

类别 研发文档

编号 QXSK-D-DG

版本 1.0

密级 公开

全息时空运营服务平台

开发者指南

编制单位: 江苏凌比特微电子技术有限公司

编 制 人: 任 珊 虹

编制日期: 2018 年 5 月 22 日

修订记录

时间	作者	版本	说明
2015. 9. 18	任珊虹	0.1	初始版本
2015. 12. 4	任珊虹	0.2	开发环境的安装与配置
2015. 12. 4	任珊虹	0.3	打包与安装指南
2018. 5. 22	任珊虹	1.0	架构与编程

对于文档中的错误或者描述不清的地方，请联系作者，谢谢！

目 录

1 概述.....	9
1.1 平台资源.....	9
1.2 本文档的内容.....	9
1.3 其它文档.....	10
2 系统架构.....	11
2.1 系统的目标.....	11
2.2 系统的技术结构图.....	12
2.3 实现系统的技术	13
2.4 技术和工具的选择.....	13
2.4.1 背景和原则.....	13
2.4.2 CS (客户端应用) 还是 BS (浏览器应用) ?	14
2.4.3 NoSQL 还是 Mysql.....	14
2.4.4 云还是 Java EE 7?	15
2.4.5 Struts+Spring+Hiberbate 还是 JSF+CDI+JPA ?	15
2.4.6 SOAP 还是 RESTful Web Service ?	16
2.4.7 轮询、MQTT 还是 WebSocket?	16
2.4.8 Eclipse、IntelliJ IDEA、还是 NetBeans?	16
2.4.9 PrimeFaces、ICEfaces、还是 RichFaces?	17
2.4.10 JBoss 还是 GlassFish?.....	18
2.4.11 自带的 JMS provider 还是 ActiveMQ?	19
2.4.12 BIRT 还是 JasperReports?	21
2.4.13 高德地图还是百度地图?	21
2.4.14 商用 SSL 证书还是免费 SSL 证书?	21
2.4.15 anaconda、Spyder、还是 pycharm?	21
2.4.16 Eclipse 还是 Android Studio?.....	21
2.4.17 HttpClient、HttpURLConnection、还是 OkHttp.....	22
2.4.18 Objective-C 还是 Swift.....	22
2.5 系统的特点.....	23
2.5.1 信息安全	23
2.5.2 按需定制	23
2.5.3 管理/使用方式.....	24
2.5.4 北斗终端的接入方法.....	24
2.5.5 非北斗的数据终端的接入方法.....	26
2.6 一些误读及其解释.....	26
2.7 系统的伸缩.....	28
2.8 跟踪分布式技术.....	29
3 平台服务器的开发.....	31
3.1 开发资源.....	31
3.2 调试环境.....	31
3.3 系统的“大表”	32
3.4 数据库设计环境 MySQL Workbench	33
3.4.1 资源地址.....	33
3.4.2 设置数据库连接参数.....	33
3.4.3 编辑 EER (Enhanced ER)	34
3.4.4 编辑索引、外键、触发器.....	35
3.4.5 编辑存储过程.....	36
3.4.6 生成数据库.....	37
3.5 数据管理.....	39
3.5.1 数据表分区	39

3.5.2 “最终用户数据权限表”	39
3.5.3 配置主从数据库.....	40
3.5.4 常用 mysql 命令 - win.....	40
3.5.5 常用 mysql 命令 - Linux.....	41
3.5.6 常用 mysql 语句.....	42
3.6 JavaEE 开发环境 NetBeans.....	45
3.6.1 资源地址.....	45
3.6.2 下载插件.....	45
3.6.3 改变界面语言.....	46
3.6.4 修改默认 JDK.....	46
3.6.5 连接数据库.....	47
3.6.6 管理服务器.....	49
3.7 JavaEE 项目.....	51
3.7.1 开发示例.....	51
3.7.2 注册外部资源.....	51
3.7.3 使用 Git 支持.....	51
3.7.4 使用 maven 支持.....	51
3.7.5 选择 MVC 框架.....	52
3.7.6 设置系统标识和版本.....	53
3.7.7 下载依赖.....	53
3.8 开发 Entity Bean.....	54
3.8.1 利用 Wizard 自动生成 Entity Bean.....	54
3.8.2 修正复合键.....	56
3.9 开发 MVC 文件.....	58
3.9.1 替换 PrimeFaces 模板.....	58
3.9.1.1 "Controller.java"的修改.....	58
3.9.1.2 "Create.xhtml" 和"Edit.xhtml"的修改.....	58
3.9.1.3 "List.xhtml" 的修改.....	59
3.9.1.4 "View.xhtml" 的修改.....	59
3.9.2 利用 Wizard 自动生成 MVC 各层文件.....	59
3.9.3 扩展 Session Bean.....	61
3.9.4 扩展 JSF controller.....	62
3.9.5 编辑 xhtml 文件.....	63
3.9.6 Xhtml 的布局模板.....	64
3.9.7 数据的 Lazy Loading.....	65
3.9.8 定制本地化文件.....	67
3.9.9 定制数据导出器.....	69
3.10 开发 RESTful Web Services.....	70
3.10.1 配置 Web Services 的环境.....	70
3.10.2 用 JAX-RS 注释定义 Web Services 的属性.....	70
3.10.3 用 JAXB 注释定义 Web Services 的对象.....	71
3.11 开发 WebSocket 服务器端.....	72
3.11.1 基础类 DataSocket.java.....	72
3.11.2 WebSocket 类.....	72
3.11.3 向客户端发送数据.....	73
3.12 开发 WebSocket 客户端-Javascript.....	74
3.13 服务器的全局数据.....	75
3.13.1 系统常量.....	75
3.13.2 系统变量.....	75
3.13.3 系统参数.....	75
3.13.4 系统工具箱.....	75
3.13.5 服务器的状态监听器.....	76

3.13.6 会话监听器.....	76
3.14 服务器的初始化.....	77
3.14.1 服务器的启动步骤.....	77
3.14.2 初始化任务的调度线程.....	77
3.14.3 初始化任务的子线程.....	78
3.14.4 周期执行的子线程.....	79
3.14.5 打开端口.....	79
3.15 服务器的安全.....	80
3.15.1 加密算法.....	80
3.15.2 资源的过滤器.....	81
3.15.3 xhtml 的认证过滤器.....	81
3.15.4 xhtml 的授权过滤器.....	82
3.15.5 公共 WebService 的认证过滤器.....	82
3.15.6 面向用户的 WebService 的认证过滤器.....	82
3.15.7 面向终端的 WebService 的认证过滤器.....	82
3.15.8 安全链接 SSL.....	82
3.16 读写分离.....	83
3.16.1 方案的选择.....	83
3.16.2 定义持久层单元.....	83
3.16.3 定义“基础 EJB”.....	84
3.16.4 定义“持久层单元的 EJB”.....	85
3.17 “Stale Read”问题.....	86
3.17.1 问题是什么？.....	86
3.17.2 观察到的现象.....	86
3.17.3 官方的解决办法.....	87
3.17.4 我的解决办法.....	87
3.18 事务.....	88
3.18.1 事务界定.....	88
3.18.2 事务属性.....	88
3.19 报表设计环境 Jaspersoft Studio.....	90
3.19.1 资源地址	90
3.19.2 修改 workspace 路径.....	90
3.19.3 导入报表模板文件.....	90
3.19.4 配置数据源.....	92
3.19.5 编辑报表模板源文件 jrxml.....	93
3.19.6 编译报表模板文件 jasper.....	96
3.20 生成统计报表.....	97
3.20.1 引用报表模板 Jasper 文件.....	97
3.20.2 初始化报表对象.....	97
3.20.3 生成统计数据.....	98
3.20.4 生成报表.....	100
3.20.5 实时报表还是定时报表？.....	101
3.21 应用百度地图.....	102
3.21.1 申请浏览器端的 AK	102
3.21.2 在网页中显示地图.....	103
3.21.3 在 https 链接的网页中显示百度地图.....	103
3.21.4 调用地址转换服务.....	104
3.22 系统的日志.....	106
3.22.1 代码记录的 log4j2 日志.....	106
3.22.2 服务器记录的日志.....	108
3.22.3 系统的审计日志.....	108
3.23 自动化测试.....	110

3.23.1 工具 selenium.....	110
3.23.1.1 资源地址.....	110
3.23.1.2 关于 Selenium IDE.....	110
3.23.1.3 关于 WebDriver.....	112
3.23.2 编程环境 pycharm.....	112
3.23.2.1 资源地址.....	112
3.23.2.2 虚拟环境和 python 解释器的版本.....	112
3.23.2.3 管理 Python 包和 Python 库.....	113
3.23.3 Selenium 编程.....	114
3.23.3.1 配置 Python 环境.....	114
3.23.3.2 示例代码.....	114
3.23.3.3 常用 selenium 代码.....	115
3.23.3.4 selenium 三种等待方式.....	116
4 安卓应用的开发.....	117
4.1 Android Studio.....	117
4.1.1 资源地址.....	117
4.1.2 HTTP proxy.....	117
4.1.3 更新 Gradle.....	117
4.2 应用的全局数据.....	118
4.2.1 应用的常量.....	118
4.2.2 应用的参数.....	118
4.2.3 应用的工具箱.....	118
4.3 应用的初始化.....	119
4.3.1 Application 类.....	119
4.3.2 应用的封面.....	119
4.4 应用百度地图.....	120
4.4.1 申请 Android 端的 AK.....	120
4.4.2 在 AndroidManifest 中添加开发密钥.....	120
4.4.3 百度地图的 Android SDK.....	121
4.4.4 导入 SDK.....	121
4.4.5 初始化 SDK.....	121
4.4.6 在布局中添加地图.....	122
4.4.7 在代码中控制地图.....	122
4.5 SQLite3.....	123
4.5.1 数据库辅助类 DatabaseHandler.....	123
4.5.2 数据表的模板 Table.....	124
4.5.3 数据表.....	124
4.5.4 显式调用事务.....	125
4.5.5 写数据库的时机.....	126
4.6 调用网络服务.....	127
4.6.1 通用的网络服务对象 WebServiceRequestObject.....	127
4.6.2 http 请求.....	128
4.6.3 发送网络服务请求.....	129
4.7 调用 WebSocket.....	130
4.7.1 继承 WebSocketListener.....	130
4.7.2 发送 WebSocket 请求.....	130
4.7.3 发送心跳.....	131
4.7.4 关闭 WebSocket 连接.....	131
4.8 定位服务.....	132
4.8.1 定位监听器.....	132
4.8.2 新位置数据的处理.....	132
4.8.3 算法 isBetterLocation.....	133

全息时空运营服务平台 开发者指南	
4.8.4 判断服务是否正在运行.....	133
4.9 蓝牙服务.....	134
4.9.1 service 类.....	134
4.9.2 在子线程中读写蓝牙串口.....	134
4.9.3 发送广播.....	135
4.9.4 接收广播.....	136
4.10 编辑本地化文件.....	137
4.11 编译、构建、打包.....	138
4.11.1 Android 的 65k 问题.....	138
4.11.2 “Failed to resolve: com.android.support:multidex:1.0.2”.....	138
4.11.3 “is not translated in "en" (English) [MissingTranslation]”.....	139
4.11.4 V1(Jar Signature)和 V2(Full APK Signature).....	139
5 iOS 应用的开发.....	140
5.1 开发者账号.....	140
5.1.1 企业账户还是个人账户？.....	140
5.1.2 注册账户.....	141
5.1.3 管理成员.....	141
5.1.4 管理账户.....	142
5.1.5 管理应用.....	143
5.1.6 App 分析数据.....	145
5.2 配置 Xcode.....	148
5.2.1 设置开发账户.....	148
5.2.2 设置声音.....	149
5.2.3 设置编辑选项.....	150
5.2.4 定制快捷键.....	151
5.3 管理项目.....	152
5.3.1 项目管理文件.....	152
5.3.2 设置目标属性.....	153
5.3.2.1 “General”.....	153
5.3.2.2 “Info”.....	154
5.3.2.3 “Build Settings”.....	155
5.3.2.4 “Build Phases”.....	155
5.3.3 设置项目属性.....	156
5.4 国际化.....	158
5.5 应用地图.....	160
5.5.1 申请 AK.....	160
5.5.2 导入地图库.....	160
5.5.3 初始化地图 SDK.....	161
5.5.4 View 中的地图.....	162
5.6 故事板（StoryBoard）.....	163
5.6.1 应用的故事板	163
5.6.2 功能按钮	164
5.6.3 缩放故事板	164
5.6.4 场景（Scene）和联系（Segues）.....	165
5.7 调试.....	168
5.7.1 模拟器.....	168
5.7.1.1 运行中的模拟器.....	168
5.7.1.2 切换模拟器	169
5.7.1.3 手动打开模拟器.....	170
5.7.1.4 缩放模拟器.....	170
5.7.2 截屏.....	170
5.7.3 管理模式（Schema）	170

5.8 Swift 编码	172
5.8.1 定制组件	172
5.8.2 应用的常量	173
5.8.3 应用的变量	173
5.8.4 应用的参数	174
5.8.5 “收起键盘”	174
5.8.6 网络请求	175
5.9 发布 iOS 应用	176
5.9.1 安装发布证书	176
5.9.2 在 iTunes Connect 中添加版本	176
5.9.3 上传应用	177
5.9.4 处理反馈	178
5.9.5 搜索 App Store	179
附录 A 其它与开发相关的信息	180
A.1 进度报告	180
A.2 功能“开发”	183
A.2 功能“测试”	183
A.3 用户反馈	184
A.3 版本历史	185
A.4 统计报告	186

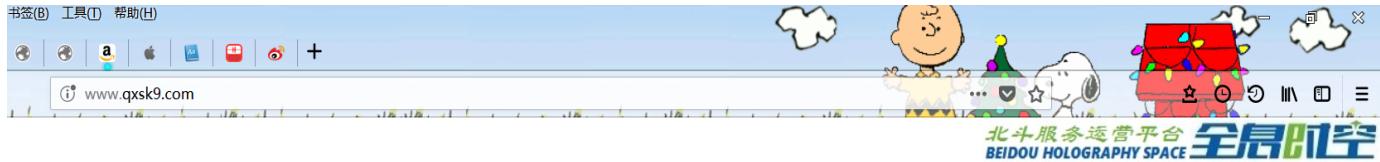
1 概述

1.1 平台资源

全息时空北斗服务运营平台对外服务的网址如下：

<http://www.qxsk9.com>

用户可以在此地址访问和下载平台的软件和文档。如下图：



资源	最新版本	版本时间	运行环境	访问/获得方式	文档
网页客户端 (服务平台功能)	v5.3	2018-04-14	支持HTML5的浏览器	进入 (建议Firefox和Safari) 进入(安全链接) (startssl证书, 已失效)	系统简介 一些错误及其解释 《服务平台用户手册》v3.0.2018-04-06 《运行环境的安装与配置手册》v4.2.2018-04-21 《开发者指南》 (未完, 4-23更新) 《用户反馈》 《版本历史》 2018-04-14
安卓应用 (三合一) 客户端、本地定位、北斗伴侣	v4.1	2018-03-23	Android 4.0.2以上	 下载 或者扫描二维码: <small>(在微信里扫描或点击均无效)</small> <small>(若用户已安装2.0以前的版本, 则须先卸载旧版再安装新版)</small>	《用户手册》
网络服务接口	v2.1	2018-03-28	RESTful(XML或JSON)	编程	《网络服务协议》
WebSocket编程接口	v1.0	2018-03-27	浏览器/iOS/Android	编程	《WebSocket接口》
苹果客户端	v1.2	2016-12-16	iOS 8.0以上	“北斗视野” (目前在苹果商店已下架)	《用户手册》
北斗伴侣增强版 (北斗终端应用)	v1.3a	2016-11-05	安卓4.0.2以上 (配合E200使用)	 下载 或者扫描二维码: <small>(在微信里扫描或点击均无效)</small>	《用户手册》

©该网站版权归江苏凌比特微电子技术有限公司所有 备案/许可证号: [苏ICP备16007070号-1](#)

1.2 本文档的内容

本文档给出平台开发的指导和经验，包括五个部分：概述、系统架构、平台服务器的开发、安卓应用的开发、iOS应用的开发、应用百度地图。本文档的内容与其它文档可能有部分重复，是为了描述语境的完整。

本文档假设读者有能力依照官方文档进行编码开发，因而不介绍基础知识，而只是描述作者认为需要特别说明的内容。

由于系统的开发和运行均基于开源软件，本文档介绍的内容并没有商业秘密，故可以公开。

本文档的下载地址为：

<http://www.qxsk9.com/QXSK-DevGuide.pdf>

1.3 其它文档

用户可以参考以下文档以了解更多的内容：

《全息时空北斗服务运营平台-用户手册》

<http://www.qxsk9.com/QXSK-Platform-UserGuide.pdf>

《全息时空北斗视野-安卓应用-用户手册》

<http://www.qxsk9.com/BeidouView-android.pdf>

《全息时空北斗视野-苹果客户端-用户手册》

<http://www.qxsk9.com/BeidouView-ios.pdf>

《全息时空北斗服务-网络服务协议》

<http://www.qxsk9.com/QXSK-Platform-WebService.pdf>

《全息时空北斗服务-WebSocket 接口》

<http://www.qxsk9.com/QXSK-Platform-WebSocket.pdf>

《北斗伴侣手机应用软件使用说明书》

<http://www.qxsk9.com/QE200.pdf>

《2015-5-25-业务发展可行性报告研究报告与技术方案》（内部文档）

2 系统架构

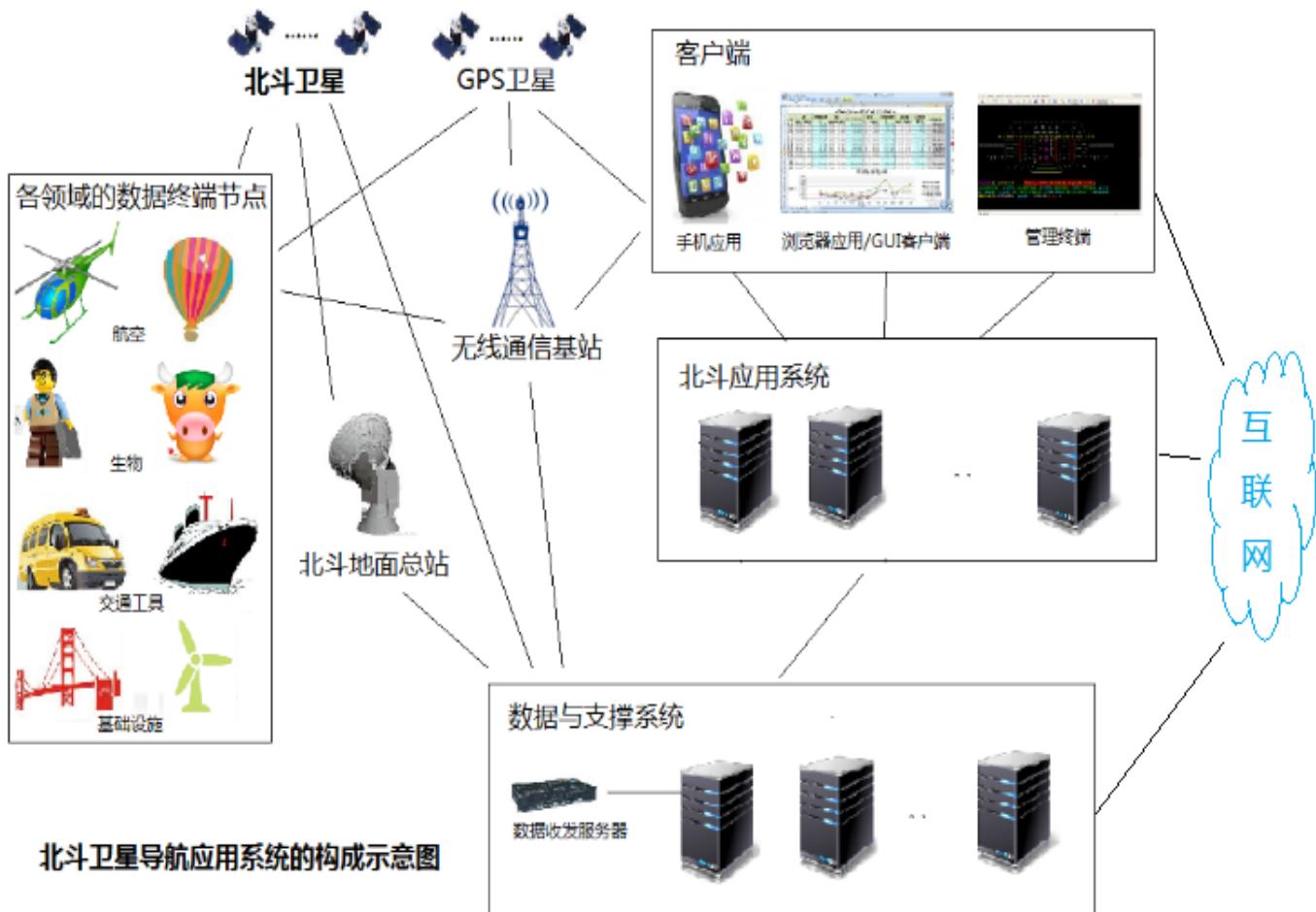
2.1 系统的目标

全息时空北斗运营服务平台支持北斗数据网与公共数据网的双向全联通：

- **北斗数据终端**通过北斗数据通道向数据中心报告位置信息、发出SOS警报、收发北斗短消息。
- **普通数据终端**通过公共数据网向数据中心报告位置信息、发出SOS警报、传输业务数据。
- 用户通过**网页客户端**或者**移动客户端**来管理终端、监视终端的位置/轨迹/警报、与北斗数据终端或者普通数据终端互发消息。
- 第三方应用通过网络服务和**WebSocket**编程接口来使用平台数据、管理北斗数据终端或者普通数据终端。

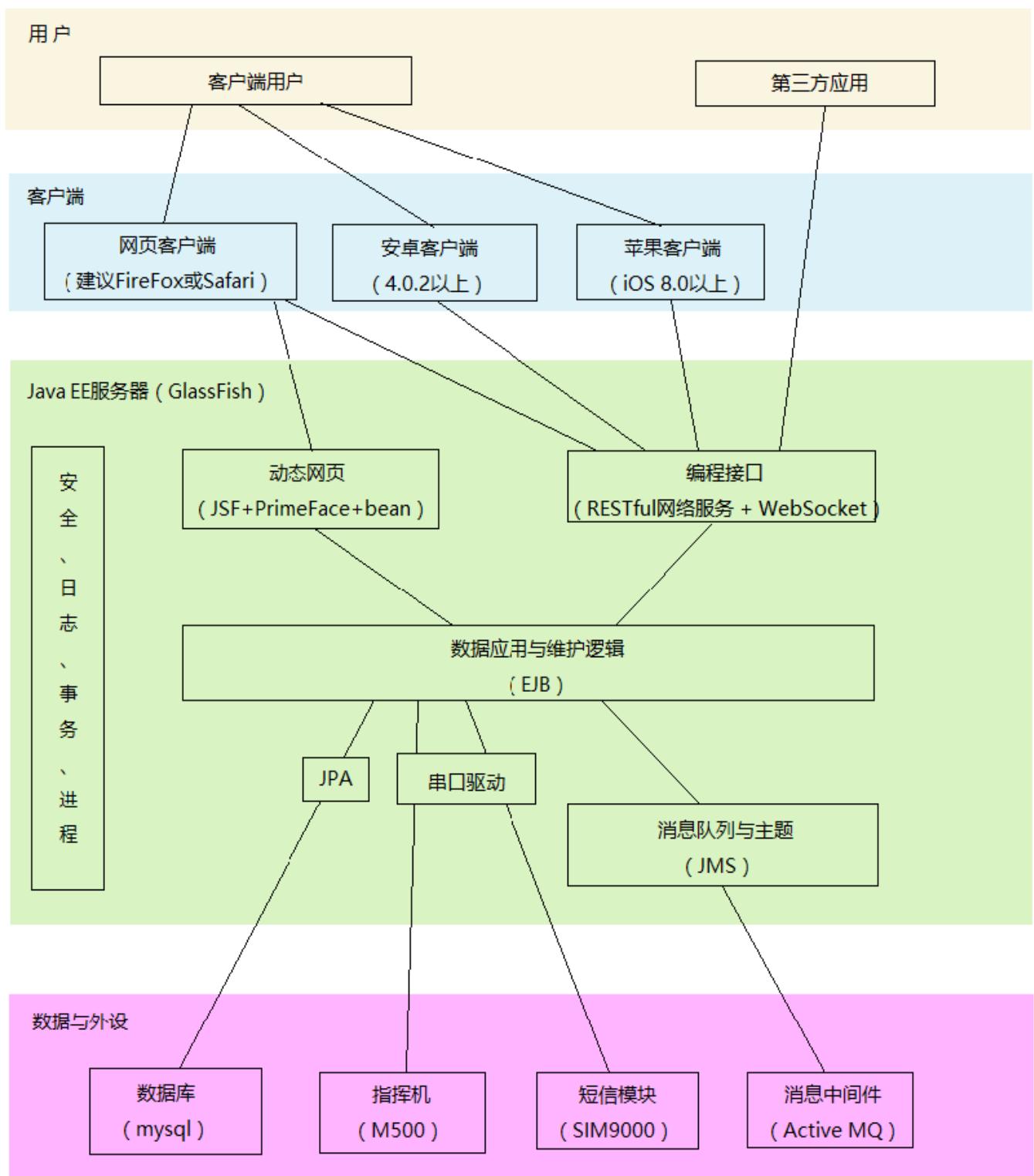
需要说明的是：服务平台支持终端与客户端（互联网）之间的双向联通，但不支持终端与手机短信（移动通信网）之间的双向联通（只支持终端向手机短信的单向联通）。这是基于安全与性能考量的设计。

