("amount"));
                bo.setOrdertime(rs.getInt("ordertime"));
                //存入到集合
                list.add(bo);


            }

            //生成json再发送给客户端
            Gson gson = new Gson();
            String json = gson.toJson(list);
            response.setCharacterEncoding("utf-8");
            response.setContentType("application/json");
            response.getWriter().write(json);


        } catch (SQLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } finally {
            try {
                MyJDBCUtil.releaseResource(conn, pstmt, rs);
            } catch (SQLException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }

    }
```


七、内网穿透工具的使用　natapp

1.内网穿透的概念



2.如何购买 natapp 的内网穿透服务

1）下载 natapp 客户端

进入到 natapp.cn 下载

客户端下载

版本: 2.3.9 更新日志

Windows	32位 64位
Mac OS X	32位 64位
Linux	32位 64位
Linux/ARM 树莓派请下载32位	32位 64位
Linux/MIPS(LE) 绝大部分路由器	32位 32位(压缩版) 64位 LE 32位 LE 32位(压缩版) LE 64位

2）注册

3）登录

4）购买隧道并购买二级域名

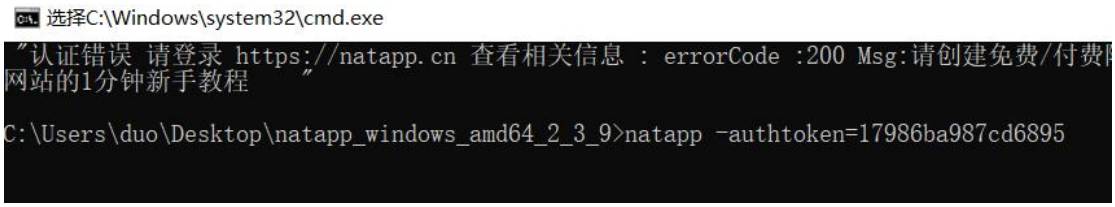
5）获取 authtoken

6）在本机运行 natapp 客户端

解压压缩包后双击运行

7）输入以下命令，该命令中包含你的 token

natapp -authtoken=17986ba987cd6895



当看到这个界面。就表示成功啦！此时可以使用域名啦！