系统的逻辑结构图：



2.2 系统的技术结构图

系统的技术结构图：



2.3 实现系统的技术

全息时空北斗运营服务平台的软件系统由最新技术实现：

- 中心基于 Java EE 7 的架构实现，它汲取近几年企业应用实现的精华，提供各项技术的最新规范。
- 基于大量企业应用开发的经验：松耦合、强内聚、面向接口编程、面向服务构建。
- 全部基础功能仅用官方基础技术实现：PrimeFace + CDI + JPA，不依赖于第三方技术。
- 串口设备与服务器之间的数据交换利用 JMS 技术（队列与主题）实现异步读写和并发处理。
- 提供 RESTful 接口，客户端或者第三方应用可以自选数据格式（XML 或者 JSON）。
- 提供 WebSocket 编程接口，客户端或者第三方应用可以获得服务器推送的实时数据。
- 最大程度实现软件复用：利用 Netbeans 的模板化技术从数据库直接自动生成 MVC 各层文件。
- 基于各类开源软件：NetBeans + GlassFish + MySQL + ActiveMQ + Log4j + Git + Maven + 百度地图接口 + JasperSoft。
- 苹果客户端以 swift 编写且利用 xcode 最新布局技术，从而一次设计和编程即适用于所有苹果机型。
- 安卓客户端采用 Android Studio 框架编程，并以 fragment 实现主页面切换。
- 完全的国际化：内部核心数据、接口交换数据、JSF 页面元素、Jasper 报表元素、客户端界面元素
- 基于 selenium/Python 的自动化功能测试
- 性能优化：JMS 中间件、JSF 数据列表的延迟加载（Lazy loading）、基于 JPA 层注入的读写分离、GTID 模式实时同步的主从数据库、避免陈旧读问题（stale read）的优化方案。

2.4 技术和工具的选择

2.4.1 背景和原则

本系统：

- 1) 从空白开始，因此尽可能应用最新的成熟技术。
- 2) 研发是探索性的，因此不必遵循任何已有流程，怎么效率高怎么做。
- 3) 没有项目需求，因此需要自行挖掘和定义用户需求和技术需求。
- 4) 在没有启动实际项目之前，尽可能少地消耗资源（人力、软硬件、时间）
- 5) 在探索阶段，系统需求宜广度拓展、不宜过深挖掘，真实需求和领域知识应该来自真实用户和项目实践。

经过实践，我的结论是：新技术都是简洁、强大、易学、易用的。

2.4.2 CS（客户端应用）还是 BS（浏览器应用）？

“从发展轨迹来说，BS 是人们对 CS 的不满积累到一定程度，才被发明出来的，是大趋势。而且随着移动互联和云计算发展，绝大部分 CS 管理软件肯定会被淘汰掉”：

<https://www.zhihu.com/question/24368490>

2.4.3 NoSQL 还是 Mysql

在系统研发初始时，对系统容量进行了评估：参见《2015-5-25-业务发展可行性报告研究报告与技术方案》。

假设：

- 1) 每条北斗数据（位置或者短报文）长度不超过 256 字节
- 2) 每个数据终端平均 10 秒发送一次位置数据或者短报文
- 3) 同时在线的最大数据终端数为 10 万台
- 4) 同时访问客户端的用户数最多为 1 万个
- 5) 同时读取数据接口的用户数不少过 1000 个
- 6) 实时数据库保存最近 15 天的数据
- 7) 历史数据库保存 1 年的北斗数据

则估算为：

- 1) 北斗数据写入流速：

$$100000 / 10 = 10,000 \text{ 条 / 秒} = 2.56 \text{ M 字节 / 秒}$$

- 2) 数据库存储量速度：

$$10000 * 24 * 60 * 60 = 864 \text{ M 条 / 天} = 221.184 \text{ G 字节 / 天}$$

- 3) 实时数据库存储量：

$$10000 * 15 * 24 * 60 * 60 = 12,960 \text{ M 条} = 3.32 \text{ T 字节}$$

- 4) 历史数据库存储量：

$$10000 * 24 * 60 * 60 * 365 = 315.360 \text{ G 条 / 年} = 80.7 \text{ T 字节 / 年}$$

Mysql 的表空间的限制：

<https://dev.mysql.com/doc/mysql-reslimits-excerpt/5.7/en/table-size-limit.html>

“MySQL 准入规范及容量评估”：

<http://blog.itpub.net/26355921/viewspace-2133632/>

“MySQL 对于千万级的大表要怎么优化？”：

<https://www.zhihu.com/question/19719997>

通过读写分离、主从架构、索引、分区等技术，mysql 可以满足系统性能需求。

2.4.4 云还是 Java EE 7?

在数据规模有限时，没有必要云。

系统基于 Java EE 7 架构实现：

技术	解决问题	作用
动态网页技术 (JavaServer Faces, JSF)	编程	丰富的表现层、简化开发
上下文与依赖注入 (Contexts and Dependency Injection CDI)	编程	解耦合、简化配置
数据持久化 (Java Persistence API, JPA)	编程	面向对象的数据操作、简化开发
消息服务 (Java Message Service , JMS)	分布式	异步传输、数据分发、解耦合
基于 XML 的网络服务 (Java API for XML Web Services, JAX-WS)	分布式	远程调用、跨网传输
RESTful 风格的网络服务 (Java API for RESTful Web Services, JAX-RS)	分布式	轻量级的网络服务
Java Transaction API (JTA)	分布式	事务处理
全双工通信 (Java API for WebSocket)	分布式、性能	浏览器和服务器之间数据直传
JSON 处理 (Java API for JSON Processing)	性能	轻量级的数据交换
企业应用的安全配置	安全	通过服务器配置实现访问控制

2.4.5 Struts+Spring+Hibernate 还是 JSF+CDI+JPA ?

实现 Java 的 web 服务器有多种技术和成熟的框架可选，经过分析和判断，我选择 Java 官方基础框架 JSF+CDI+JPA。理由如下：

1) 虽然过去十几年因为 EJB2 的拖累而使 SSH(Struts+Spring+Hibernate)大行其道，但是近年来 java 官方已经吸取 ssh 的精华、提供了更为简洁强大的标准框架 JSF+CDI+JPA。

“Introduction to JavaServer Faces 2.x”：

<https://netbeans.org/kb/docs/web/jsf20-intro.html>

“Getting Started with Contexts and Dependency Injection and JSF 2.x”：

<https://netbeans.org/kb/docs/javaee/cdi-intro.html>

“Using the Embedded EJB Container to Test Enterprise Applications ”：

<https://netbeans.org/kb/docs/javaee/javaee-entapp-junit.html>

2) 制定 JSF 标准的核心人物是 Struts 的核心开发成员。

- 3) 制定 CDI 标准的核心人物是 Spring 的核心开发人员。
- 4) 国外迁移到官方技术的大拿：“我为何停止使用 Spring”
<http://www.infoq.com/cn/news/2013/12/why-i-stop-using-spring/>
- 5) 国内弃用 SSH 的大拿：“还 ssh，多少年前啊！知识要更新啊！！！”
<https://www.zhihu.com/question/21142149>
- 6) 入门门槛非常低：从 JavaEE 小白到生成 ajax 网页只是 2 分钟的事。
<https://netbeans.org/kb/docs/web/jsf20-crud.html>
- 7) 官方技术意味着通用和持续：
<https://stackoverflow.com/questions/7238407/will-spring-support-cdi>

2.4.6 SOAP 还是 RESTful Web Service？

因为系统是从空白开始的，所以毫无疑问地采用最新最强最简洁的远程过程调用(RPC)模式：RESTful Web Service。而且它是 Java EE 的标配了。

“Web Services Learning Trail”：

<https://netbeans.org/kb/trails/web.html>

“怎样用通俗的语言解释 REST，以及 RESTful？”：

<https://www.zhihu.com/question/28557115?sort=created&page=2>

“SOAP web service 与 Restfull web service 之间的区别”：

<https://www.cnblogs.com/dairuiqing/p/6750340.html>

2.4.7 轮询、MQTT 还是 WebSocket？

感谢技术进步，现在的服务器数据推送机制已被标准化了：WebSocket 众望所归成为实时监控和即时消息的终极武器。而且它是 Java EE 的标配了。

“Using the WebSocket API in a Web Application”：

<https://netbeans.org/kb/docs/javaee/maven-websocketapi.html>

“WebSocket 是什么原理？为什么可以实现持久连接？”：

<https://www.zhihu.com/question/20215561>

本系统正是利用 WebSocket 实现了服务器与客户端（网页/智能终端）的双向通信。学习曲线非常平滑。

2.4.8 Eclipse、IntelliJ IDEA、还是 NetBeans？

集成编程环境是程序员工作效率的关键因素。我选择 NetBeans 的原因是：

- 1) 官方技术与标准最为贴切：“What's the Difference between NetBeans Platform and Eclipse RCP?”
<https://netbeans.org/features/platform/compare.html>
- 2) 所需的调试环境齐全、文档丰富：
<https://netbeans.org/kb/trails/java-ee.html>
- 3) “Eclipse 平台是对厂商有利，NetBeans 是对开发者有利。”：
<https://blog.csdn.net/stevech/article/details/1444786>
- 4) “做 java 窗体程序，我用 netbeans，很舒服。对比我 eclipse 的插件，我觉得想哭，搞得头疼”

<https://www.zhihu.com/question/52750081>

<https://stackoverflow.com/questions/13150484/eclipse-window-builder-vs-netbeans-gui-builder>

5) NetBeans 模板化技术实现自动生成从数据库到页面的 MVC 各层文件:

<https://blog.csdn.net/garfield007/article/details/53876060>

6) 老王卖瓜：“换到 NetBeans IDE 的人员讲述的真实故事！”

https://netbeans.org/switch/realstories_zh_CN.html

在我的实践中：

1) 系统有约 80 个数据库表，利用 NetBeans 的 Wizard 几分钟就自动生成了 103 个 Entity Bean、80 个 session beans, 80 个 JSF controller beans, 320 个 JSF files, 还有语言国际化文件，以及其它支撑代码。

2) 无需写 jQuery、前端的开发工作只是修改自动生成的 xhtml、后端的工作主要是改写自动生成的 controller beans 和 session beans，还可以通过定制模板来批量定制 xhtml，工作量被大大节省。

3) 学习曲线平滑。依照官方指南快速上手，在研发中逐步应用和实践了 JavaEE 几乎所有的特性（MVC、JSF、CDI、JPA、EJB、EL、JCA、事务、安全、并发等等）。

4) 这种框架最适合全栈程序员：一个需求改动，从数据库到网页可以直达，无需中间文档和人员沟通。

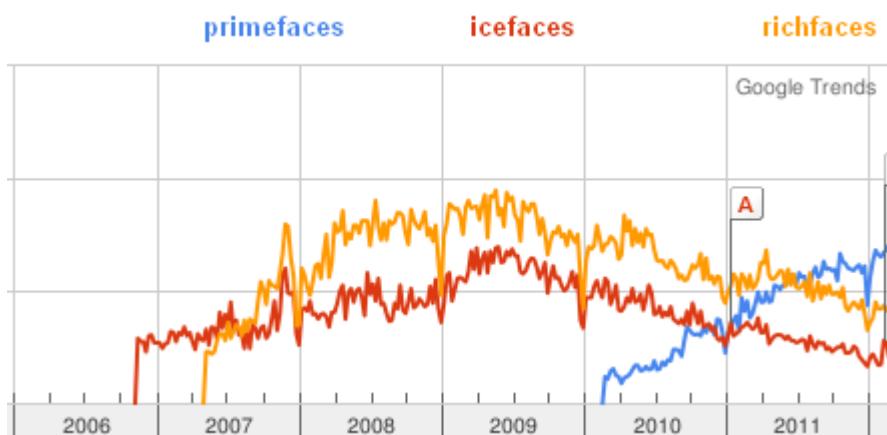
2.4.9 PrimeFaces、ICEfaces、还是 RichFaces？

“组件式、拿来即用，熟悉之后可进行快速开发”：

<https://zhidao.baidu.com/question/1447347030477322900.html>

以下是 2012 年的一个评测：

<http://www.mastertheboss.com/jboss-web/richfaces/primefaces-vs-richfaces-vs-icefaces?showall=1>



PrimeFaces 的 Demo 显示了它丰富的界面组构能力，文档详实、且有代码示例：

<https://www.primefaces.org/showcase/index.xhtml>

TreeTable - Basic

TreeTable displays hierarchical data in tabular format.

Document Viewer		
Name	Size	Type
► Documents	-	Folder
► Pictures	-	Folder
► Movies	-	Folder

basic.xhtml | BasicView.java | DocumentService.java | Document.java

```

1 <h:form id="form">
2   <p:treeTable value="#{ttBasicView.root}" var="document">
3     <f:facet name="header">
4       Document Viewer
5     </f:facet>
6     <p:column headerText="Name">
7       <h:outputText value="#{document.name}" />
8     </p:column>
9     <p:column headerText="Size">
10      <h:outputText value="#{document.size}" />
11    </p:column>
12    <p:column headerText="Type">
13      <h:outputText value="#{document.type}" />
14    </p:column>
15    <p:column style="width:24px">
16      <p:commandLink update=":form:documentPanel" oncomplete="PF('documentDialog').show()" title="View Detail" styleClass="ui-icon ui-icon-search" />
17      <f:setPropertyActionListener value="#{document}" target="#{ttBasicView.selectedDocument}" />
18    </p:commandLink>
19  </p:treeTable>
20 <p:dialog id="dialog" header="Document Detail" showEffect="Fade" widgetVar="documentDialog" modal="true" resizable="false">
21   <o:outPanel id="documentPanel">
22 </p:dialog>
23 
```

2.4.10 JBoss 还是 GlassFish?

我选择 GlassFish 的理由：

1) 它是 java 官方提供的免费开源 JavaEE 服务器，与 NetBeans 打包提供。

<https://javaee.github.io/glassfish/>

2) 它实现了 JavaEE 标准规定的所有特性。

<https://stackoverflow.com/questions/3821640/what-is-the-difference-between-tomcat-jboss-and-glassfish>

<https://stackoverflow.com/questions/24239978/java-ee-web-profile-vs-java-ee-full-platform?noredirect=1>

<https://my.oschina.net/diedai/blog/271367>

3) 良好的管理工具：既有命令行管理、也有网页管理。

<https://bbs.csdn.net/topics/360033323>

4) 它即利于研发（通用、标准），也适用于生产环境。

<https://javaee.github.io/firstcup/>

当然，GlassFish 相对于 JBoss 或者其它商业 JavaEE 服务器在稳定性和安全性方面有所不足：

1) 一个严重的安全漏洞，曾使我放弃升级 4.1：

http://www.nxca.gov.cn/user/user_find.action?artical.id=945

事实上，4.1.1 仍然存在这个漏洞。现在只能禁止远程管理 GlassFish。

2) 5.0 推出时我欣欣然升级，然而一个 webService 解析的 bug 使得我不得不退回到 4.1.1：

查看 用户反馈

数据编号	437
用户反馈的对象	安卓客户端
用户反馈类型	功能需求
标题	解析数据时注意标签可能被强制转换为驼峰命名
描述	实践发现，新版Android Studio似乎把接收到json/xm数据的标签强制转换为驼峰命名。服务器数据格式仍按下划线命名规则，客户端解析时需要判定被强制转换的处理驼峰命名。
报告者	任珊瑚
报告时间	2017-12-08 10:27:35
响应说明	进一步查实，问题不是AS客户端的解析，而是新版GlassFish 5的问题。。。目前仍用GlassFish4.1.2吧~ 已向开发者提交问题报告 https://github.com/javaee/glassfish/issues/22378
响应时间	2017-12-09 21:58:22
响应者	任珊瑚
用户反馈状态	无需解决
用户反馈的来源	网页客户端
解决的版本	-

在具体的项目实践中，应该评估用户需求和资源配置来做服务器的选型。

2.4.11 自带的 JMS provider 还是 ActiveMQ？

GlassFish 自带 JMS provider，支持最新的 JMS 标准 2.0:

<http://ivywang.iteye.com/blog/1930201>

ActiveMQ 是最成功最成熟的开源免费消息中间件，但是仍然只能支持 JMS1.1:

<http://activemq.apache.org/>

我最初选择 GlassFish 自带 JMS provider，研发得很顺利。然而某次指挥机发生数据洪流时（串口源源不断地写入数据），系统崩溃了。查看日志，确定问题出在 JMS 异常处理不够强壮（亦或是我理解不深、代码写得不够强壮），然后就改用 ActiveMQ 了。

查看 用户反馈

数据编号	176
用户反馈的对象	平台服务器
用户反馈类型	问题报告
标题	系统日志过多，服务器奔溃
描述	当数据错误时，服务器写入过多日志文件，以至于导致服务器奔溃
报告者	任珊瑚
报告时间	2016-07-12 10:35:56
响应说明	改用activeMQ作为消息中间件以加强异常处理。配置服务器参数，提高JVM可使用内存上限。
响应时间	2016-09-09 14:01:20
响应者	任珊瑚
用户反馈状态	已解决
用户反馈的来源	网页客户端
解决的版本	-

以下是 2015-8-1 进度报告的一页：

关于消息中间件

GlassFish 4.1自带的JMS适配器：实现JMS 2.0；调试简捷；对C语言支持有限；缺少更多的扩展功能。

ActiveMQ：实现JMS 1.1和J2EE 1.4；最广泛使用的开源消息中间件；网络资源相对丰富；功能多样；支持REST收发消息。

目前两种方式均已在平台上实现。选择ActiveMQ，理由是其对C/C++的支持。

ActiveMQ的“坑”：

- 1.注意其目前只支持JMS 1.1，不能用JMS简化编程接口。
- 2.对消息驱动Bean的支持有问题，可能是因为不能兼容EJB 3.0（GlassFish 4.1是Java EE 7）。目前是自定义消费类。
- 3.在GlassFish上ActiveMQ资源的JNDI查询有问题，没有找到解决办法，目前是直接写参数。
- 4.文档滞后，官方资料有不少不能适用最新版本的地方，需要自行排难。

我尝试用通过 JCA 方式来连接 ActiveMQ，但是存在问题，因此只好采用客户端直连方式来使用 ActiveMQ。

查看 用户反馈	
数据编号	222
用户反馈的对象	平台服务器
用户反馈类型	问题报告
标题	当透过JCA来使用activeMQ时有很大的延迟
描述	利用JCA，可以兼容性地采用activeMQ技术（它仍然只支持JMS 1.1接口），代码无需改动，只需配置。但是每次创建连接时，固定有2分钟延迟！原因未知。
报告者	任珊虹
报告时间	2016-09-02 17:23:08
响应说明	已经花费了很多精力试图解决这个问题，但是无解。目前，只能用client方式直接连接，即activeMQ非兼容的代码。
响应时间	2016-09-02 17:28:17
响应者	任珊虹
用户反馈状态	以后解决
用户反馈的来源	网页客户端
解决的版本	-

关闭

总之，这部分需要根据技术发展而改进。

2.4.12 BIRT 还是 JasperReports？

网评两个开源报表工具各有千秋：

<http://www.innoventsolutions.com/comparison-matrix.html>

<http://www.innoventsolutions.com/open-source-reporting-review-birt-jasper-pentaho.html>

<https://community.jaspersoft.com/questions/528364/jasperreports-vs-birt>

http://www.finereport.com/knowledge/acquire/jasperreport_birt.html

我选择 JasperReports 的理由：

- 1) 它有自己的 IDE、文档丰富、生命力持久。
- 2) 既然没有选择 Eclipse 平台，放弃 BIRT 也在意料之中。

2.4.13 高德地图还是百度地图？

网上评议：

<https://www.zhihu.com/question/23770466>

http://www.360doc.com/content/16/0609/21/9200790_566360984.shtml

当初选择百度地图接口纯粹是因为：学习时找到的示例是百度的。

2.4.14 商用 SSL 证书还是免费 SSL 证书？

在没有正式项目之前，没有必要购买商用 SSL。以下是关于 SSL 证书的讨论：

<https://www.zhihu.com/question/19578422>

本系统曾经在 2017 年 1 月申请并且配置了免费证书 StartSSL（<https://startssl.com/Certificates>），三个月后过期没有延期。现在这个网站已经关闭、并且不再提供服务了~

当前热火的免费 SSL 是 Let's Encrypt。

2.4.15 anaconda、Spyder、还是 pycharm？

这三个 Python 编程环境我都尝试了，最后选定了 pycharm，因为它是 jetbrains（Intelli IDEA）出品的、来自最专业的 IDE 厂商：

<https://www.jetbrains.com/products.html#lang=python>

<https://www.zhihu.com/question/48168875>

2.4.16 Eclipse 还是 Android Studio？

我选择 Android Studio(AS)的理由是：

- 1) AS 是谷歌官方出品，必然越来越好。

<https://stackoverflow.com/questions/17849078/which-android-ide-is-better-android-studio-or-eclipse?noredirect=1>

<https://stackoverflow.com/questions/29574732/android-studio-vs-eclipse-with-adt-2015?noredirect=1>

<https://stackoverflow.com/questions/35437264/what-is-the-main-difference-between-using-eclipse-with-android-plugin-and-using?noredirect=1>

2) AS 基于 IntelliJ IDEA, 非常专业:

<https://www.zhihu.com/question/21534929>

3) 谷歌已声明不再支持 Eclipse:

<https://blog.csdn.net/ricohzhanglong/article/details/46681333>

4) AS 带来大量最新技术:

<https://www.airpair.com/android/android-studio-vs-eclipse>

2.4.17 HttpClient、HttpURLConnection、还是 OkHttp

目前 Android 缺省的 http 客户端是 HttpURLConnection。网上的结论是 OkHttp 更好:

“OkHttp 不仅具有高效的请求效率, 并且提供了很多开箱即用的网络疑难杂症解决方案”

<https://www.jianshu.com/p/ca8a982a116b>

“Android 4.4 is using OkHttp for its internal HttpURLConnection implementation”

<https://stackoverflow.com/questions/26000027/does-android-use-okhttp-internally>

“最近有时间看了 OkHttp 网络请求, 体验感觉上升了好几个档次”

<https://blog.csdn.net/qqqGL/article/details/50518840>

2.4.18 Objective-C 还是 Swift

我选择 Swift 的理由:

- 1) 没有历史包袱, 新系统用新技术。
- 2) Swift 很简洁, 语法风格与 java 非常类似, 学习曲线平滑。
- 3) Swift 是苹果官方主推的未来之星, OC 将被完全取代。

“如何评价 Swift 语言” :

<https://www.zhihu.com/question/24002984>

<https://www.zhihu.com/question/24304009>

“如何将旧的 Objective-C 项目逐渐转为 Swift 项目” :

<https://www.jianshu.com/p/8adfa42ce784>

2.5 系统的特点

2.5.1 信息安全

全息时空北斗运营服务平台保障信息安全：

- “用户必须合法地获得资源”。用户、客户端、第三方应用均需要提供合法账户信息（认证）才能访问被指派的资源（授权）。
- “数据的来源必须合法”。北斗数据通道天然地保障了北斗终端报告数据的合法性；而普通数据终端需要提供报告码才能提交/访问数据。
- 功能权限控制由账户的角色决定。不同的角色可以使用不同的系统功能集合。
- 数据权限控制由账户/用户组与终端之间的指派关系决定。普通用户必须明确被指派可访问的终端，未指派的数据被禁止访问。
- 权限管理数据均有灵活的管理手段：通过设置“合法性”可以随时启停权限；通过设置“有效期”可以自动启停权限。
- 用户组用以实现集团用户模式的权限管理；同时单个用户也可以被灵活授权以实现个人用户模式。
- 专门的数据中心：自建的机房、顶级的服务器、防火墙和存储设备、自主研发的北斗硬件设备、专职的网络管理员、严格的管理规范。
- 平台运行在最强的 Linux64 (Cent OS 7) 平台上，并执行严格的安全配置。
- 平台的开发和运行均基于开源软件。
- 完备的系统日志和应用日志

2.5.2 按需定制

全息时空北斗运营服务平台支持快捷地按需求定制软件：

- 既适用于北斗定位应用，也适用于非北斗的定位应用（如普通的 GPS 车载应用、儿童手表、或者运动手环）。
- 既兼容集团用户模式的应用，也兼容个人用户模式的应用。
- 针对具体的数据类型可简易地扩展成不同的业务系统。例如，可插件式地添加新的数据采集模块；又如，可新增报警类型，添加少许代码即可实现数据融合和界面展示。

查看 用户反馈

数据编号	432
用户反馈的对象	数据
用户反馈类型	功能需求
标题	衍生数据：步长、速度、和行程
描述	基于位置数据，可以自动计算终端步长，即每个位置点与上一个位置点的距离。步长又可以用来计算速度和累计行程。
报告者	任珊虹
报告时间	2017-11-24 09:11:02
响应说明	根据不同类型的应，步长、速度、和行程数据可以提供不同的功能。例如，终端可能是计步器、车辆、手机。
响应时间	2017-12-29 11:31:47
响应者	任珊虹
用户反馈状态	以后解决
用户反馈的来源	网页客户端
解决的版本	-

2.5.3 管理/使用方式

全息时空北斗运营服务平台提供便捷的管理/使用方式：

- 管理员可以通过网页来使用系统全部功能。
- 可配置的系统参数包括串口管理参数、线程管理参数、数据管理参数、队列管理参数等。
- 系统管理员可以打开/关闭数据串口、查看后端周期性线程的状态。
- 北斗波束和北斗日志被自动收集和存储，管理员可以直接检查北斗数据通道的状态。
- 开发者可以通过调试功能来检查系统异常。如发送北斗命令、解析北斗数据、模拟接收北斗数据、管理GSM模块、模拟网络服务客户端等。
- 在线提交和处理用户反馈。普通用户、管理员、开发者用统一的方式提交报告问题、改进建议、和功能需求。处理进度直接显示给所有用户。
- 普通用户可选择网页客户端、苹果客户端、或者安卓客户端

2.5.4 北斗终端的接入方法

全息时空北斗运营服务平台支持北斗终端无代码接入：

- 北斗数据的收发格式请符合国家制定的北斗协议。
- 北斗终端把数据发往服务平台卡号：166762，则数据就被注入平台了，通过客户端即可访问数据。
- 目前服务平台相信所有北斗卡：当收到任何陌生北斗卡发来的数据时，系统自动注册此终端。（将来可能只允许人工注册）
- 若终端需要服务平台转发消息给手机号，请使用格式“手机号::消息内容”。
- 若终端需要向服务平台的账户发送消息，请使用格式“服务平台账户名::消息内容”。
- 若终端收到服务平台发来的消息符合格式“字符串::字符串”，则这是服务平台账户发给终端的消息，即“服务平台账户名::消息内容”。
- 平台可以与终端约定特殊的数据交换协议。（例如北斗伴侣要求转发手机短信的特殊短报文协议）

请注意：服务平台只转发终端发往手机的短信，但不转发任何手机短信给任何终端。

查看 用户反馈	
数据编号	468
用户反馈的对象	数据
用户反馈类型	改进建议
标题	北斗位置数据的注入加入时间和距离因素
描述	目前是在数据库中写入每一个读取到的北斗位置数据。当位置持续不变时，可以借鉴GPS定位API的策略：加入最长时间间隔和最小距离间隔。即：在最长时间间隔之后才会处理新的定位数据、只有大于最小距离的位置才是新的有效数据。
报告者	任珊虹
报告时间	2018-02-11 10:19:46
响应说明	参考了Android官方的BetterLocation算法。
响应时间	2018-04-14 20:17:34
响应者	任珊虹
用户反馈状态	已解决
用户反馈的来源	网页客户端
解决的版本	平台服务器-5.300

查看 用户反馈

数据编号	518
用户反馈的对象	平台服务器
用户反馈类型	功能需求
标题	加入用于处理新位置数据的系统参数
描述	加入三个系统参数：新位置的最小距离、新位置的最小间隔、新位置的最大间隔
报告者	任珊瑚
报告时间	2018-04-13 14:45:35
响应说明	系统管理员可以设置这三个参数。
响应时间	2018-04-14 20:14:49
响应者	任珊瑚
用户反馈状态	已解决
用户反馈的来源	网页客户端
解决的版本	平台服务器-5.300

查看 用户反馈

数据编号	425
用户反馈的对象	平台服务器
用户反馈类型	功能需求
标题	轨迹数据的抽样算法
描述	在查询显示轨迹时，如何抽取数据来有效显示轨迹的主要特征，需要设计合理的算法。等时抽样是最简单的算法，但是多数情况下并不能展示主要轨迹。例如，终端长时间停留在同一位置，则这些数据在轨迹上的重要性就不如终端移动的位置数据。抽样算法涉及压缩率和统计方差领域的知识。
报告者	任珊瑚
报告时间	2017-11-20 09:32:06
响应说明	通过改进位置数据注入的算法（参见编号为468的用户反馈），可以避免大量冗余位置数据（长时间不移动）写入数据库。目前暂时不需要复杂的抽样算法。
响应时间	2018-04-25 10:01:27
响应者	任珊瑚
用户反馈状态	无需解决
用户反馈的来源	网页客户端
解决的版本	-

2.5.5 非北斗的数据终端的接入方法

全息时空北斗运营服务平台支持非北斗的数据终端的接入：平台与数据终端约定数据交换协议，终端数据注入平台后，用户即可通过客户端访问数据。

数据交换协议包括：

- 数据终端身份认证的模式。“数据的来源必须合法”，因此终端需要与平台约定如何证明自己是合法数据源的方式。可采用的方案有：终端的标识码/报告码，或者在数据传输层约定特殊标识。
- 数据终端与平台交换数据的途径。可采用的方案包括：通过网络服务、或者通过 socket 编程。
- 其它应用需求。

实际上，一旦终端数据被注入平台的数据池，后续的逻辑对于北斗终端或者非北斗终端都是一样的：

- 平台对数据进行备份和统计管理。
- 用户在各种客户端上操作和查看终端数据。
- 第三方应用通过网络服务或者 socket 编程访问平台数据。

查看 用户反馈	
数据编号	484
用户反馈的对象	平台服务器
用户反馈类型	功能需求
标题	位置数据的需要区分坐标体系
描述	由于加入百度定位方式，则本地保存的位置数据可能是百度坐标（没有正式的逆转换方法）也可能是GPS坐标。当允许本地数据上传服务器时，服务器的表需要区分不同的坐标数据。
报告者	任珊虹
报告时间	2018-03-01 16:12:11
响应说明	所有包含位置数据的表都加了字段“坐标系”。但是数据处理仍然只有GPS坐标系，将来需要区分。
响应时间	2018-04-14 20:15:34
响应者	任珊虹
用户反馈状态	正在处理
用户反馈的来源	网页客户端
解决的版本	平台服务器-5.300

2.6 一些误读及其解释

北斗应用并不为多数人所知，因此会有一些误读。以下针对一些已知的概念混淆进行解释：

1. **请区分北斗定位终端和常规定位终端。** 北斗定位芯片日益广泛，智能终端通常是多种定位模式可选、在硬件层面实现自动切换，所以在公网覆盖下软件应用对于定位终端并没有特殊处理的需要。当定位终端包含北斗通信卡时，它就可以脱离公网而通过北斗卫星实现数据收发了，此时就需要相应的应用软件来收发和解析北斗数据。因此，北斗定位终端特指包含北斗定位芯片和北斗通信卡的数据终端。
2. **请区分北斗应用和非北斗应用。** 装有北斗定位芯片而以公共网络收发数据，这是常规应用；定位和通信的数据通过北斗卫星收发，这是北斗应用。根本区别在于终端数据传输的方式而不是定位方式。这是狭隘区分法，会有人把凡是使用北斗卫星定位的应用叫北斗应用。对于软件开发，我们用狭隘定义，常规应用不在我们目前的开发范围内。当我们谈到北斗应用时，是以数据终端不使用公网而使用北斗卫星收发数据为前提假设的。北斗应用的最佳场景是公网无法覆盖的荒郊野外。
3. **请区分终端和客户端。** 定位数据来自终端，客户端是用来监视终端的，它们往往是在不同地点由不同的人

持有。例如，驴友拿着终端（如北斗伴侣）四处晃悠，而他们的家人或者旅游局看着客户端关注他们的动向并且与他们实时对话。客户端本身不必要定位。北斗终端以北斗卫星收发数据，客户端总是在公网覆盖的区域内用常规网络来收发数据。

4. **请区分终端应用和客户端应用。** 终端这一端本身可能有专用的应用，用来与客户端或者其它终端进行数据交换；客户端这一端也可能实现定位。终端应用着重于收集“我的哪里”的数据，客户端应用着重于监视“那些人在哪里”。
5. **请区分用户和终端。** 在我们的系统中，“用户”是使用客户端的人。终端的使用者不在系统控制范围内。
6. **请区分客户端和手机。** 客户端是用户登录系统使用功能的入口，不一定是手机。
7. **请区分用户和手机。** 我们的系统是基于用户账号，“用户”与手机或者手机号没有必然的联系。用户可以用同一账号以任意客户端登录系统。
8. **请区分终端需求和客户端需求。** 例如，终端之间需要彼此共享位置和彼此通信，这是终端需求，与客户端无关。例如，终端与客户端之间需要彼此共享位置，这就需要客户端也要实现定位。
9. **请评估客户端需求的必要性。** 有的人可能会希望：客户端定位、客户端之间共享位置、客户端之间对话。“客户端”意味着常规网络和常规定位，目前已有大量的、满足这些需求的、成熟的应用（例如地图类、微信）。即使将来添加这些实现，也只是锦上添花。
10. **手机可以回复终端的短信吗？** 我们支持从终端向手机直接发送短信，于是有用户希望能直接回复短信给终端。这个需求在技术上很容易实现，但是基于安全和性能的考虑，我们决定不支持。允许终端直接向手机发送消息，并不意味我们鼓励用户用短信方式在终端与手机之间通话。终端发往手机的短信应该用来应付紧急情况，即需要报警时可以把消息发往一个临时的手机号。终端与手机之间的对话，请使用我们的手机客户端。

定位应用范围非常广，各类应用有不同的需求，我们的系统目前尽量做最大公约数，谋求一个稳定的内核和灵活扩展的架构。

最新版的安卓应用已实现三合一：既是服务平台的客户端、也可提供本地定位、同时提供北斗终端（北斗伴侣 E200）的管理功能。

查看 用户反馈

数据编号	480
用户反馈的对象	数据
用户反馈类型	功能需求
标题	位置数据，站在用户的角度还是终端的角度？
描述	位置数据是定位芯片直接产生的，但是数据在逻辑上最终是为终端持有者服务。例如，一个人可以换不同的手机定位，我们关注的重点是这个人的位置而不是具体某个手机的位置。又如，一个定位终端可以在不同时间为不同人或物服务，我们关注的重点仍然是使用这个终端的人或物的位置。问题产生于：终端与持有者可以是多对多的关系。
报告者	任珊虹
报告时间	2018-02-28 18:28:18
响应说明	一个位置数据同时记录“用户”和“终端”两个信息。在数据分析时，可以基于不同的角度来抽取和统计位置数据：针对终端、还是针对用户。
响应时间	2018-02-28 19:51:23
响应者	任珊虹
用户反馈状态	已解决
用户反馈的来源	网页客户端
解决的版本	平台服务器-5.120

2.7 系统的伸缩

以下表格分析了各个环节的可能出现的问题以及相应可用技术：

处理环节	问题	可用技术
数据接收	高并发：大量数据写入	消息队列
数据交换	性能/可靠性：与外网交换大量数据	消息中间件、网络服务
客户端请求	高并发：大量用户同时访问	Ajax(网页异步更新)
客户端下载	响应时间慢：页面内容丰富多样（渲染）	缓存、CDN、页面优化
服务器响应	大流量、高负载	分层、集群、缓存
业务处理	逻辑关系复杂	拆分业务、面向服务、解耦、分层
数据库访问	读写互锁	双机热备、读写分离
数据库访问	大流量、高负载	数据库纵向分割、数据库集群
数据库访问	大数据表的查询统计分析	数据库横向分割、hadoop/NoSql
磁盘阵列	存储量持续增加	虚拟硬盘（云存储）
服务器集群	负载持续增大	虚拟主机（云主机）
网络传输	大文件占带宽	多媒体走专网

2.8 跟踪分布式技术

当系统性能出现问题时，我曾分析过分布式技术（如图）的可能性。但是对 JavaEE 代码和 Mysql 全面优化以后，百万行数据的响应时间已小于 1 秒，因此没有继续跟踪分布式技术。

查看 用户反馈

数据编号	392
用户反馈的对象	平台服务器
用户反馈类型	性能优化
标题	物联网：评估分布式计算平台
描述	目前的系统（JavaEE）对于两百万数据行的复杂查询（未深入优化）的响应速度超过10秒。物联网必然是PB级的实时计算，高负载的大数据计算需要采用分布式计算平台。
报告者	任珊虹
报告时间	2017-08-13 11:38:48
响应说明	Flink适用于大规模终端数据的实时收集（分布式流处理），Spark适用于大数据的实时计算和分析挖掘（分布式批处理）。配合时序数据库应该是不错的大数据计算与存储架构。ps. 目前不必考虑尚未成熟的区块链技术。
响应时间	2017-09-23 12:34:03
响应者	任珊虹
用户反馈状态	以后解决
用户反馈的来源	网页客户端
解决的版本	-

查看 用户反馈

数据编号	387
用户反馈的对象	数据
用户反馈类型	性能优化
标题	物联网：评估时序数据库
描述	物联网持续实时更新大量数据，且所有数据都带有时标，所有需求都指向“时序数据库”的优势。评估当前时序数据库的现状，判断是否有可用的、较成熟的、开源的时序数据库。ps. 昂贵过时的“实时数据库”必然不是选择。
报告者	任珊虹
报告时间	2017-07-16 10:48:28
响应说明	需求：海量数据持续实时更新、实时查询数据、历史数据在实时监测中作用小、历史数据积累惊人、历史数据分析
响应时间	2017-09-23 12:34:40
响应者	任珊虹
用户反馈状态	以后解决
用户反馈的来源	网页客户端
解决的版本	-

数据编号	409
用户反馈的对象	数据
用户反馈类型	功能需求
标题	评估OLAP工具
描述	目前系统已利用JasperSoft实现了统计报表的预生成。可以利用OLAP技术提供在线定制报表。
报告者	任珊虹
报告时间	2017-10-18 09:42:51
响应说明	考虑采用kylin，它是较为成熟的基于hadoop的分布式数据仓库产品。另外，还有成熟的数据挖掘工具。
响应时间	2017-11-23 11:17:44
响应者	任珊虹
用户反馈状态	以后解决
用户反馈的来源	网页客户端

3 平台服务器的开发

3.1 开发资源

平台服务器的开发资源包含以下内容：

1) 服务器源码包 “QXSKweb.zip”: 利用 NetBeans 编写平台 JavaEE 服务器的所有资源

2) 文档:

《开发者指南》

《运行环境的安装与配置手册》

《平台用户手册》

《网络服务协议》

《WebSocket 接口》

3) 数据库设计文件 “*.mbw”: 利用 Mysql WorkBench 设计和自动生成数据库

4) 数据库脚本: (必须按这个顺序执行)

`bd_schema_sp.sql` 创建数据库表和存储过程。由 WorkBench 自动生成

`bd_adjust_schema.sql` 调整大表的定义。必须在 MVC 文件自动生成以后执行

`bd_data.sql` 初始数据。必须导入以初始化平台服务

`bd_events.sql` 定义事件。必须导入以每月分区大表

`bd_triggers.sql` 定义触发器。必须导入以维护数据表的一致性

5) 定制的 PrimeFaces 在 NetBeans 中的模板文件:

`Controller.java` `create.xhtml` `edit.xhtml` `list.xhtml` `view.xhtml`

6) 报表设计源码包 “QXSKreport.zip”: 利用 JasperStudio 设计报表模板的所有资源

7) 最近一个版本的安装包:

`sun-resources.xml` 注册外部资源

`QXSKweb.war` JavaEE 服务器

8) 网站文件包 “web.zip”

3.2 调试环境

参见文档《全息时空运营平台 - 运行环境的安装与配置手册》，即以下软件需要安装配置：
JDK、mysql、GlassFish、串口驱动。

由于开发工具 NetBeans 会自动安装 GlassFish，不必专门下载和安装 GlassFish。

3.3 系统的“大表”

本系统中有 6 个数据表是“大表”，即它们的数据会随着时间不断累增，规模极有可能超过 1 百万行（这个规模在现在的技术环境中只能算是“小表”，但是在实际项目中这些表的规模可能是亿级的）。这些表为：

`t_location:` 位置数据表

`t_user_bd_message:` 用户北斗消息表

`t_bd_log:` 北斗日志表

`t_bd_beam:` 北斗波束表

`t_sms_message:` 短信消息表

`t_system_log:` 系统日志表

而以下表是“准大表”，即它们虽然也随着时间累增，但总规模可能没有那么大，因此不被判定为大表。但是若实际项目中表规模超过 1 百万行，则也应该被当作大表处理：

`t_terminal_status:` 终端状态表（含各类警报）

`t_user:` 用户表

`t_user_operate_terminal:` 用户权限表

对于大表，有以下特殊处理：

- 1) 数据表分区（Mysql 脚本）
- 2) 读写分离（JavaEE 代码）
- 3) 数据懒加载（PrimeFaces 代码）

具体描述参见后面的章节。

3.4 数据库设计环境 MySQL Workbench

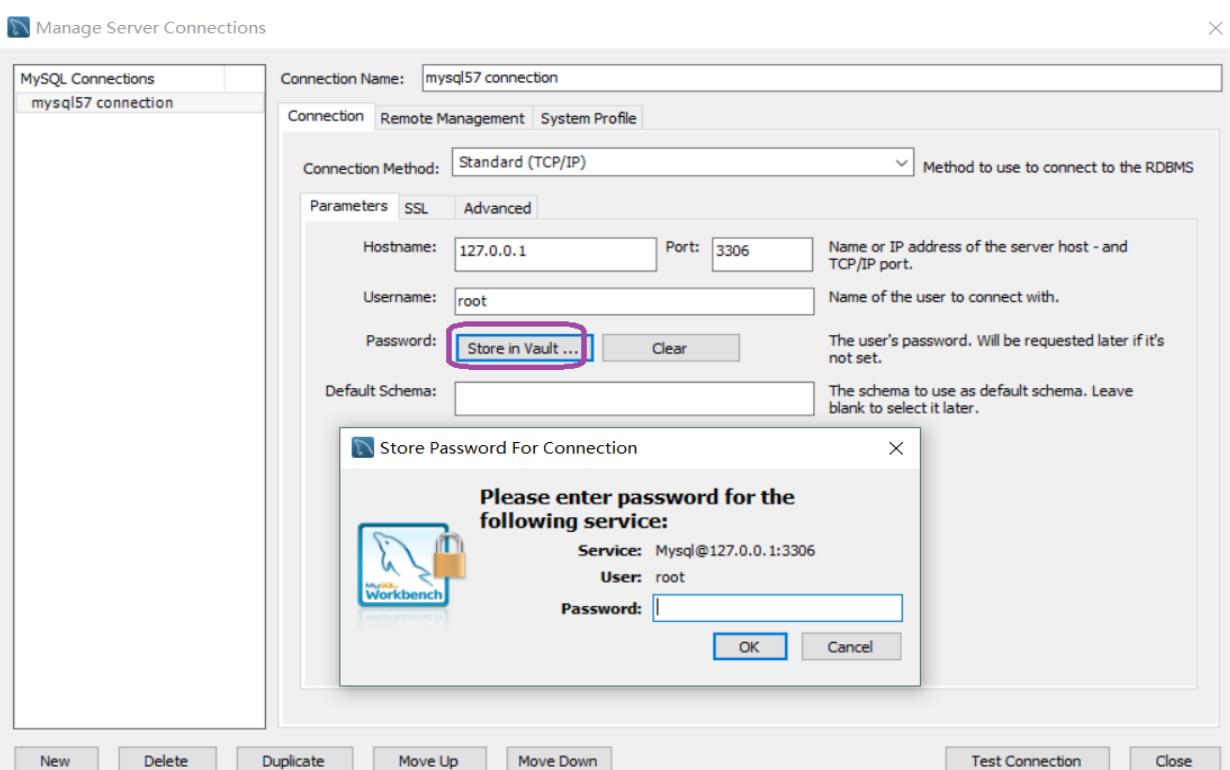
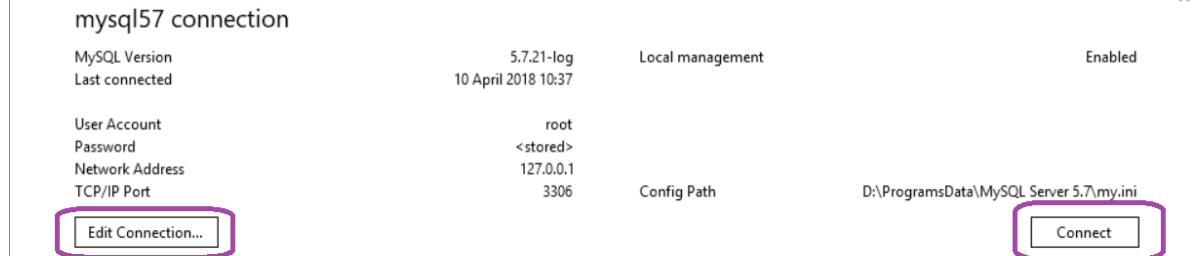
3.4.1 资源地址

官网地址：

<https://www.mysql.com/products/workbench/>

3.4.2 设置数据库连接参数

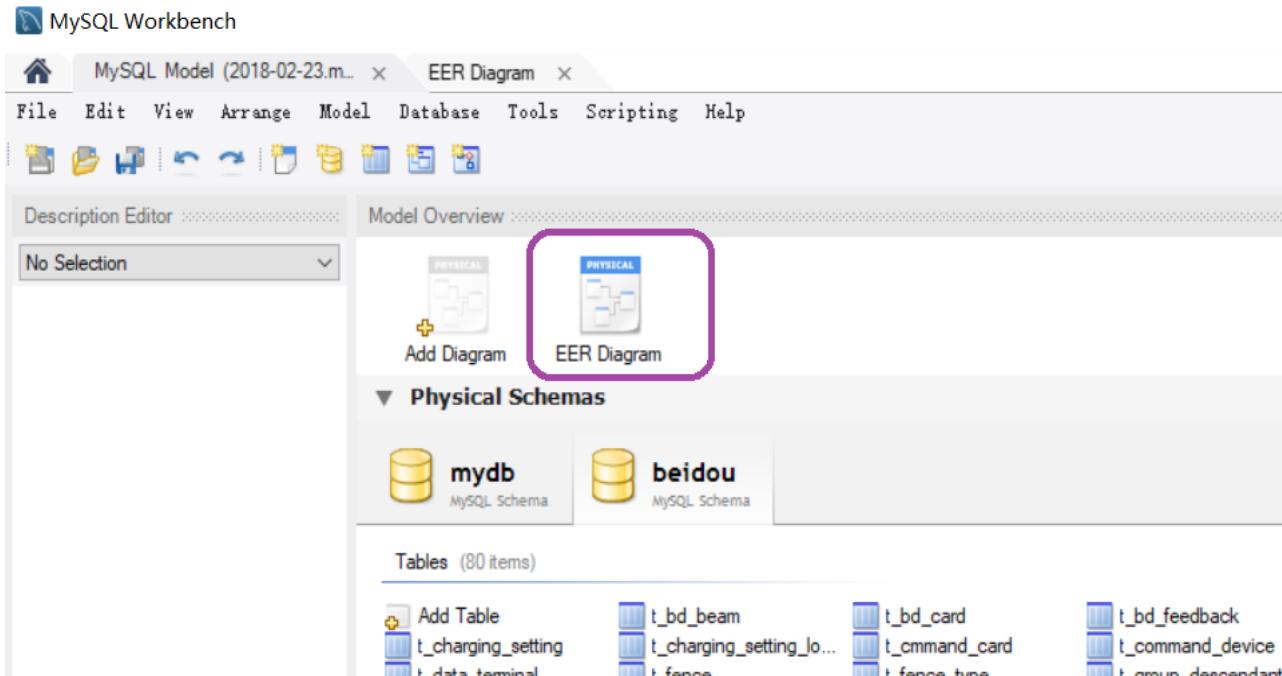
打开 workbench，点击左上角的主页按钮、点击连接板的右上角、点击按钮“Edit Connection”、输入连接账户：



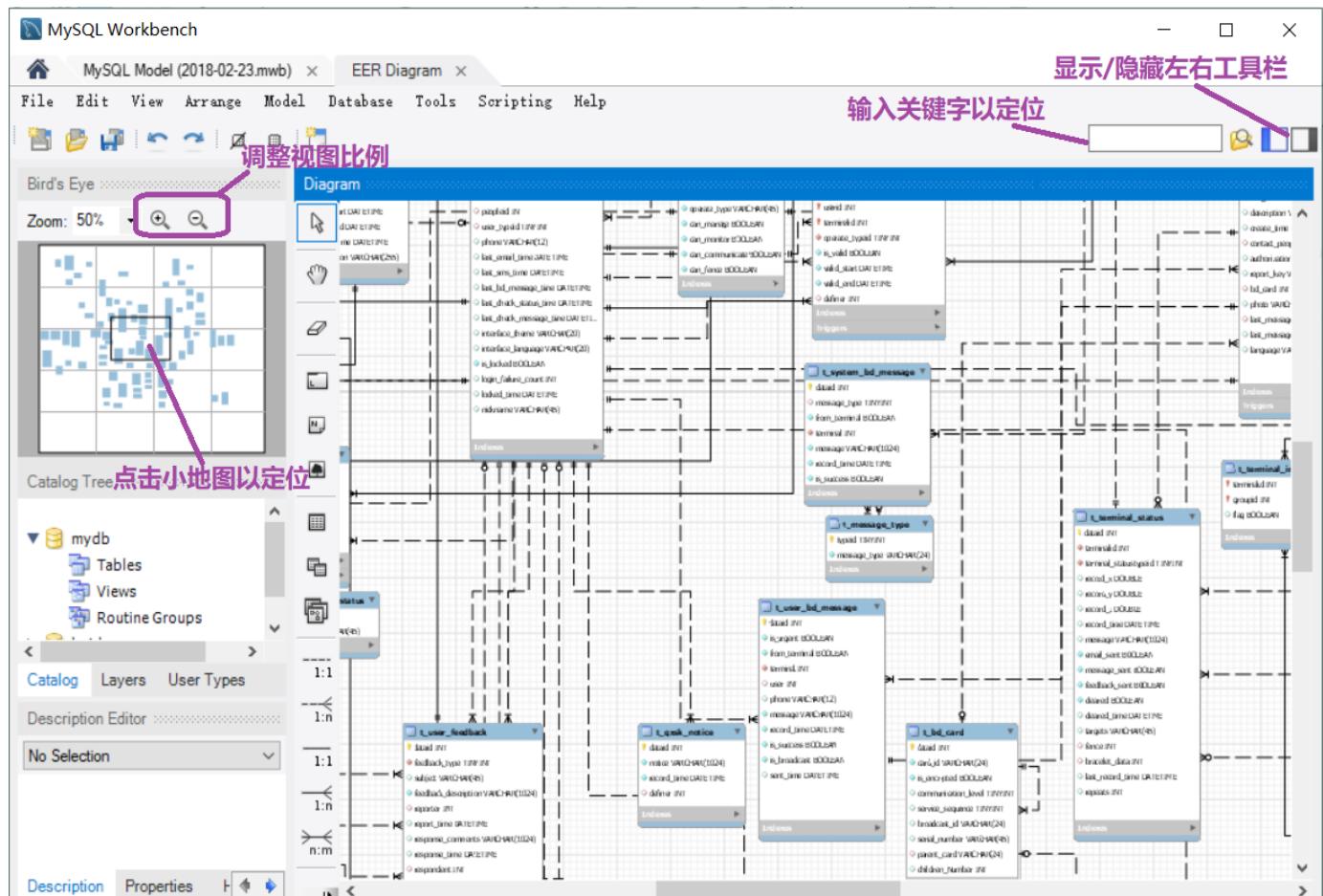
3.4.3 编辑 EER (Enhanced ER)

系统的数据库是在 Mysql workbench 设计并且导出自动生成的：

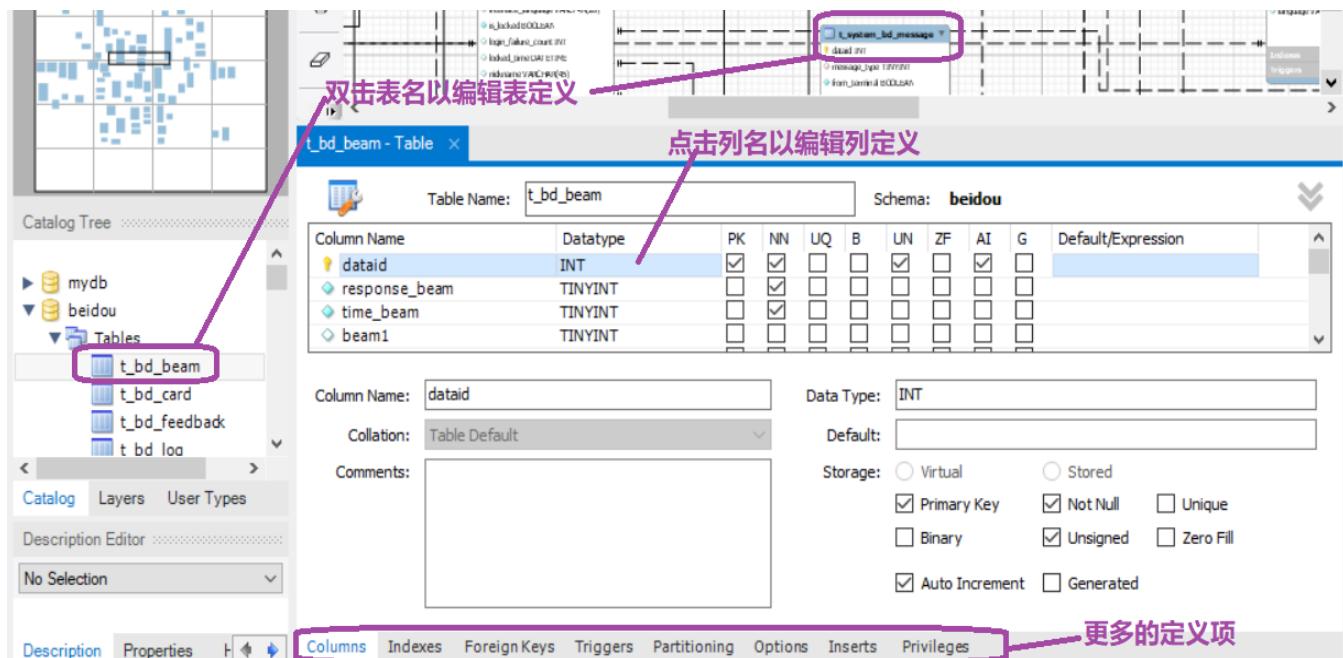
1) 双击数据库模型文件，如“2018-02-23.mwb”，在 workbench 中打开：



2) 双击“EER Diagram”，打开关系图：



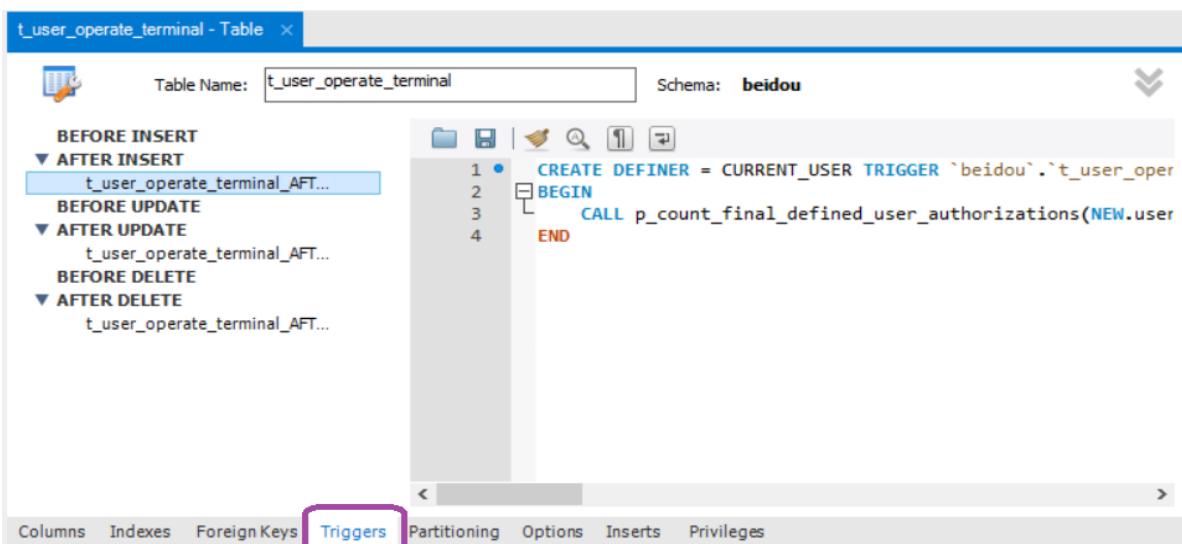
3) 双击表名以编辑表定义：增删改列定义



3.4.4 编辑索引、外键、触发器

在表定义底部选择不同页签，定义表的索引、外键、触发器等特性：

The top screenshot shows the 'Indexes' tab for the 't_data_terminal' table. It lists various indexes, including a primary key and several unique and regular indexes. The bottom screenshot shows the 'Foreign Keys' tab for the same table. It lists foreign keys and their options, such as 'On Update' and 'On Delete' actions.

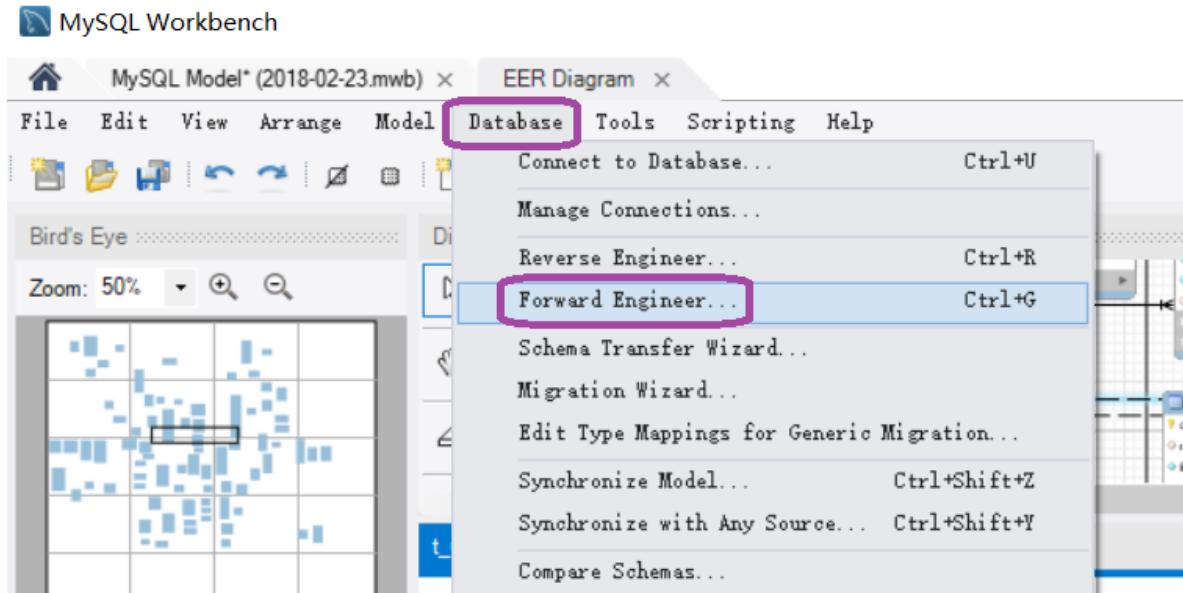


3.4.5 编辑存储过程

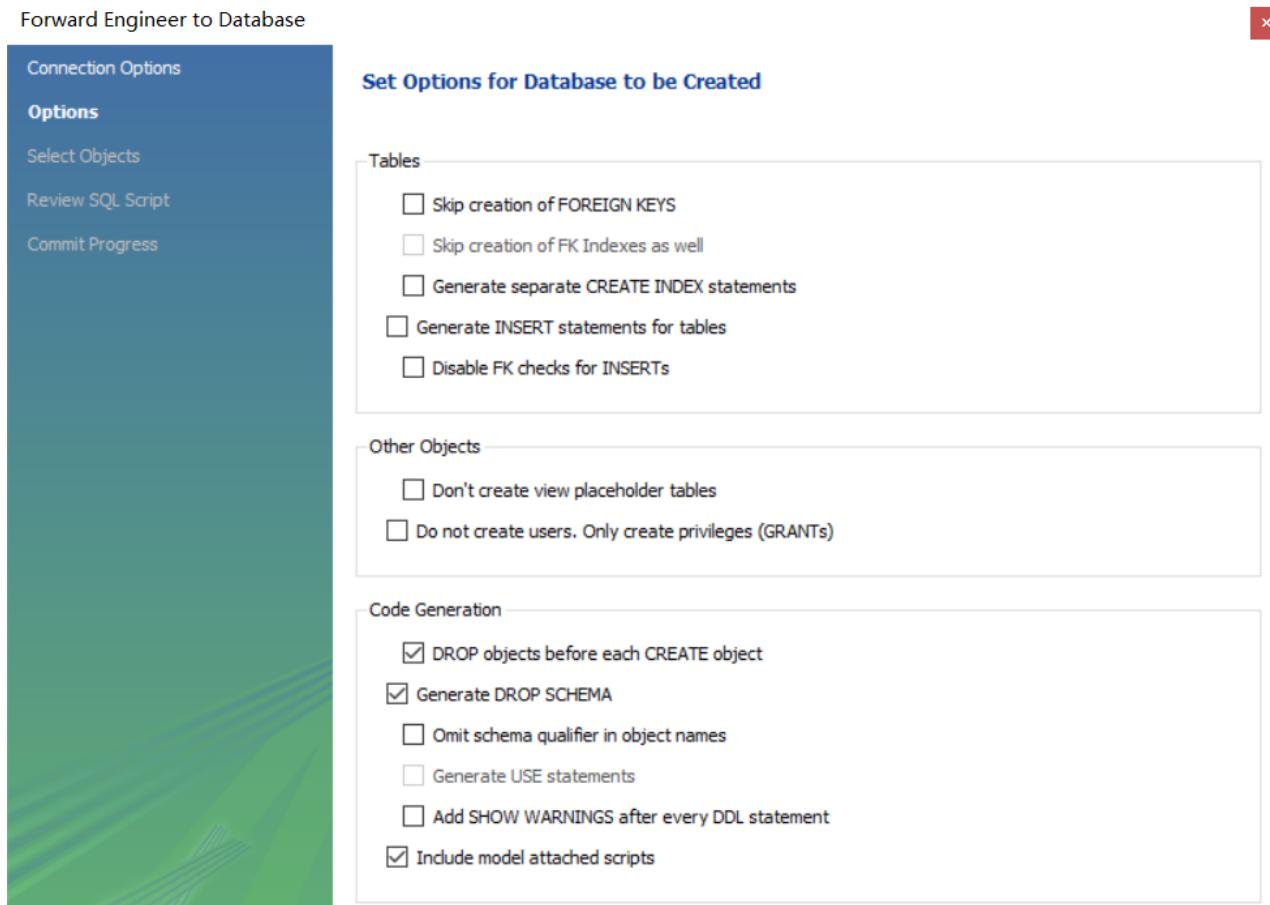
在数据模型视图里，增删改存储过程：

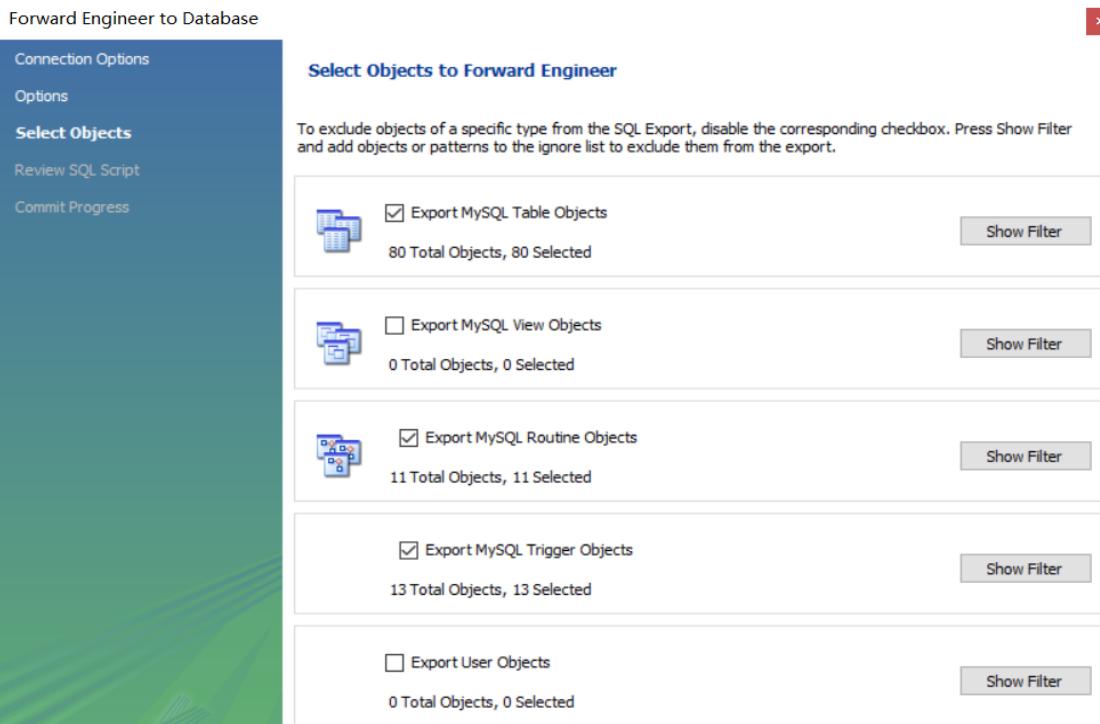
3.4.6 生成数据库

1) 点击菜单项“Database → Forward Engineer”：



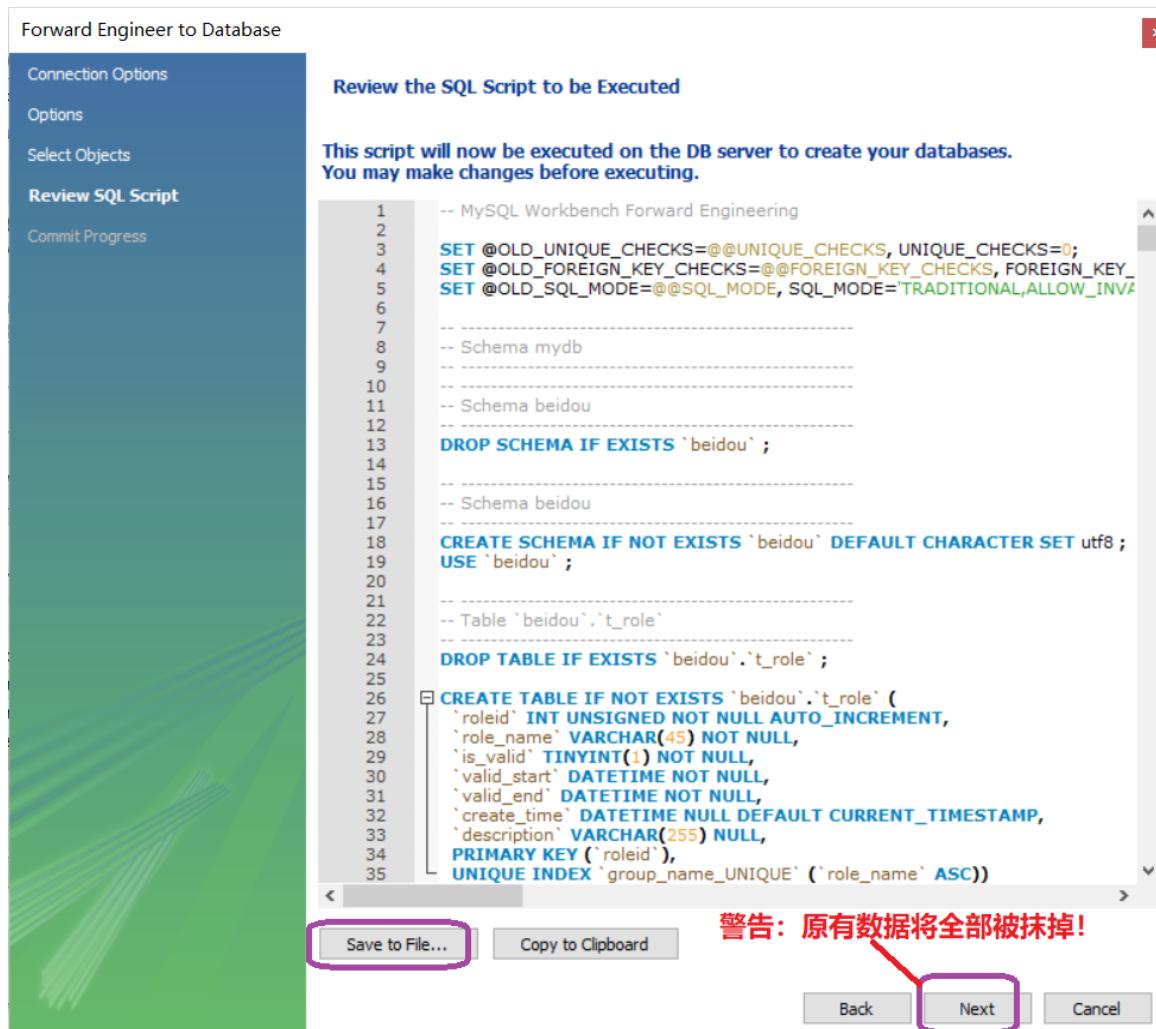
2) 选择导出选项。





3) 既可以生成 SQL 脚本文件，也可以直接写入数据库。

警告：当写入数据库时，数据库原有数据将被完全抹掉！因此在操作之前需要备份有用的数据。



3.5 数据管理

3.5.1 数据表分区

对于系统中的大表，可以按时间（每月）进行分区。而 Mysql 的表分区由如下限制：

- 1) 表定义中不允许包含或者引用外键：

<https://dev.mysql.com/doc/refman/5.7/en/partitioning-limitations.html>

- 2) 表中所有唯一键必须包含所有的分区键：

<https://dev.mysql.com/doc/refman/5.7/en/partitioning-limitations-partitioning-keys-unique-keys.html>

因此大表的定义需要调整：

- 1) 删除外键和外键引用
- 2) 修改主键，把分区键（时间）加到主键里。

脚本“bd_adjust_schema.sql”正是为了这个目的。

但是这个脚本应该在利用 NetBeans 模板技术自动生成 MVC 各层文件以后再执行。开发过程如下：

- 1) 在数据库设计时仍然应该定义大表的外键和外键引用
- 2) 导出数据库定义“bd_schema_sp.sql”，并创建数据库。
- 3) 利用 NetBeans 的 Wizards 自动生成 Entity Beans、Session Beans、JSP Controller bean、和其它文件。这样，MVC 文件中仍然包含自动生成的外键约束处理逻辑、而不必自己手写外键约束的代码。
- 4) 然后执行脚本“bd_adjust_schema.sql”以调整大表的定义、并且初始化它们的分区。
- 5) 再执行其它脚本，如“bd_data.sql”、“bd_events.sql”、“bd_triggers.sql”

结果是：虽然数据表最终没有外键，但 MVC 代码中实现了外键约束。这个过程实践证明可行。

脚本“bd_events.sql”和“bd_schema_sp.sql”包含了事件和存储过程，以每月自动对大表创建新的分区。

3.5.2 “最终用户数据权限表”

用户访问任何资源时，系统都会先检查用户是否拥有权限。本系统中，用户的最终数据权限由以下过程产生：（参见《全息时空北斗服务运营平台-用户手册》）

- 1) 角色决定用户的权限范围，即普通用户只能访问被授权的终端、管理员可以访问所有终端。
- 2) 普通用户的数据权限是其单独数据权限加上其所属各个用户组的数据权限。
- 3) 用户组形成树状层次关系、用户组自动拥有其所有子用户组的所有数据权限。

这是一个麻烦的逻辑，而组层次更是造成递归判断。于是，本系统采用以下策略：

- 1) 定义存储过程：基于各表的当前数据，按照以上逻辑，生成内部数据“最终用户数据权限表”。
- 2) 系统初始化时，调用该存储过程、产生初始的“最终用户数据权限表”。
- 3) 定义触发器，使得用户、组、权限发生变化时，调用该存储过程、更新“最终用户数据权限表”。
- 4) 系统直接利用“最终用户数据权限表”来检查用户权限。

这样，相对稳定的“最终用户数据权限表”可以避免频繁的多表查询。

脚本“bd_triggers.sql”和“bd_schema_sp.sql”包含了系统触发器和存储过程的定义。

3.5.3 配置主从数据库

参见《全息时空运营平台-运行环境的安装与配置手册》。

查看 用户反馈	
数据编号	455
用户反馈的对象	数据
用户反馈类型	问题报告
标题	从库执行事件时错误导致数据同步停止
描述	从库的事件应当由主库复制，以保障从库不再重复执行事件。本次错误是由于从库文件是由主库直接复制过来的，包含了已创建好的事件，因此造成同一事件分别在主从上定义而重复执行。要避免错误，要么构建好从库以后再在主库定义事件、要么把从库上重复定义的事件删除或者关闭。
报告者	任珊虹
报告时间	2018-01-04 13:23:25
响应说明	删除从库上的事件
响应时间	2018-01-04 13:29:11
响应者	任珊虹
用户反馈状态	已解决
用户反馈的来源	网页客户端
解决的版本	-

3.5.4 常用 mysql 命令 - win

初始化系统数据库，导入表结构、存储过程、初始化数据、事件、触发器（必须这个顺序）：

```
mysql -u root -p"密码" < bd_schema_sp.sql
mysql -u root -p"密码" beidou < bd_adjust_schema.sql
mysql -u root -p"密码" beidou < bd_data.sql
mysql -u root -p"密码" beidou < bd_events.sql
mysql -u root -p"密码" beidou < bd_triggers.sql
```

导出所有表结构和过程（无数据、无 triggers）

```
mysqldump -u root -p"密码" -d beidou --set-gtid-purged=OFF --skip-triggers -R > bd_schema_sp.sql
```

导出所有数据（无表结构、无 triggers）

```
mysqldump -u root -p"密码" -t beidou --set-gtid-purged=OFF --skip-triggers -c > bd_data.sql
```

导出特定表的数据（无表结构、无 triggers）

```
mysqldump -u root -p"密码" -t beidou --set-gtid-purged=OFF --no-create-info --skip-triggers 表名 > table_data.sql
```

导出所有 Triggers（无表结构、无数据）

```
mysqldump -u root -p"密码" -d beidou --set-gtid-purged=OFF --no-create-info > bd_triggers.sql
```

导入数据:

```
mysql -u root -p"密码" beidou < bd_data-99.sql
```

导出 binlog:

```
mysqlbinlog -v --base64-output=DECODE-ROWS "C:\ProgramData\MySQL\MySQL Server 5.7\Data\mysql-bin.000003" > d:\tmp\1\log.txt
```

登录复制库:

```
mysql --port=3307 -u root -p"密码" beidou
```

登录历史库:

```
mysql --port=3308 -u root -p"密码" beidou
```

添加/删除系统服务:

```
mysqld --install MySQL57-his --defaults-file="D:\ProgramsData\MySQL Server 5.7 - his\my.ini"  
mysqld --remove mysql57-his
```

启停系统服务（必须以管理员身份打开命令窗口来执行）

```
net stop MySQL57-repl  
net stop MySQL57-his  
net stop MySQL57
```

```
net start MySQL57  
net start MySQL57-repl  
net start MySQL57-his
```

3.5.5 常用 mysql 命令 - Linux

初始化系统数据库，导入表结构、存储过程、初始化数据、事件、触发器（必须这个顺序）：

```
mysql -u root -p 密码 < bd_schema_sp.sql  
mysql -u root -p 密码 beidou < bd_adjust_schema.sql  
mysql -u root -p 密码 beidou < bd_data.sql  
mysql -u root -p 密码 beidou < bd_events.sql  
mysql -u root -p 密码 beidou < bd_triggers.sql
```

导出所有表结构和过程（无数据、无 triggers）

```
mysqldump -uroot -p 密码 -d beidou --set-gtid-purged=OFF --skip-triggers -R > bd_schema_sp.sql
```

导出所有数据（无表结构、无 triggers）

```
mysqldump -uroot -p 密码 -t beidou --set-gtid-purged=OFF --skip-triggers -c > bd_data.sql
```

导出特定表的数据（无表结构、无 triggers）

```
mysqldump -uroot -p 密码 -t beidou --set-gtid-purged=OFF --no-create-info --skip-triggers 表名 > table_data.sql
```

导出所有 Triggers（无表结构、无数据）

```
mysqldump -uroot -p 密码 -d beidou --set-gtid-purged=OFF --no-create-info > bd_triggers.sql
```

导入数据：

```
mysql -uroot -p 密码 beidou < bd_data-99.sql
```

登录复制库：

```
mysql --host=127.0.0.1 --port=3307 -uroot -p 密码 beidou
```

登录历史库：

```
mysql --host=127.0.0.1 --port=3308 -uroot -p 密码 beidou
```

启停 linux 系统服务（必须以 root 来执行）

```
systemctl stop mysqld@repl
```

```
systemctl stop mysqld@his
```

```
systemctl stop mysqld
```

```
systemctl start mysqld
```

```
systemctl start mysqld@repl
```

```
systemctl start mysqld@his
```

3.5.6 常用 mysql 语句

检查字符集：

```
show variables like 'character_set%';
```

```
show variables like 'collation_%';
```

修改密码：

```
set password for 'root'@'localhost'=password('密码');
```

查看表定义：

```
show create table t_location\G
```

增删改数据表字段:

```
alter table t_location drop column is_history;
alter table t_location add column repeats INT UNSIGNED DEFAULT 1;
alter table t_location add column last_record DATETIME DEFAULT CURRENT_TIMESTAMP;
alter table t_location add INDEX `tl_sourceid_idx`(`data_source` ASC);
alter table t_location add CONSTRAINT `tl_sourceid`  
    FOREIGN KEY(`data_source`)  
    REFERENCES `beidou`.`t_location_source`(`sourceid`)  
    ON DELETE SET NULL  
    ON UPDATE SET NULL;  
alter table t_test_execution_feedback modify column is_success boolean NOT NULL default false;
```

开启/关闭事件:

```
set GLOBAL event_scheduler=1;
```

查看过程、触发器、事件:

```
SHOW PROCESSLIST;  
SELECT name from mysql.proc where db="beidou";  
show triggers\G  
show variables like 'event_scheduler';  
show events\G
```

查看数据表分区:

```
select TABLE_SCHEMA, TABLE_NAME, PARTITION_NAME, PARTITION_DESCRIPTION, UPDATE_TIME,  
TABLESPACE_NAME from INFORMATION_SCHEMA.partitions where TABLE_SCHEMA="beidou";  
SELECT PARTITION_NAME, TABLE_ROWS FROM INFORMATION_SCHEMA.PARTITIONS WHERE  
TABLE_NAME='t_try';
```

检查语句分区:

```
explain partitions select * from t_try;  
explain select count(*) from t_location where record_time > "2017-10-21";
```

定义数据表分区:

```
alter table t_try ADD PARTITION (PARTITION p2018_02 VALUES LESS THAN ('2018-3-01'));  
alter table t_try drop PARTITION;  
ALTER TABLE t_try REORGANIZE PARTITION n0 INTO (  
    PARTITION p1 VALUES LESS THAN (TO_DAYS('2017-11-02')),  
    PARTITION p3 VALUES LESS THAN (TO_DAYS('2017-11-03'))  
);  
ALTER TABLE t_try REORGANIZE PARTITION p0, p1, p3 INTO (
```

PARTITION n0 VALUES LESS THAN (TO_DAYS('2017-11-03'))

);

查看 binlog:

show binlog events\G;

show binlog events in 'mysql-bin.000056' from 3 limit 2,5\G;

执行脚本:

source myscript.sql

设置主从关系:

change master to master_host='localhost', master_user='repl', master_password='密码', master_auto_position=1;

避免主库事件重复执行:

alter event e_add_partition_monthly ON COMPLETION PRESERVE DISABLE;

3.6 JavaEE 开发环境 NetBeans

3.6.1 资源地址

官网地址：（最好下载所有包）

<https://netbeans.org/downloads/>

NetBeans IDE 8.2 下载

8.1 | 8.2 | 开发版 | 早期版

电子邮件地址 (可选) :

语言: 简体中文 平台: Windows

订阅新闻邮件: 每月 每周
 NetBeans 可使用此地址联系我

请注意: 在此平台上不支持变成灰色的技术。

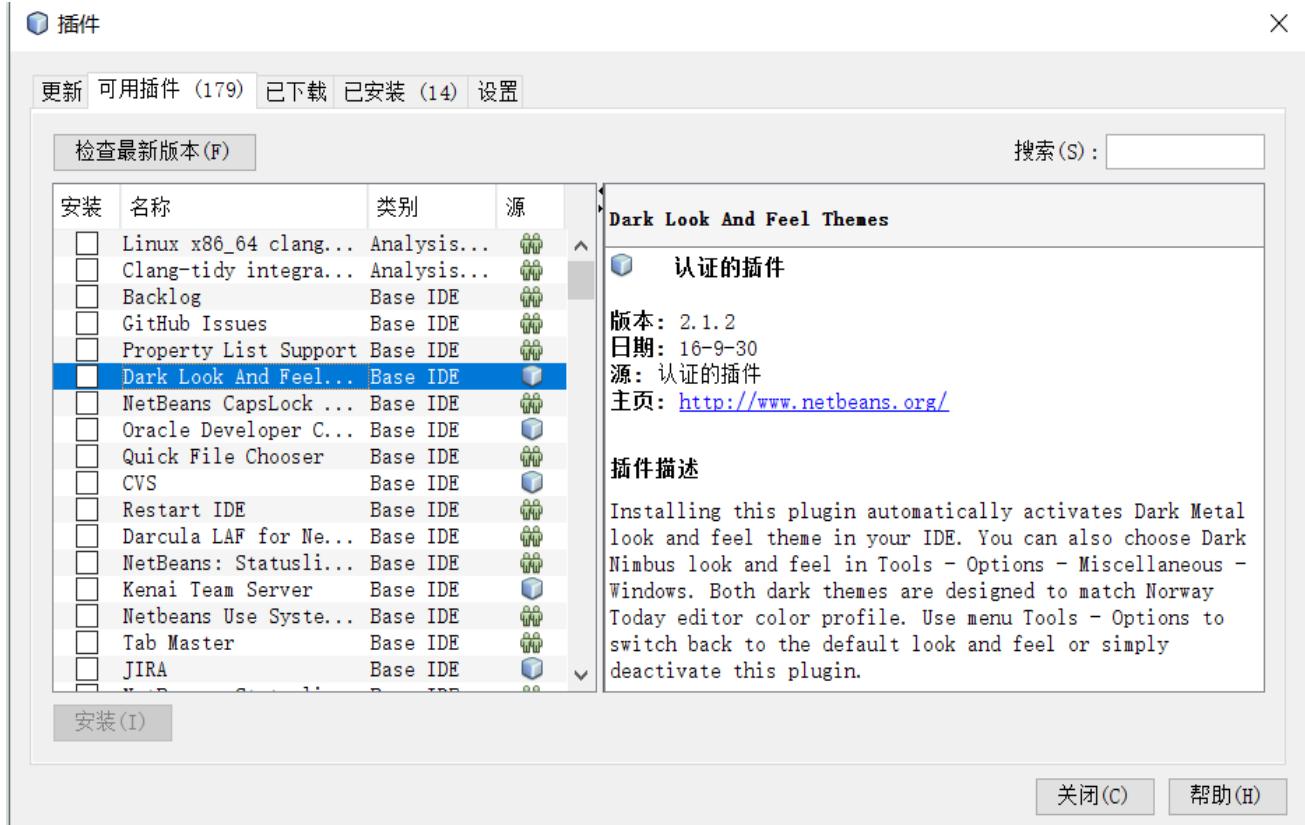
所支持的技术 *	Java SE	Java EE	HTML5/JavaScript	PHP	C/C++	All
NetBeans 平台 SDK	●	●				●
Java SE	●	●				●
Java FX	●	●				●
Java EE		●				●
Java ME						●
HTML5/JavaScript	●		●	●		●
PHP			●	●		●
C/C++					●	●
Groovy						●
Java Card(tm) 3 Connected						●
绑定的服务器						
GlassFish Server Open Source Edition 4.1.1		●				●
Apache Tomcat 8.0.27	●					●

下载 下载 下载 x86 下载 x86 下载 x86 下载 x64 下载 x64 下载 下载

免费, 95 MB 免费, 197 MB 免费, 108 - 112 MB 免费, 108 - 112 MB 免费, 107 - 110 MB 免费, 221 MB

3.6.2 下载插件

菜单: 工具->插件



3.6.3 改变界面语言

下载的 NetBeans 若是中文版的，则操作界面都是中文且没有提供更改的功能。如果需要改成英文，以下是网上给出的方法，实践有效：

找到 %NetBeans_HOME%\etc 目录下 netbeans.conf 配置文件，修改 netbeans_default_options 配置项，增加-J-Duser.language=en -J-Duser.country=US，修改成：

```
netbeans_default_options="-J-client -J-Xss2m -J-Xms32m -J-Dapple.laf.useScreenMenuBar=true -J-Dapple.awt.graphics.UseQuartz=true -J-Dsun.java2d.nodraw=true -J-Dsun.java2d.dpiaware=true -J-Dsun.zip.disableMemoryMapping=true -J-Duser.language=en -J-Duser.country=US"
```

重新启动，就变成英文界面了。

3.6.4 修改默认 JDK

以下是网上给出的方法，实践有效。

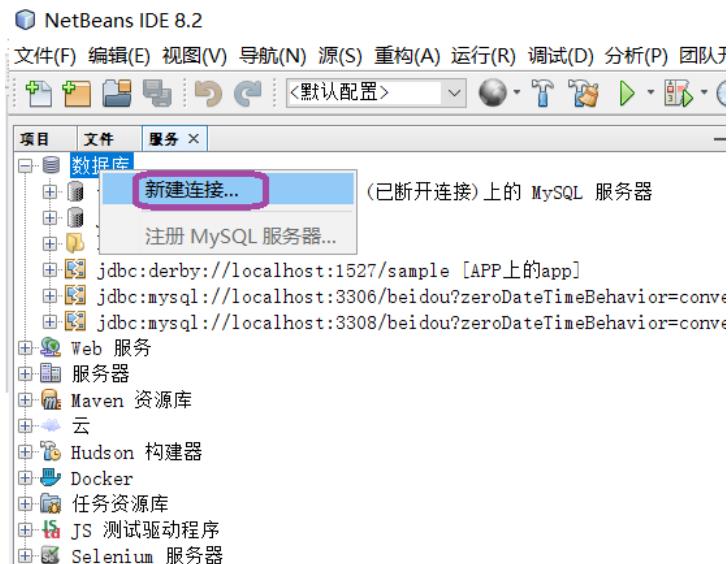
Netbeans 默认的 JDK 是在安装时指定的，如果因为卸载或者更换 JDK 发生了路径变化，会引起 Netbeans 无法启动，这时就需要去 Netbeans 安装目录下找到 etc 文件夹，里面有个 netbeans.conf 文件。打开这个文件找到：

```
# default location of J2SE JDK, can be overridden by using --jdkhome <dir> switch
netbeans_jdkhome="实际的 JDK 路径"
```

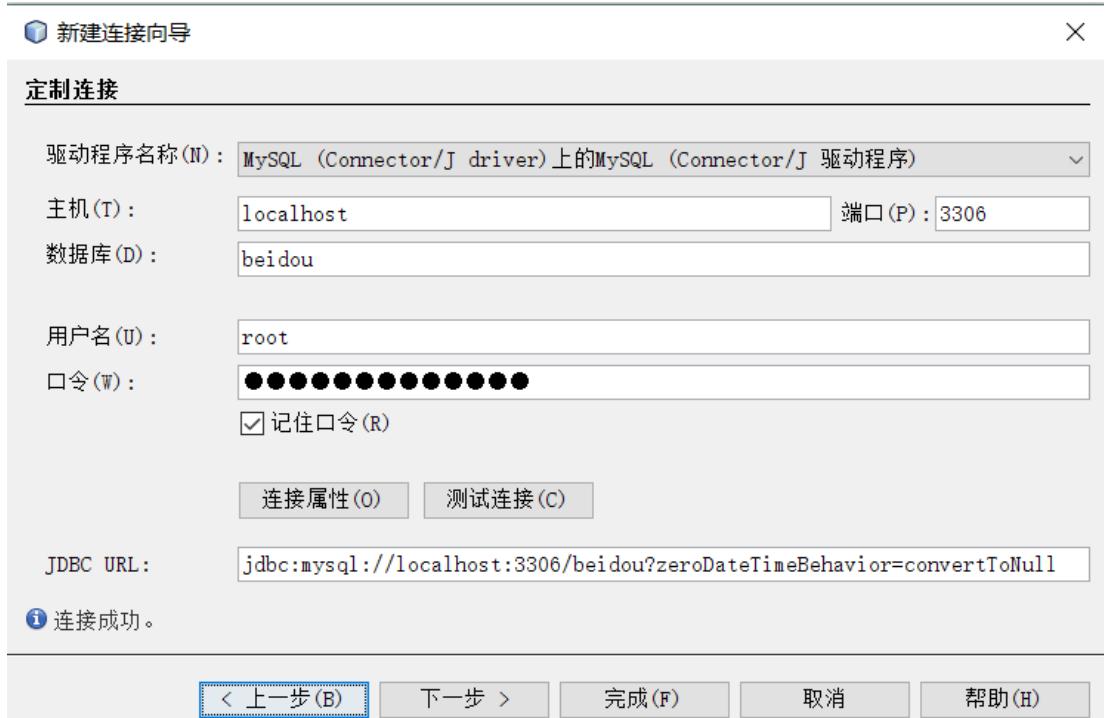
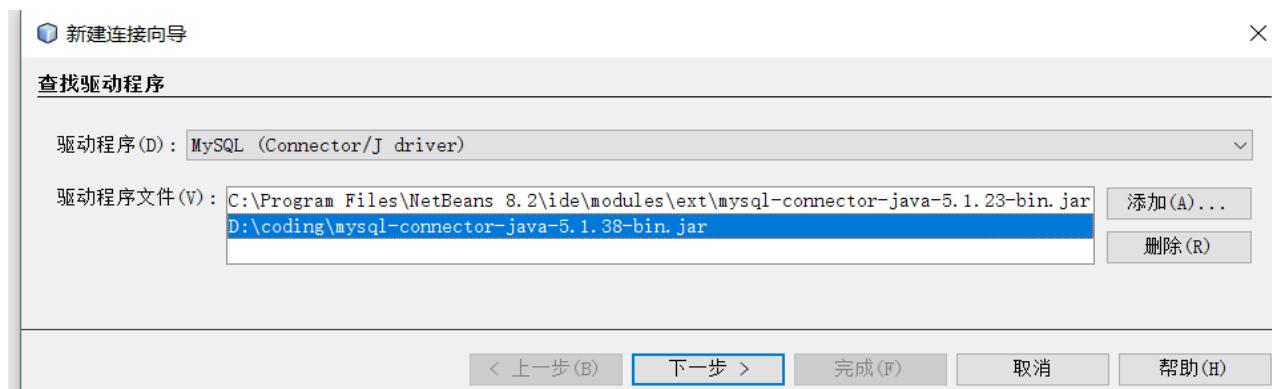
修改 netbeans_jdkhome 的值，启动 Netbeans 就可以了。

3.6.5 连接数据库

1) 在“服务”视图中，右键点击“数据库”选择“新建连接...”：

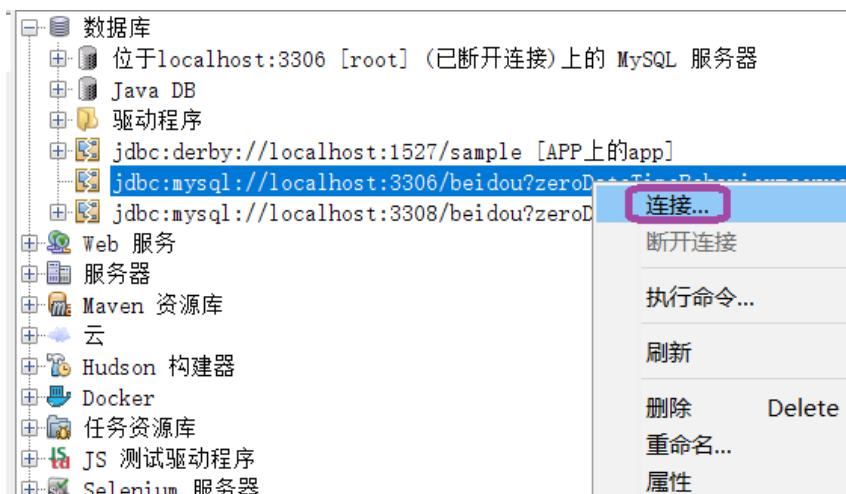


2) 设置 mysql 的 jdbc 驱动程序。既可以用 NetBeans 自带的，也可以用自己下载的：



3) 填写连接参数、点击按钮“测试连接”以验证参数。然后点击按钮“完成”。

4) 右键点击创建好的连接名、选择“连接...”：



5) 连接后，右键点击数据库名“beidou”、选择“设置为默认目录”：



6) 右键点击数据表名、选择“查看数据”，则查询 SQL 语句自动显示在右边、并且执行结果也显示出来。

#	dataid	response_beam	time_beam	beam1	beam2	beam3	beam4
1		1		5	4	4	3
2		2		5	4	4	2
3		3	1	5	4	4	2
4		4	1	5	4	4	2

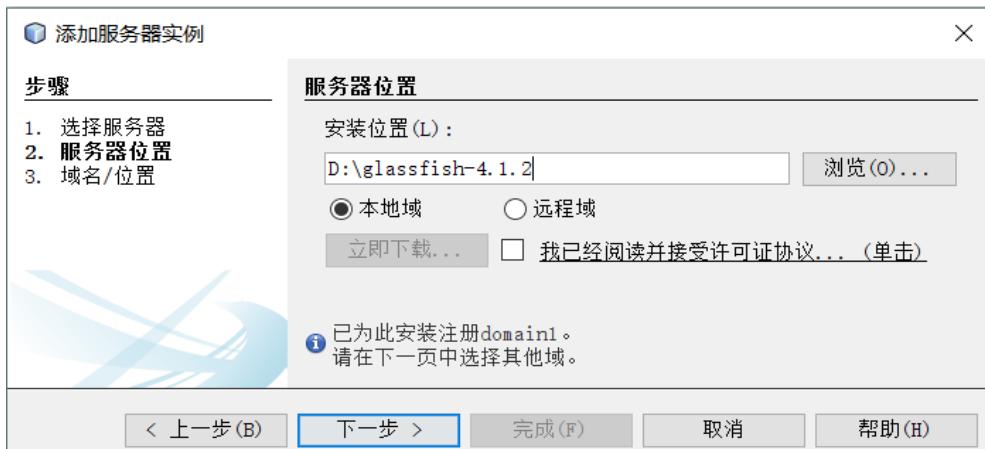
在 0 秒内成功执行。
提取结果集用时 0.053 秒。
第 1 行, 第 1 列
执行在 0.253 秒后完成, 出现 0 个错误。

3.6.6 管理服务器

1) 在“服务”视图中，右键点击“服务器”、选择“添加服务器”。可以选择多种服务器



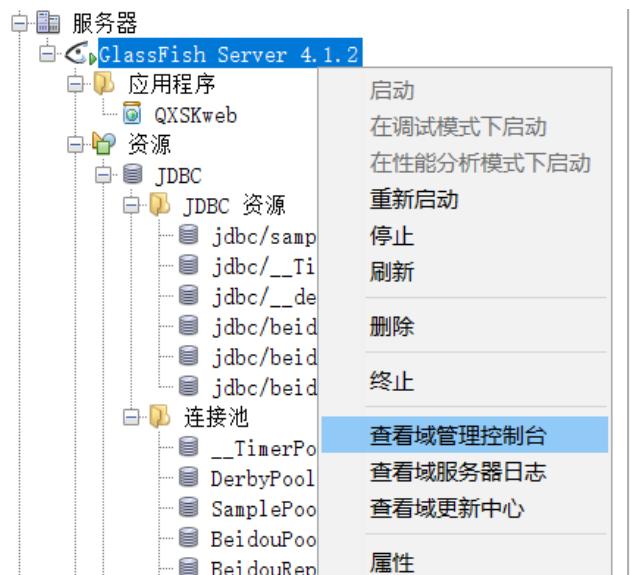
2) 选择服务器目录。既可以选择本地已安装的、也可以直接下载:



3) 输入自己的域名、其它参数默认即可:



4) 右键点击已创建的服务器，选择管理命令来执行：



5) 利用“域管理控制台”方便地操作服务器资源：

Select	Name	Port #	Address	Enabled
<input type="checkbox"/>	admin-listener	<code>\$(ASADMIN_LISTENER_PORT)</code>	0.0.0.0	true
<input type="checkbox"/>	http-listener-1	<code>\$(HTTP_LISTENER_PORT)</code>	0.0.0.0	true
<input type="checkbox"/>	http-listener-2	<code>\$(HTTP_SSL_LISTENER_PORT)</code>	0.0.0.0	true

3.7 JavaEE 项目

3.7.1 开发示例

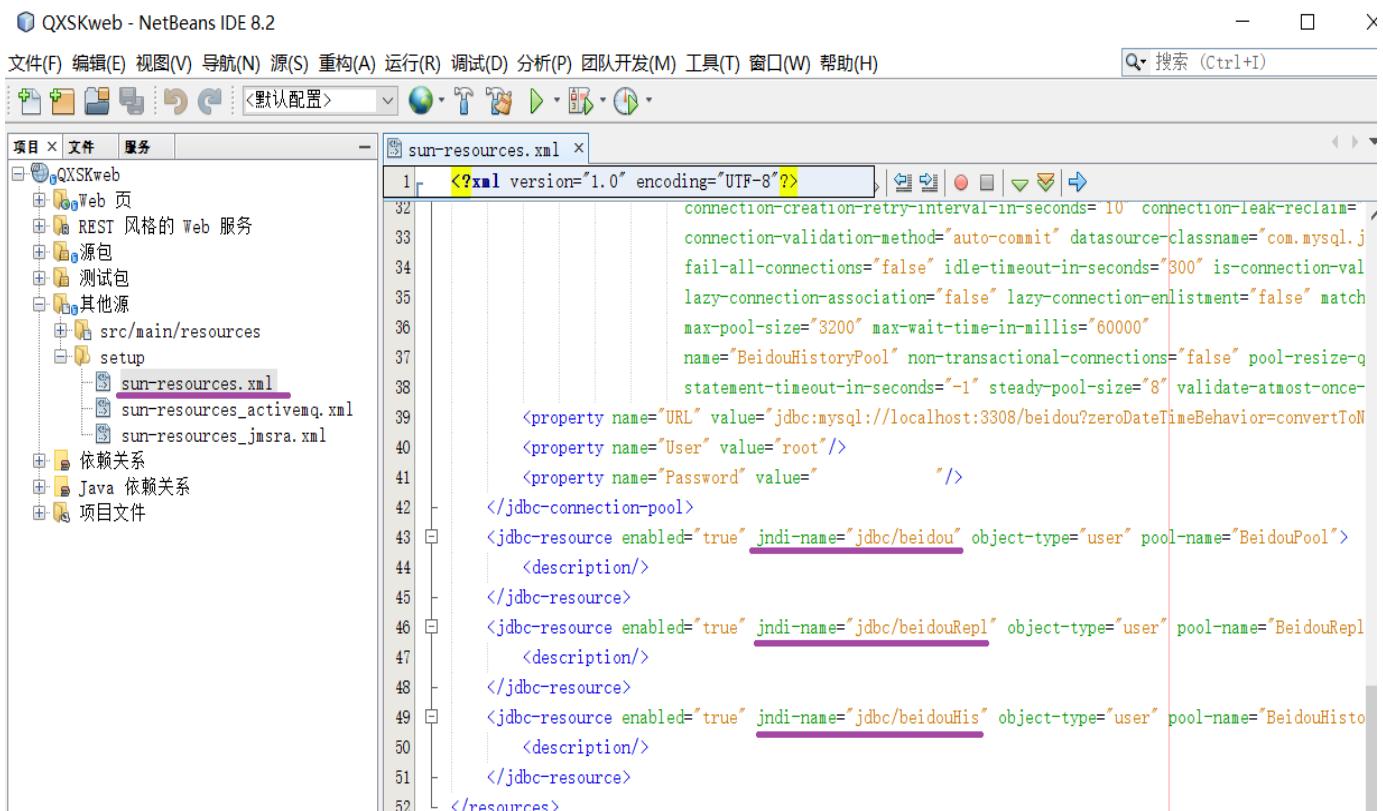
官网地址给出了详实、标准的开发示例：

<https://netbeans.org/kb/trails/java-ee.html>

按照官方教程学习，可以帮助理解 JavaEE 各方面的重要概念。

3.7.2 注册外部资源

服务器的外部资源，如 JDBC 连接池、JNDI、JMS 连接器，都已在文件“setup/sun-resources.xml”中定义好了。



The screenshot shows the NetBeans IDE interface with the title bar "QXSKweb - NetBeans IDE 8.2". The menu bar includes "文件(F) 编辑(E) 视图(V) 导航(N) 源(S) 重构(A) 运行(R) 调试(D) 分析(P) 团队开发(M) 工具(T) 窗口(W) 帮助(H)". The search bar at the top right says "搜索 (Ctrl+I)". The left sidebar shows the project structure under "QXSKweb": Web 页, REST 风格的 Web 服务, 源包, 测试包, 其他源 (src/main/resources, setup), 依赖关系, Java 依赖关系, and 项目文件. The main editor window displays the XML file "sun-resources.xml" with the following content:

```

<?xml version="1.0" encoding="UTF-8"?>
<resources>
    <jdbc-connection-pool connection-creation-retry-interval-in-seconds="10" connection-leak-reclaim="true" connection-validation-method="auto-commit" datasource-classname="com.mysql.jdbc.Driver" fail-all-connections="false" idle-timeout-in-seconds="300" is-connection-validation-required="false" lazy-connection-association="false" lazy-connection-enlistment="false" max-concurrent-user-count="3200" max-wait-time-in-millis="60000" max-pool-size="3200" max-wait-time-in-millis="60000" name="BeidouHistoryPool" non-transactional-connections="false" pool-resize-qstatement-timeout-in-seconds="-1" steady-pool-size="8" validate-atmost-once="true">
        <property name="URL" value="jdbc:mysql://localhost:3308/beidou?zeroDateTimeBehavior=convertToNull" />
        <property name="User" value="root" />
        <property name="Password" value="" />
    </jdbc-connection-pool>
    <jdbc-resource enabled="true" jndi-name="jdbc/beidou" object-type="user" pool-name="BeidouPool">
        <description/>
    </jdbc-resource>
    <jdbc-resource enabled="true" jndi-name="jdbc/beidouRepl" object-type="user" pool-name="BeidouRepl">
        <description/>
    </jdbc-resource>
    <jdbc-resource enabled="true" jndi-name="jdbc/beidouHis" object-type="user" pool-name="BeidouHistoryPool">
        <description/>
    </jdbc-resource>
</resources>

```

在 NetBeans 中构建项目代码时，这个资源文件将被自动执行、从而资源被自动注册在服务器中。

也可以执行以下命令来注册资源：（注意文件中密码应该正确设置。参见《全息时空运营平台-运行环境的安装与配置手册》）

`asadmin add-resources sun-resources.xml`

由于使用的是 ActiveMQ 并且是 client 直连模式，因此 JMS 资源不必提前注册，而是程序中生成。（代码包中另两个资源文件是注册模式的，可供参考。）

3.7.3 使用 Git 支持

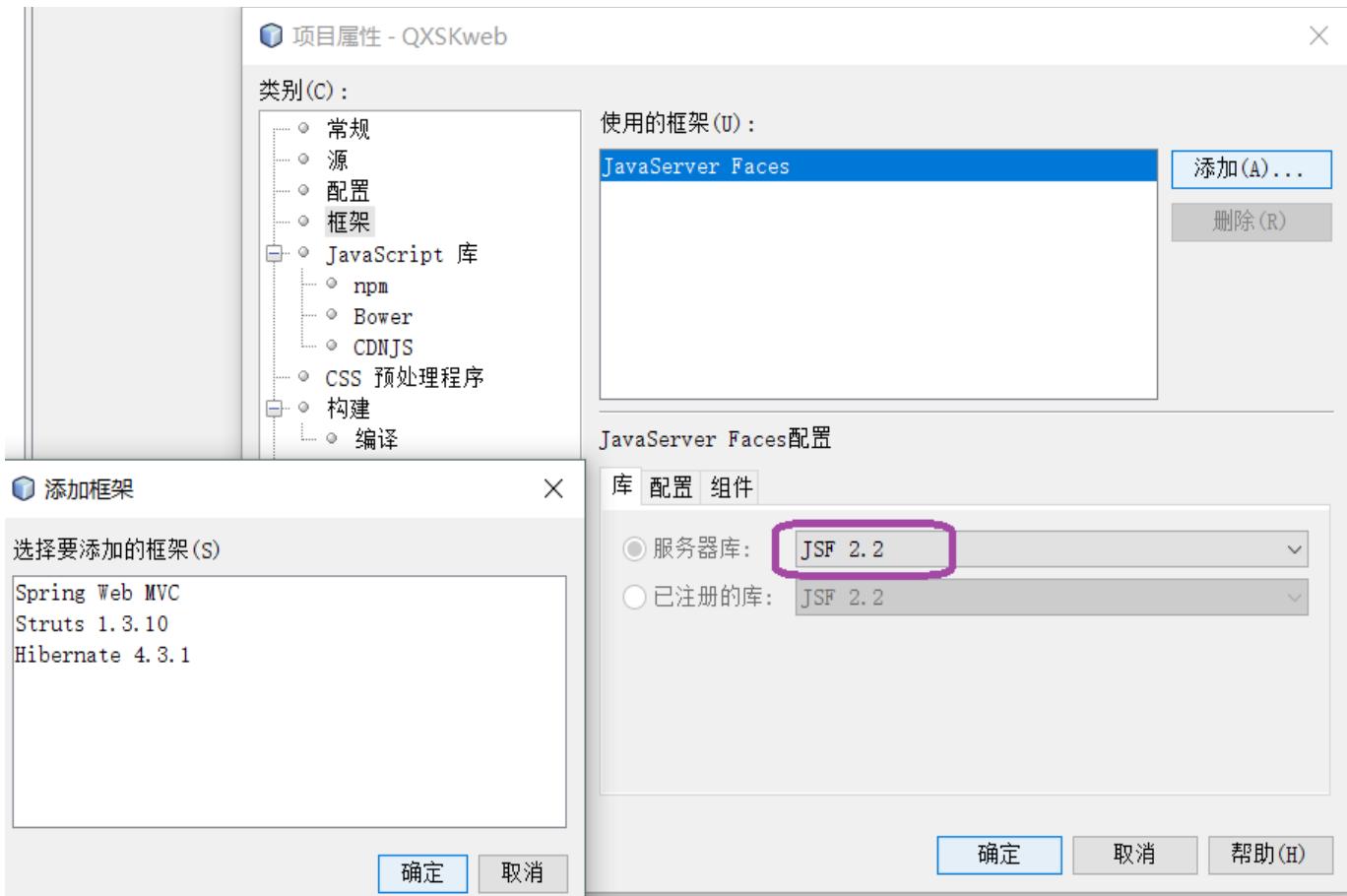
参见：https://netbeans.org/kb/docs/ide/git_zh_CN.html

3.7.4 使用 maven 支持

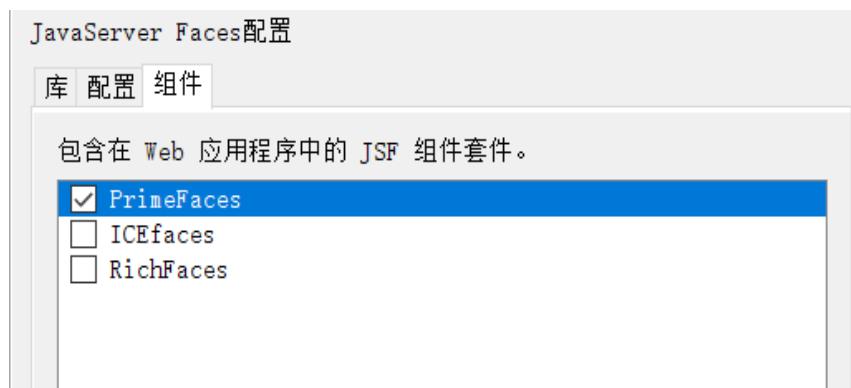
参见：<https://netbeans.org/kb/docs/javaee/maven-entapp.html>

3.7.5 选择 MVC 框架

- 1) 右键点击项目名、选择“属性”、点击“框架”。
- 2) 确保“使用的框架”是“JavaServer Faces”。可选的三个框架本系统均不采用，因此不必添加。
- 3) 确保所选库为“JSF 2.2”：

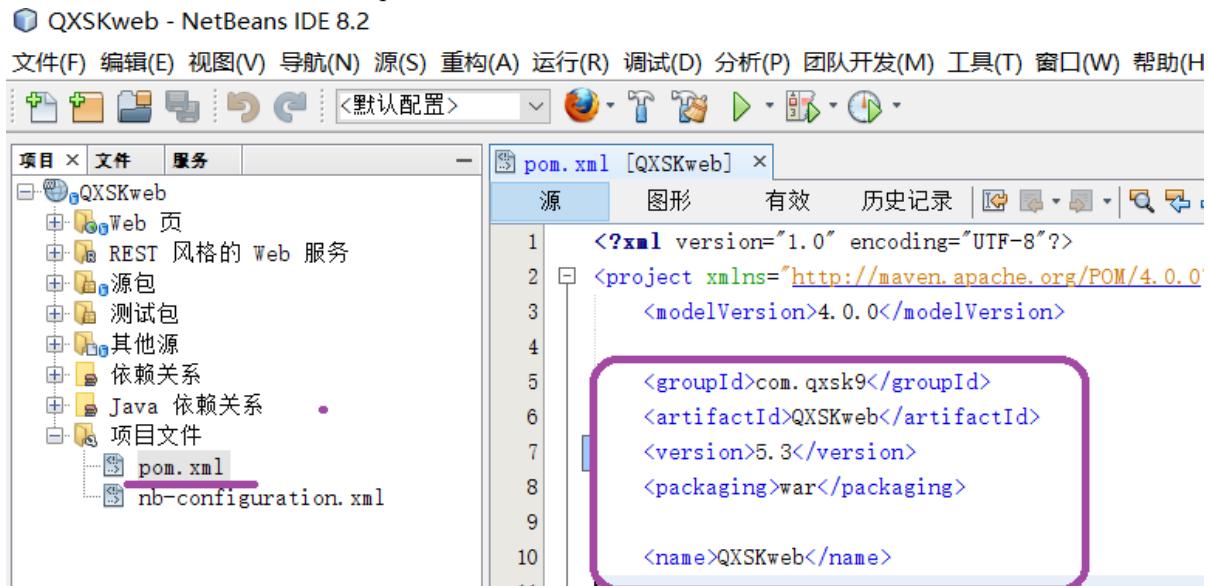


- 4) 确保组件选的是“PrimeFaces”：



3.7.6 设置系统标识和版本

系统的名称、组编号、版本需要在 pom.xml 编辑：

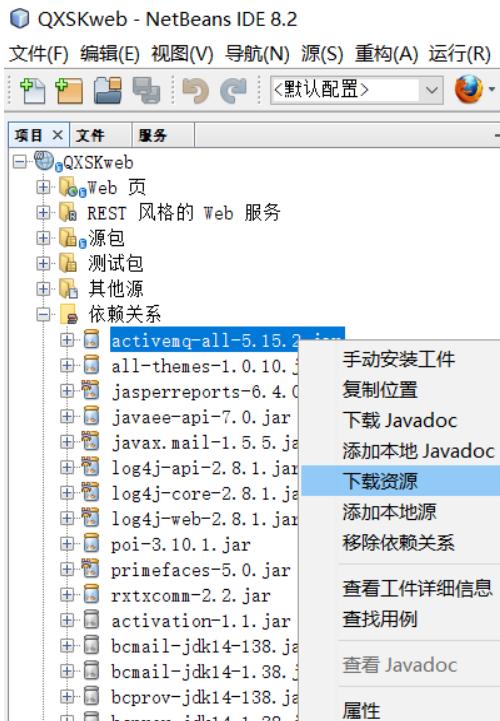


3.7.7 下载依赖

1) 在 pom.xml 文件已定义了系统依赖的外部库。Active MQ 和 jasperreports 都是既可以作为外部独立的服务器提供功能，也可以作为插件嵌入到应用系统中，本系统均采用后者。因此不需要下载安装 Active MQ 介质和 Jasper Server 的介质，只需要在应用的 pom.xml 中写上依赖即可。

2) 在项目资源树中，“依赖关系”显示了 pom.xml 文件中声明的依赖以及所有依赖的依赖。只有当依赖资源都已下载到本地 maven 仓库中，系统才可能被构建成功。

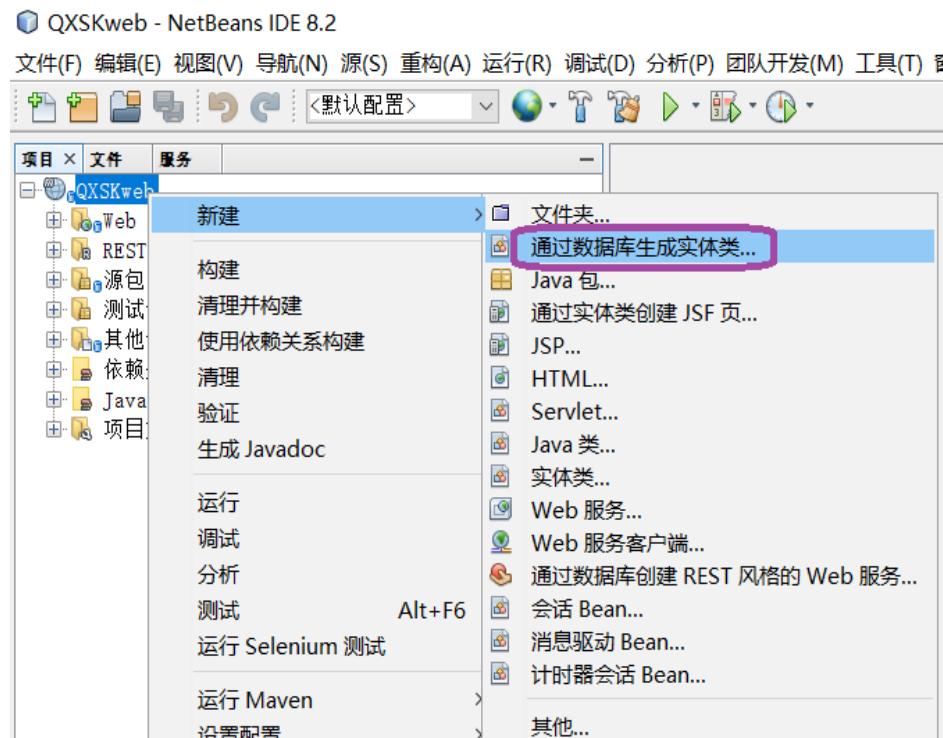
3) 当所依赖的资源不在 maven 仓库中，会显示黄色问号在资源名上。可以重构系统来启动资源的自动下载。也可以右键点击资源名手动下载：



3.8 开发 Entity Bean

3.8.1 利用 Wizard 自动生成 Entity Bean

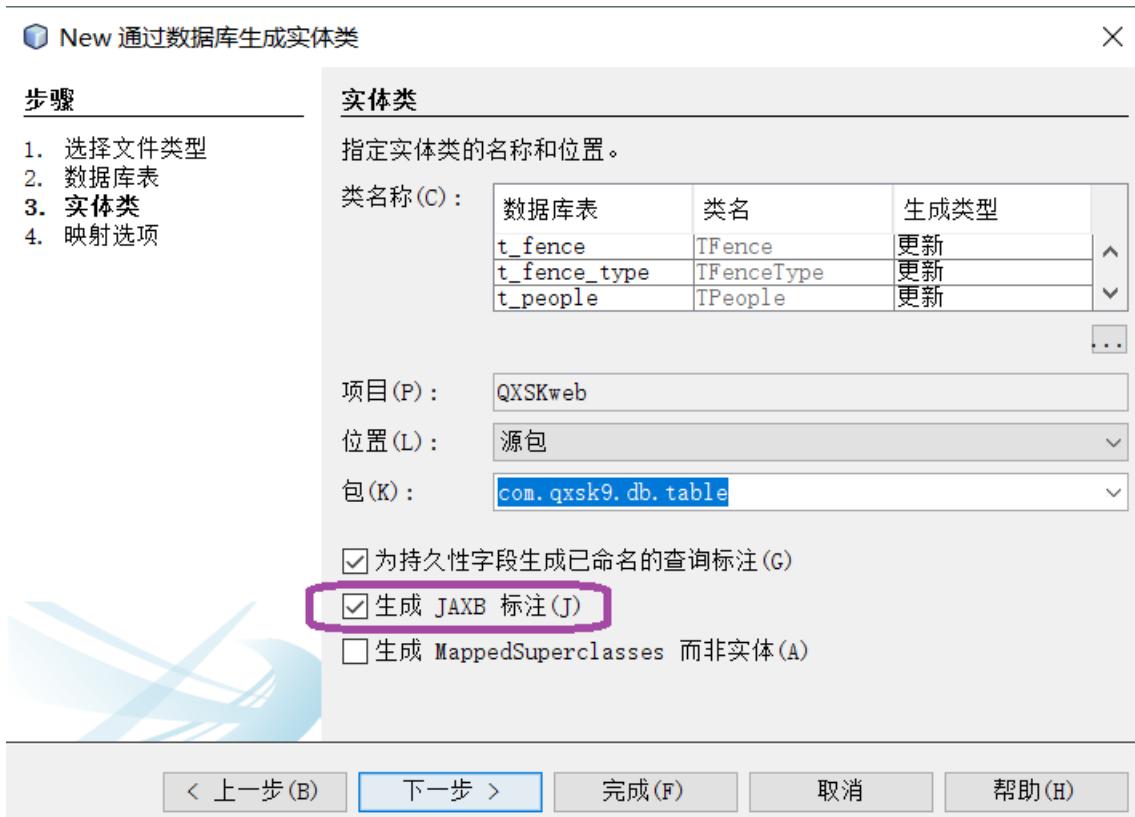
1) 右键点击项目名、选择“新建”-“通过数据库生成实体类”：



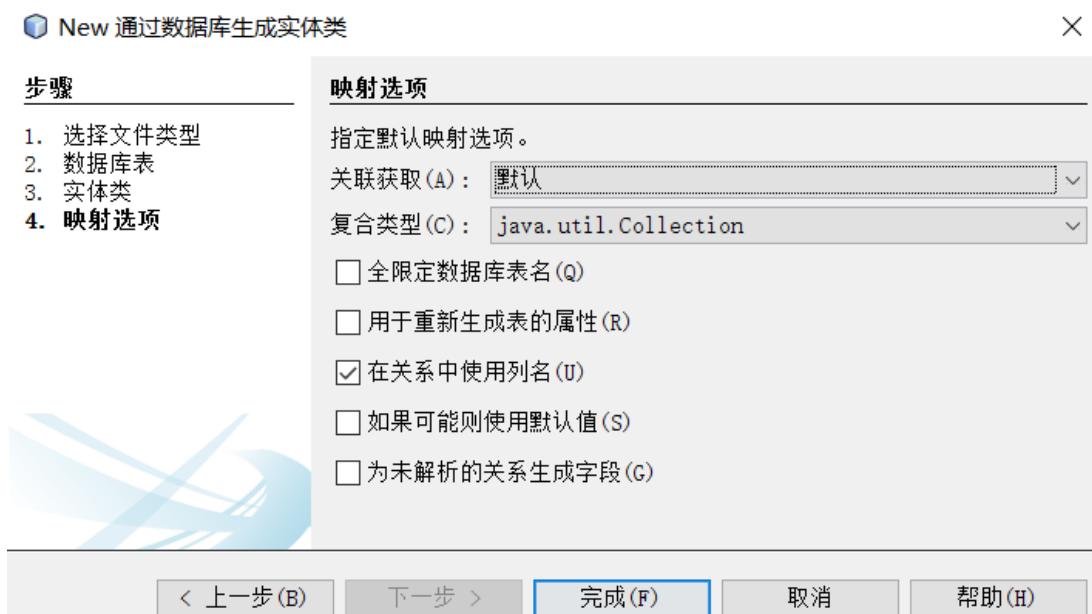
2) 选择数据源、选择数据表：（通常是选择全部可用表）



3) 选择参数、输入包名:



4) 选择映射参数、完成代码生成:



5) 本系统所有自动生成的 Entity Bean 均在包 com.qxsk9.db.table 中。

The screenshot shows the NetBeans IDE interface. The title bar says "QXSKweb - NetBeans IDE 8.2". The menu bar includes "文件(F)" (File), "编辑(E)" (Edit), "视图(V)" (View), "导航(N)" (Navigate), "源(S)" (Source), "重构(A)" (Refactor), "运行(R)" (Run), "调试(D)" (Debug), "分析(P)" (Analyze), "团队开发(M)" (Team Development), "工具(T)" (Tools), "窗口(W)" (Windows), and "帮助(H)" (Help). The toolbar has icons for file operations like Open, Save, and Build. The left sidebar shows the project structure under "QXSKweb": "Web 页", "REST 风格的 Web 服务", and "源包" (Source Packages) containing various sub-packages like "com.qxsk9.bd", "com.qxsk9.bd.bean", etc. The right pane shows the code editor for "TBdLog.java". The code is annotated with JPA annotations:

```

29  @Entity
30  @Table(name = "t_bd_log")
31  @XmlRootElement
32  @NamedQueries({
33      @NamedQuery(name = "TBdLog.findAll", query = "SELECT t
34          , @NamedQuery(name = "TBdLog.findByDataid", query = "SE
35          , @NamedQuery(name = "TBdLog.findByIsSent", query = "SE
36          , @NamedQuery(name = "TBdLog.findByIsBd2", query = "SEL
37          , @NamedQuery(name = "TBdLog.findByCommandType", query
38          , @NamedQuery(name = "TBdLog.findByBdStatement", query
39          , @NamedQuery(name = "TBdLog.findByRecordTime", query =
40          , @NamedQuery(name = "TBdLog.findByPort", query = "SEL
41  public class TBdLog implements Serializable {
42
43      private static final long serialVersionUID = 1L;
44      @Id
45      @GeneratedValue(strategy = GenerationType.IDENTITY)
46      @Basic(optional = false)
47      @Column(name = "dataid")
48      private Integer dataid;
49      @Basic(optional = false)
50      @NotNull
51      @Column(name = "is_sent")
52      private boolean isSent;

```

6) 可以多次执行这个过程，以适应数据库发生的变化。

3.8.2 修正复合键

若数据表的主键是多个字段组成（复合键），则 PrimeFaces 数据表的行选择会失败：

<https://stackoverflow.com/questions/29880155/overriding-implementing-getrowkey-and-getrowdata-methods-when-there-is-a-composite-key>

解决办法如以上网页：对包 com.qxsk9.db.table 中所有名为 “*PK.java”的文件，替换方法 “toString()”中的逗号为下划线。例如自动生成的 “TUserIsRolePK.java”中：

```

public String toString() {
    return "com.qxsk9.db.table.TUserIsRolePK[ userid=" + userid + ", roleid=" + roleid + "]";
}

```

修改为：

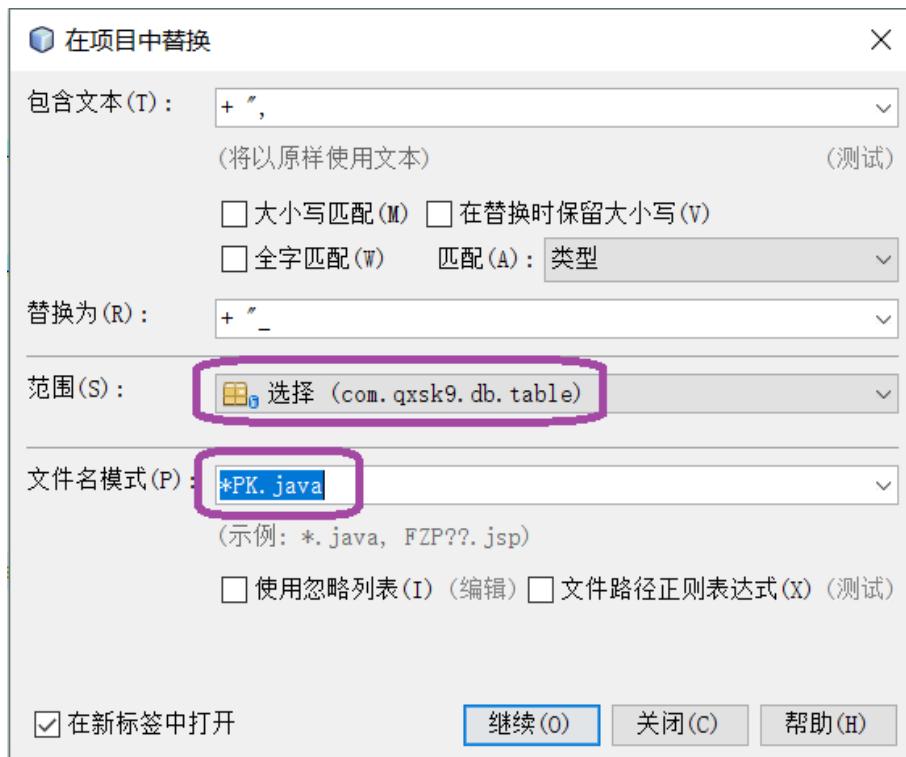
```

public String toString() {
    return "com.qxsk9.db.table.TUserIsRolePK[ userid=" + userid + "_roleid=" + roleid + "]";
}

```

批量修改的办法：

- 1) 选择菜单项“编辑 → 在项目中替换...”
- 2) 设置替换参数：注意被替换的字串最后包含一个空格



- 3) 在搜索结果里检查是否都为预期的语句，然后点击左下角按钮以实施替换：

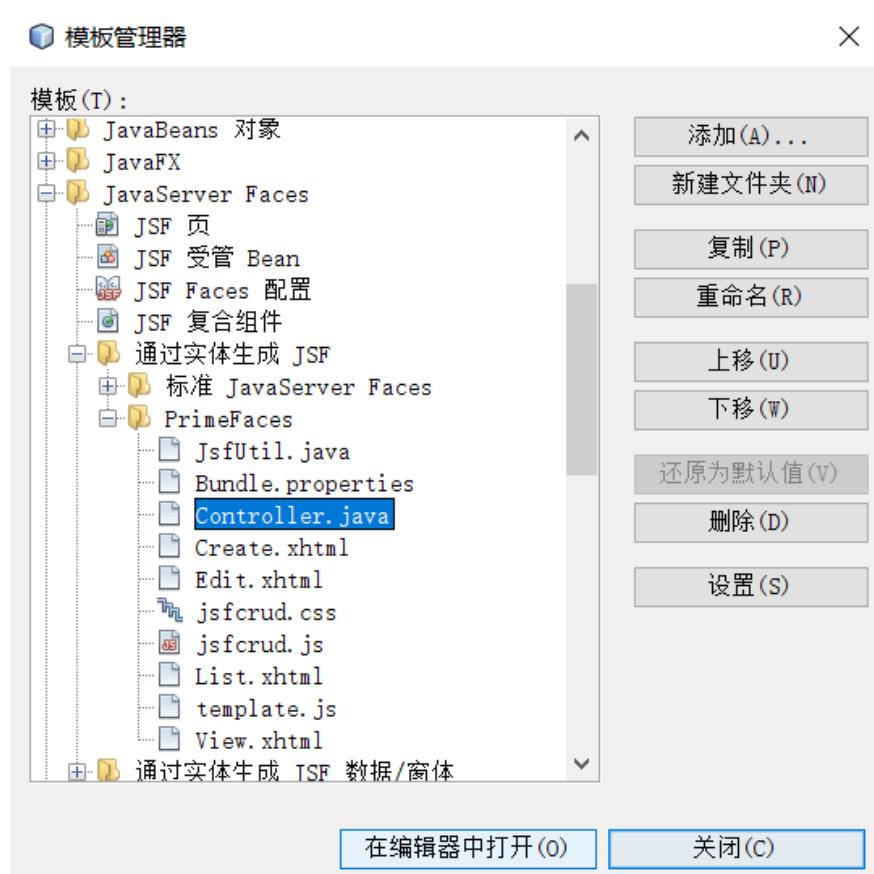


3.9 开发 MVC 文件

3.9.1 替换 PrimeFaces 模板

根据系统的需要，我修改了用以自动生成 MVC 文件的 PrimeFaces 模板。替换过程如下：

- 1) 选择菜单项“工具 → 模板”。
- 2) 打开层级“JavaServer Faces”-“通过实体生成 JSF”-“PrimeFaces”。
- 3) 选择击模板文件名，点击按钮“在编辑器中打开”
- 4) 将模板文件的内容全部换成源码包中目录“模板”下对应文件的内容。



3.9.1.1 “Controller.java”的修改

主要修改包含：

- 1) 变量从“private”改为“protected”，以便从生成的 bean 继承为定制的 bean。
- 2) 定义了“selectRows”，用以数据行的多选。
- 3) 定义了“needLoadItems”，取代“items == null”，用以重新查询数据。

3.9.1.2 “Create.xhtml” 和“Edit.xhtml”的修改

主要修改包含：

- 1) 对于自增字段添加特定模板：它不能输入、并被显示为“自动生成”。
- 2) 对于密码字段添加特定模板：它的标签为“p:password”。

3) 修改时间字段的模板：定义时间字段的模式（pattern）、时区（timeZone）和语言（locale），并且日历的格式也有了一点变化。

4) 对于字段"createTime" and "recordTime"添加特定模板：它不能被输入、而是将被 bean 自动填写为当前时间值。

3.9.1.3 "List.xhtml" 的修改

主要修改包含：

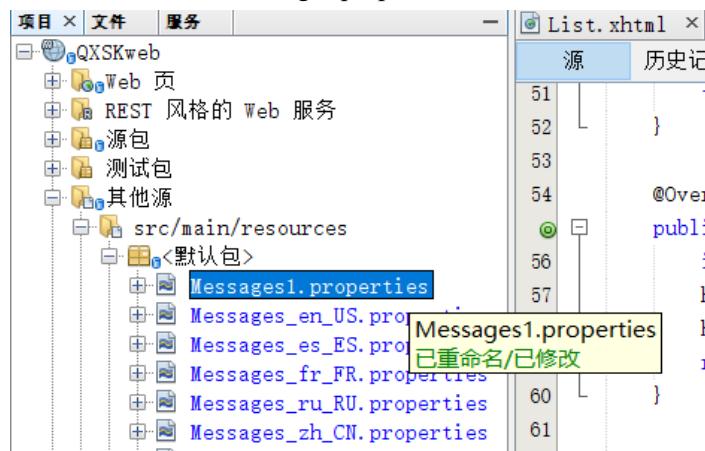
- 1) 数据行选择模式（selectionMode）被改为多选（multiple），相应的 ajax 事件被添加。
- 2) 分页器（Paginator）被增强了。
- 3) 数据表头加入了列选择（toggler）和小提示（tips）
- 4) 数据列可以被排序和过滤。
- 5) 时间字段被定义了模式（pattern）、时区（timeZone）和语言（locale）。
- 6) 可以批量删除。
- 7) 删除前有确认对话框。

3.9.1.4 "View.xhtml" 的修改

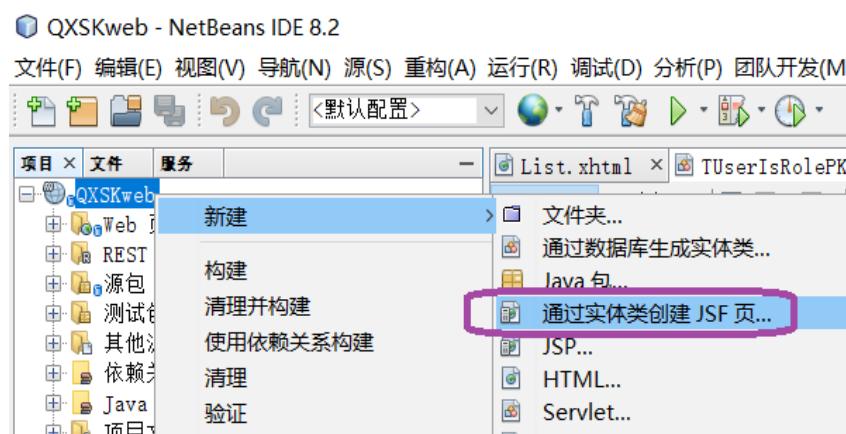
主要修改包含：时间字段被定义了模式（pattern）、时区（timeZone）和语言（locale）。

3.9.2 利用 Wizard 自动生成 MVC 各层文件

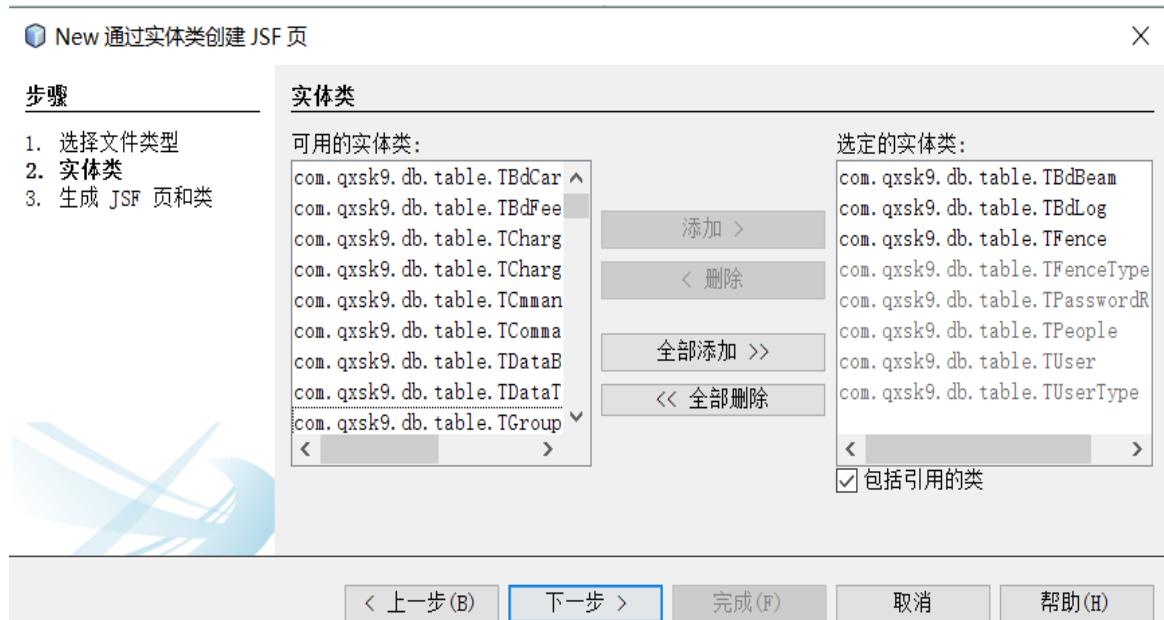
- 1) 把本地化文件“main/resources/Messages.properties”暂时改名，如改成“Messages1.properties”：



- 2) 右键点击项目名、选择“新建”-“通过实体类创建 JSF 页...”：



3) 选择实体类: (通常是全部选择)



4) 设置参数:

- (1) 所有自动生成的 session bean 都在包 com.qxsk9.db.ejb 中
- (2) 所有自动生成的 JSF controller 都在包 com.qxsk9.db.bean 中
- (3) 模板选择“PrimeFaces”(上一节替换的模板文件将发挥作用)
- (4) “JSF 页文件夹”可以设置为任意临时目录，因为以后将把生成的 JSF 文件归类到预定目录下。
- (5) “本地化包名称”设置为“/Messages”
- (6) 选择“覆盖现有文件”



5) 点击按钮“完成”，则系统开始自动生成 MVC 所有文件。

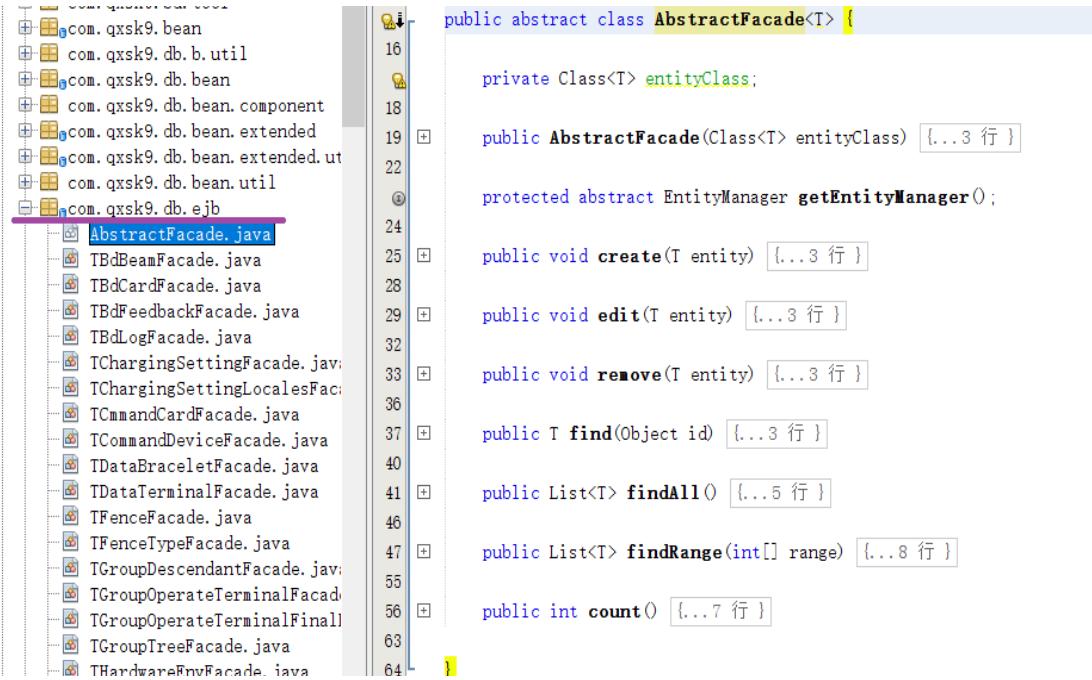
6) 把自动生成的本地化文件“main/resources/Messages.properties”删除，把原先的文件改回此名。

说明：通常系统自动生成的本地化文件并不符合实际需要。第一次自动生成的本地化文件可以作为基础、以编辑成自己定制的本地化文件，以后再自动生成文件时可以与自己定制的文件编辑合并起来。

7) 可以多次执行这个过程，以适应 Entity Bean 发生的变化。

3.9.3 扩展 Session Bean

每个 Entity Bean（包 com.qxsk9.db.table 中）都对应一个自动生成一个 Session Bean（包 com.qxsk9.db.ejb 中），它们都继承于 AbstractFacade，包含对数据表进行增删改查的方法：



```

public abstract class AbstractFacade<T> {
    private Class<T> entityClass;

    public AbstractFacade(Class<T> entityClass) { ... 3 行 }

    protected abstract EntityManager getEntityManager();

    public void create(T entity) { ... 3 行 }

    public void edit(T entity) { ... 3 行 }

    public void remove(T entity) { ... 3 行 }

    public T find(Object id) { ... 3 行 }

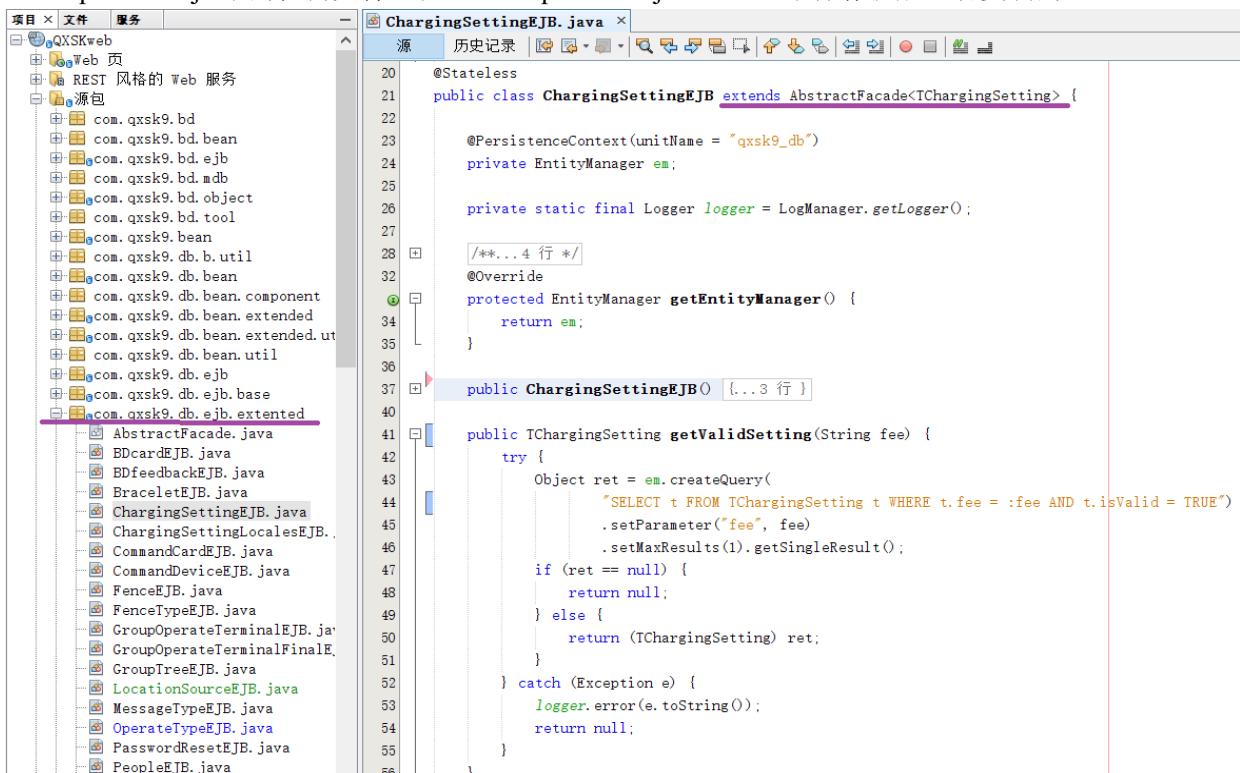
    public List<T> findAll() { ... 5 行 }

    public List<T> findRange(int[] range) { ... 8 行 }

    public int count() { ... 7 行 }
}

```

如果要对数据表定制自己的处理逻辑，则应该扩展或者复制而不是直接修改自动生成的 Session Bean，即不要修改包 com.qxsk9.db.ejb 中的任何文件。在包 com.qxsk9.db.ejb.extented 中保存扩展或者复制的 session bean：



```

@Stateless
public class ChargingSettingEJB extends AbstractFacade<TChargingSetting> {
    @PersistenceContext(unitName = "qxsk9_db")
    private EntityManager em;

    private static final Logger logger = LogManager.getLogger();

    /**
     * ...
     */
    @Override
    protected EntityManager getEntityManager() {
        return em;
    }

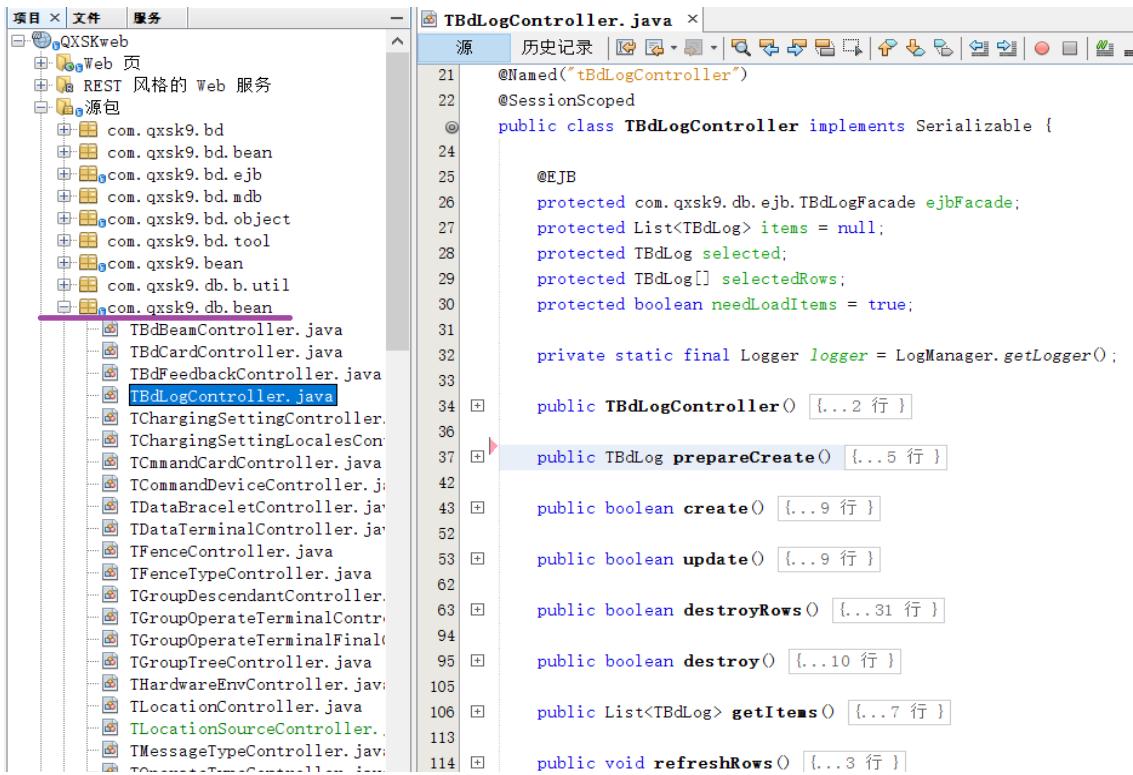
    public ChargingSettingEJB() { ... 3 行 }

    public TChargingSetting getValidSetting(String fee) {
        try {
            Object ret = em.createQuery(
                "SELECT t FROM TChargingSetting t WHERE t.fee = :fee AND t.isValid = TRUE"
            .setParameter("fee", fee)
            .setMaxResults(1).getSingleResult();
            if (ret == null) {
                return null;
            } else {
                return (TChargingSetting) ret;
            }
        } catch (Exception e) {
            logger.error(e.toString());
            return null;
        }
    }
}

```

3.9.4 扩展 JSF controller

每个自动生成的 session bean (包 com.qxsk9.db.ejb 中) 对应一个自动生成一个 JSF controller (包 com.qxsk9.db.bean 中), 包含 JSF 页面的处理逻辑和对 session bean 的调用。



```

    项目 × | 文件 | 服务 | 
    QXSKweb
      |- Web 页
      |- REST 风格的 Web 服务
      |- 源包
        |- com.qxsk9.bd
          |- com.qxsk9.bd.bean
          |- com.qxsk9.bd.ejb
          |- com.qxsk9.bd.mdb
          |- com.qxsk9.bd.object
          |- com.qxsk9.bd.tool
          |- com.qxsk9.bean
          |- com.qxsk9.db.b.util
          |- com.qxsk9.db.bean
            |- TBdBeamController.java
            |- TBdCardController.java
            |- TBdFeedbackController.java
            |- TBdLogController.java
            |- TChargingSettingController.java
            |- TChargingSettingLocalesController.java
            |- TCommandCardController.java
            |- TCommandDeviceController.java
            |- TDataBraceletController.java
            |- TDataTerminalController.java
            |- TFenceController.java
            |- TFenceTypeController.java
            |- TGGroupDescendantController.java
            |- TGGroupOperateTerminalController.java
            |- TGGroupOperateTerminalFinalController.java
            |- TGGroupTreeController.java
            |- THardwareEnvController.java
            |- TLocationController.java
            |- TLocationSourceController.java
            |- TMessageTypeController.java
            |- TUserManagementController.java

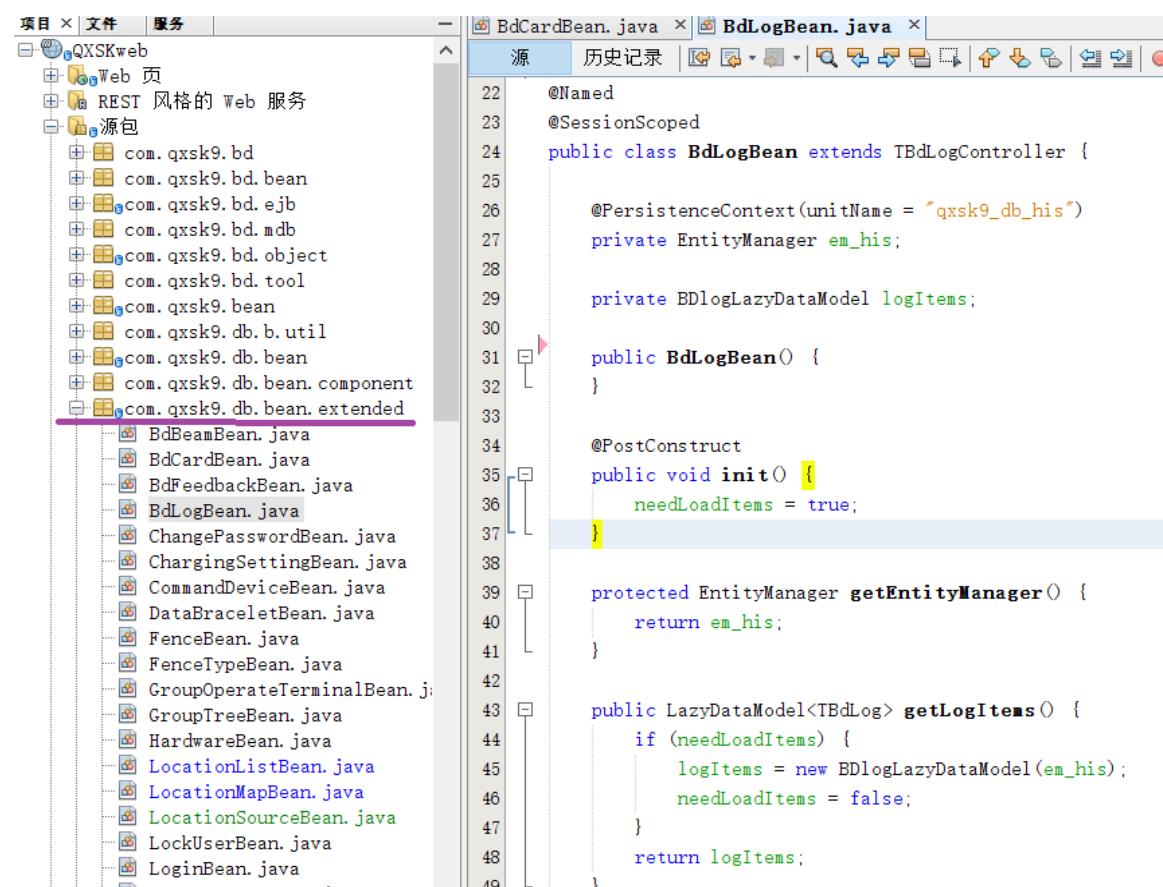
```

```

TBdLogController.java ×
源 历史记录 | 
21  @Named("tbDLogController")
22  @SessionScoped
23
24  public class TBdLogController implements Serializable {
25
26      @EJB
27      protected com.qxsk9.db.ejb.TBdLogFacade ejbFacade;
28      protected List<TBdLog> items = null;
29      protected TBdLog selected;
30      protected TBdLog[] selectedRows;
31      protected boolean needLoadItems = true;
32
33      private static final Logger logger = LogManager.getLogger();
34
35      public TBdLogController() { ... 2 行 }
36
37      public TBdLog prepareCreate() { ... 5 行 }
38
39      public boolean create() { ... 9 行 }
40
41      public boolean update() { ... 9 行 }
42
43      public boolean destroyRows() { ... 31 行 }
44
45      public boolean destroy() { ... 10 行 }
46
47      public List<TBdLog> getItems() { ... 7 行 }
48
49      public void refreshRows() { ... 3 行 }

```

如果要定制自己的 JSF 处理逻辑, 则应该扩展或者复制而不是直接修改自动生成的 JSF controller, 即不要修改包 com.qxsk9.db.bean 中的任何文件。在包 com.qxsk9.db.bean.extended 中保存扩展或者复制的 JSF controller:



```

    项目 × | 文件 | 服务 | 
    QXSKweb
      |- Web 页
      |- REST 风格的 Web 服务
      |- 源包
        |- com.qxsk9.bd
          |- com.qxsk9.bd.bean
          |- com.qxsk9.bd.ejb
          |- com.qxsk9.bd.mdb
          |- com.qxsk9.bd.object
          |- com.qxsk9.bd.tool
          |- com.qxsk9.bean
          |- com.qxsk9.db.b.util
          |- com.qxsk9.db.bean
          |- com.qxsk9.db.bean.component
          |- com.qxsk9.db.bean.extended
            |- BdBeamBean.java
            |- BdCardBean.java
            |- BdFeedbackBean.java
            |- BdLogBean.java
            |- ChangePasswordBean.java
            |- ChargingSettingBean.java
            |- CommandDeviceBean.java
            |- DataBraceletBean.java
            |- FenceBean.java
            |- FenceTypeBean.java
            |- GroupOperateTerminalBean.java
            |- GroupTreeBean.java
            |- HardwareBean.java
            |- LocationListBean.java
            |- LocationMapBean.java
            |- LocationSourceBean.java
            |- LockUserBean.java
            |- LoginBean.java

```

```

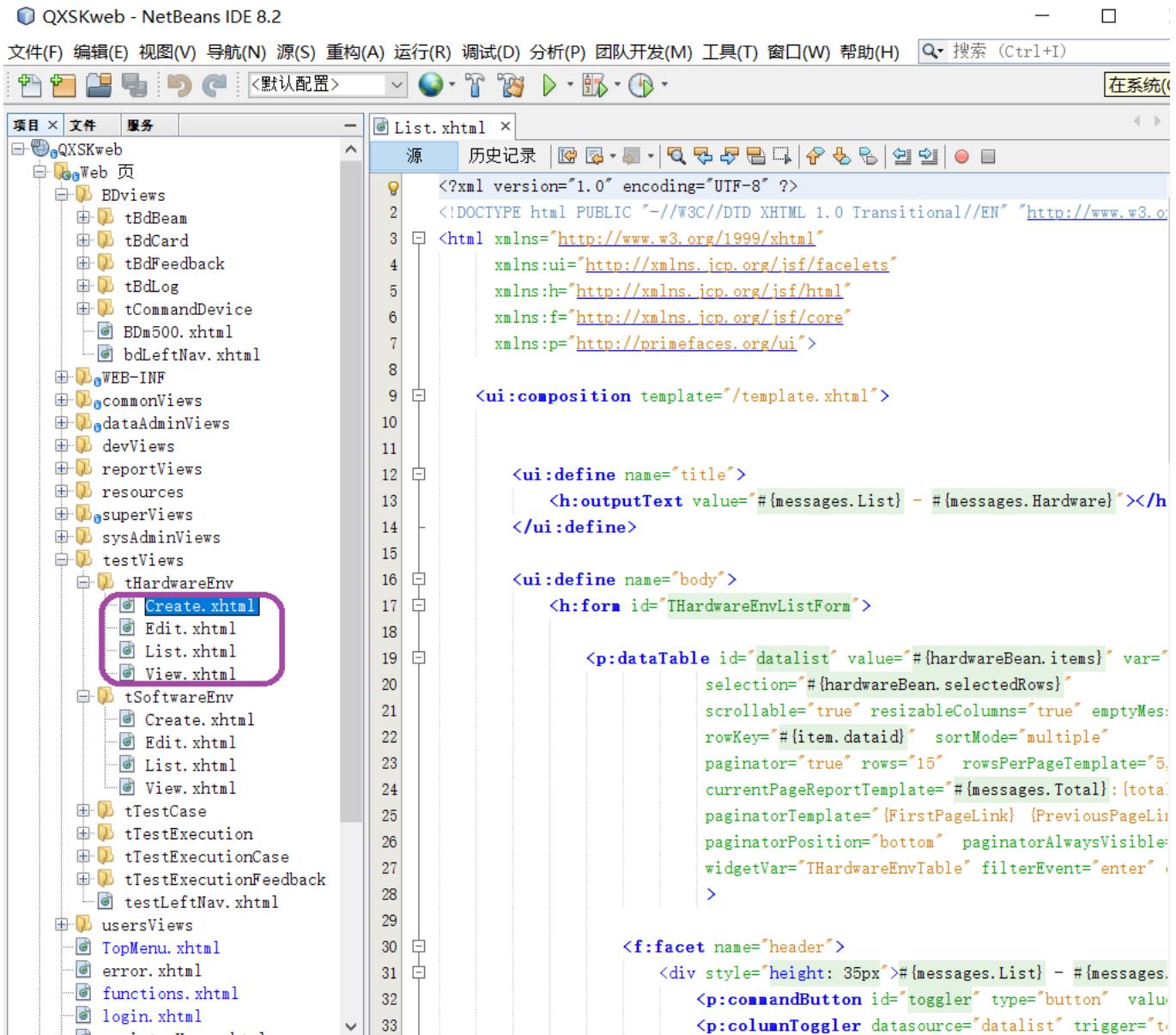
BdLogBean.java ×
源 历史记录 | 
22  @Named
23  @SessionScoped
24  public class BdLogBean extends TBdLogController {
25
26      @PersistenceContext(unitName = "qxsk9_db_his")
27      private EntityManager em_his;
28
29      private BDlogLazyDataModel logItems;
30
31      public BdLogBean() {
32      }
33
34      @PostConstruct
35      public void init() {
36          needLoadItems = true;
37      }
38
39      protected EntityManager getEntityManager() {
40          return em_his;
41      }
42
43      public LazyDataModel<TBdLog> getLogItems() {
44          if (needLoadItems) {
45              logItems = new BDlogLazyDataModel(em_his);
46              needLoadItems = false;
47          }
48          return logItems;
49      }

```

3.9.5 编辑 XHTML 文件

每个自动生成的 JSF controller 对应一个自动生成的目录、其下有 4 个自动生成的 XHTML 文件，分别是：List.xhtml、Create.xhtml、Edit.xhtml、View.xhtml，对应于数据表的增删改查。

可以把这些目录按要求（如权限）移动到不同目录下，然后编辑修改这些 XHTML 以满足系统需求：



The screenshot shows the NetBeans IDE interface with the following details:

- Project View:** Shows the project structure under "QXSKweb". A folder named "tHardwareEnv" contains four files: Create.xhtml, Edit.xhtml, List.xhtml, and View.xhtml. These four files are highlighted with a red rounded rectangle.
- Source Editor:** The current file is "List.xhtml". The code is as follows:

```

<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/1999/xhtml">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:ui="http://xmlns.jcp.org/jsf/facelets"
      xmlns:h="http://xmlns.jcp.org/jsf/html"
      xmlns:f="http://xmlns.jcp.org/jsf/core"
      xmlns:p="http://primefaces.org/ui">

<ui:composition template="/template.xhtml">

    <ui:define name="title">
        <h:outputText value="#{messages.List} - #{messages.Hardware}"></h:outputText>
    </ui:define>

    <ui:define name="body">
        <h:form id="THardwareEnvListForm">

            <p:dataTable id="datalist" value="#{hardwareBean.items}" var="item"
                         selection="#{hardwareBean.selectedRows}"
                         scrollable="true" resizableColumns="true" emptyMessage="No data available"
                         rowKey="#{item.dataid}" sortMode="multiple"
                         paginator="true" rows="15" rowsPerPageTemplate="5"
                         currentPageReportTemplate="#{messages.Total}: {total}"
                         paginatorTemplate="{FirstPageLink} {PreviousPageLink} {PageLinks} {NextPageLink} {LastPageLink}"
                         paginatorPosition="bottom" paginatorAlwaysVisible="true"
                         widgetVar="THardwareEnvTable" filterEvent="enter" >
                <f:facet name="header">
                    <div style="height: 35px">#{messages.List} - #{messages.Hardware}</div>
                    <p:commandButton id="toggler" type="button" value="Toggle" />
                    <p:columnToggler datasource="datalist" trigger="true" />
                </f:facet>
                <tr>
                    <td>#<span>{item.dataid}</span></td>
                    <td>#<span>{item.name}</span></td>
                    <td>#<span>{item.type}</span></td>
                    <td>#<span>{item.value}</span></td>
                    <td>#<span>{item.unit}</span></td>
                    <td>#<span>{item.createTime}</span></td>
                    <td>#<span>{item.updateTime}</span></td>
                    <td>#<span>{item.state}</span></td>
                    <td>#<span>{item.description}</span></td>
                </tr>
            </p:dataTable>
        </h:form>
    </ui:define>
</ui:composition>

```

3.9.6 Xhtml 的布局模板

系统中大多数 xhtml 都以 template.xhtml 为模板，它的布局如下：

```

10 <f:view locale="#{localeBean.language}"> ━━━━ 界面语言的选择
11   <h:head>
12     <title><ui:insert name="title">Default Title</ui:insert></title> ━━━━ 网页标题
13     <h:outputStylesheet library="css" name="jsfcrud.css"/>
14     <h:outputScript library="js" name="tools.js"/>
15     <h:outputScript library="js" name="jsfcrud.js"/>
16     <h:outputScript library="js" name="PrimeFaces_locales.js"/>
17   </h:head>
18   <h:body>
19     <style type="text/css">
20       .ui-growl { position: absolute; top: 50%; left: 40%; } ━━━━ 弹出信息的样式
21     </style>
22     <p:layout fullPage="true">
23       <p:layoutUnit position="north" size="60">
24         <ui:include src="TopMenu.xhtml"/> ━━━━ 主菜单在北面
25       </p:layoutUnit>
26
27       <p:layoutUnit position="west" size="140" header="#{messages.Choose}" collapsible="true" resizable="true">
28         <ui:insert name="leftnav"/> ━━━━ 子菜单在西面
29       </p:layoutUnit>
30
31       <p:layoutUnit position="center" collapsible="true" resizable="true">
32         <p:growl id="growl" life="6000" /> ━━━━ 弹出信息的定义
33         <ui:insert name="body"/> ━━━━ 网页主体在中间
34       </p:layoutUnit>
35
36       <p:layoutUnit position="south" size="28" collapsible="true" resizable="true">
37         <div align="center" style="font-size:10px">
38           #{messages.QXSKCopyright} #{messages.QXSKLicence}:
39           <a href="http://www.miibeian.gov.cn" target="_blank">苏ICP备16007070号-1</a>
40         </div>
41       </p:layoutUnit>
42     </p:layout>
43   </h:body>
44 </f:view>

```

以下是一个网页主体的示例：

```

9 <ui:composition template="/template.xhtml"> ━━━━ 应用模板文件
10   <ui:define name="title">
11     <h:outputText value="#{messages.List} - #{messages.UserFeedback}"></h:outputText> ━━━━ 替换网页标题
12   </ui:define>
13
14   <ui:define name="body">
15     <style ...14行 /> ━━━━ 具体定义网页主体
16     <h:form ...172行 />
17
18
19
202     <ui:include src="Create.xhtml"/> ━━━━ 插入子网页
203     <ui:include src="Edit.xhtml"/>
204     <ui:include src="View.xhtml"/>
205     <ui:include src="Export.xhtml"/>
206   </ui:define>
207
208   <ui:define name="leftnav"> ━━━━ 具体定义子菜单
209     <ui:include src="/commonViews/helpLeftNav.xhtml"/>
210   </ui:define>
211 </ui:composition>

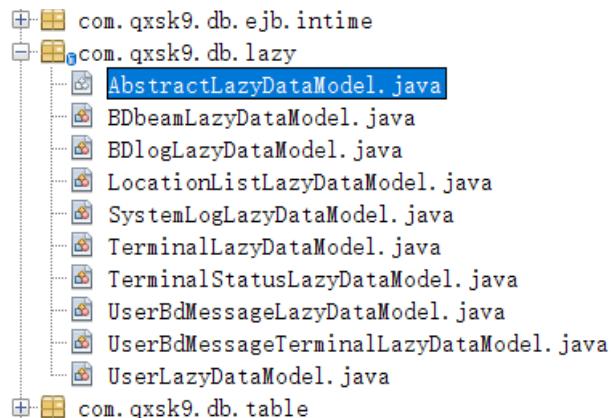
```

3.9.7 数据的 Lazy Loading

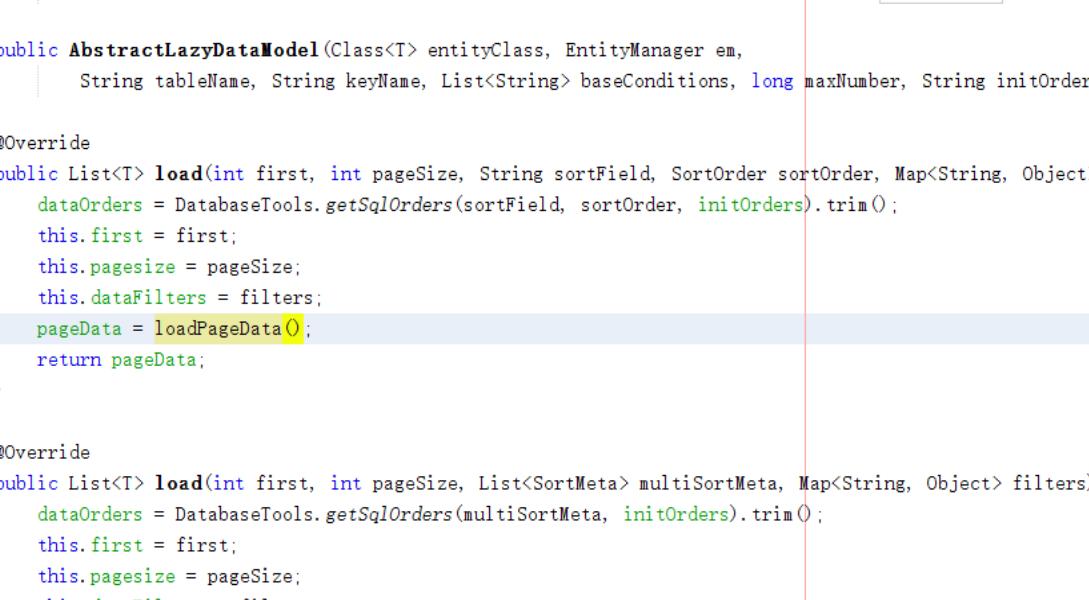
ajax 机制在分页时只渲染当前页的数据，做到了页面懒加载，但是仍然需要把整个数据集加载到内存里。而数据的懒加载则是对数据按需加载：只把页面需要数据读到内存里，同时实现了页面懒加载和数据懒加载。

为了实现数据懒加载，需要开发者具体编写 org.primefaces.model.LazyDataModel 的方法。本系统只对“大表”做了数据懒加载处理：

1) 所有懒加载类都在包 com.qxsk9.db.lazy 中:



2) 抽象类 AbstractLazyDataModel 具体实现两个 load 方法, 根据分页参数只把当前页面的数据读取出来:



```
45     public AbstractLazyDataModel(Class<T> entityClass, EntityManager em,
46             String tableName, String keyName, String baseCondition, String initOrders) { ... 22 行 }
47
48
49     public AbstractLazyDataModel(Class<T> entityClass, EntityManager em,
50             String tableName, String keyName, List<String> baseConditions, long maxNumber, String initOrders) { ... 20 行 }
51
52
53     @Override
54     public List<T> load(int first, int pageSize, String sortField, SortOrder sortOrder, Map<String, Object> filters) {
55         dataOrders = DatabaseTools.getSqlOrders(sortField, sortOrder, initOrders).trim();
56         this.first = first;
57         this.pageSize = pageSize;
58         this.dataFilters = filters;
59         pageData = loadPageData();
60         return pageData;
61     }
62
63
64     @Override
65     public List<T> load(int first, int pageSize, List<SortMeta> multiSortMeta, Map<String, Object> filters) {
66         dataOrders = DatabaseTools.getSqlOrders(multiSortMeta, initOrders).trim();
67         this.first = first;
68         this.pageSize = pageSize;
69         this.dataFilters = filters;
70         pageData = loadPageData();
71         return pageData;
72     }
73
74
75     private List<T> loadPageData() {
76         // Implementation of loadPageData method
77     }
78 }
```

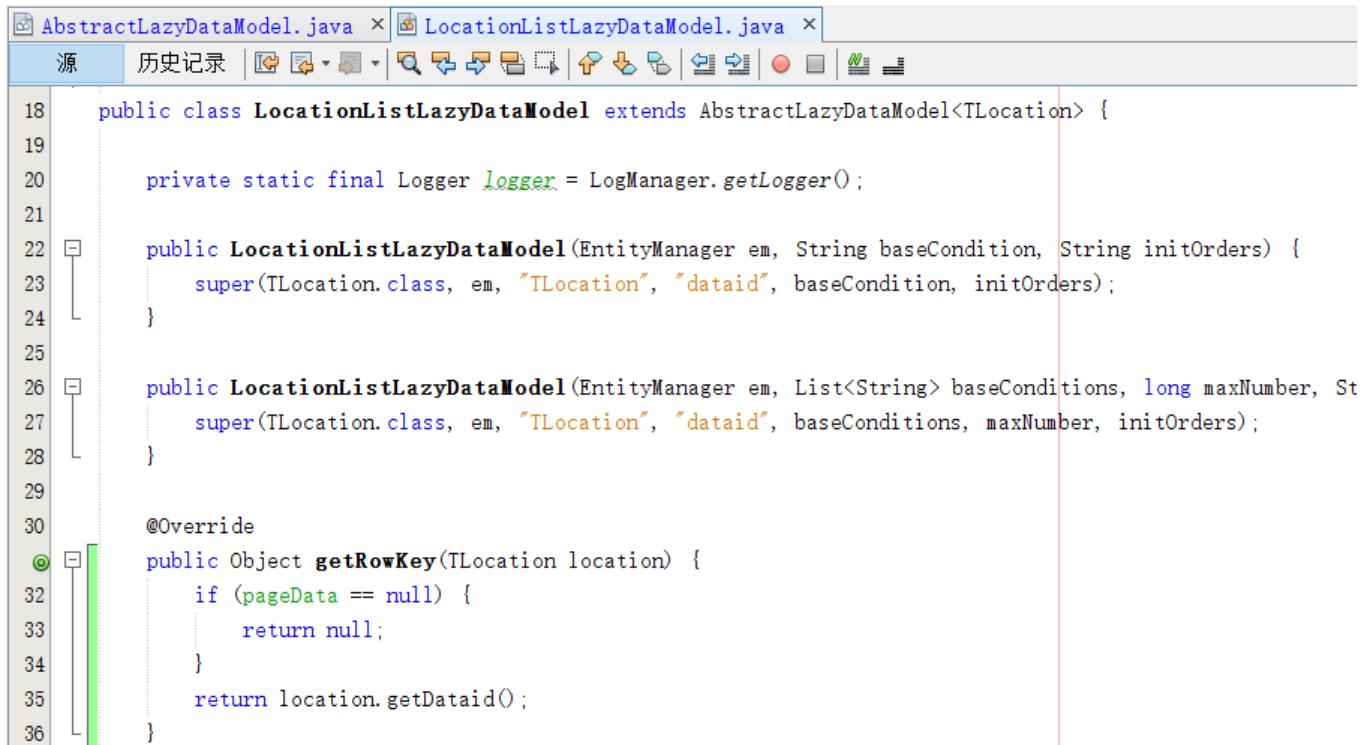


```

248     public List<T> loadPageData() {
249         try {
250             thePageDataSql = loadDataSql();
251             if (thePageDataSql == null) {
252                 this.setRowCount(0);
253                 pageData = null;
254                 return null;
255             }
256             pageData = (List<T>) em.createQuery(thePageDataSql)
257                 .setFirstResult(first)
258                 .setMaxResults(pagesize)
259                 .getResultList();
260
261             long count = (long) em.createQuery(loadPageSizeSql())
262                 .getSingleResult();
263             this.setRowCount((int) count);
264             return pageData;
265         } catch (Exception e) {
266             logger.error(e.toString());
267             pageData = null;
268             this.setRowCount(0);
269         }
270     }
271 }

```

3) 数据表的类继承 AbstractLazyDataModel，在构造方法中传入与数据表相关的参数：



```

18     public class LocationListLazyDataModel extends AbstractLazyDataModel<TLocation> {
19
20         private static final Logger logger = LogManager.getLogger();
21
22         public LocationListLazyDataModel(EntityManager em, String baseCondition, String initOrders) {
23             super(TLocation.class, em, "TLocation", "dataid", baseCondition, initOrders);
24         }
25
26         public LocationListLazyDataModel(EntityManager em, List<String> baseConditions, long maxNumber, St
27             super(TLocation.class, em, "TLocation", "dataid", baseConditions, maxNumber, initOrders);
28
29
30         @Override
31         public Object getRowKey(TLocation location) {
32             if (pageData == null) {
33                 return null;
34             }
35             return location.getDataid();
36         }

```

3.9.8 定制本地化文件

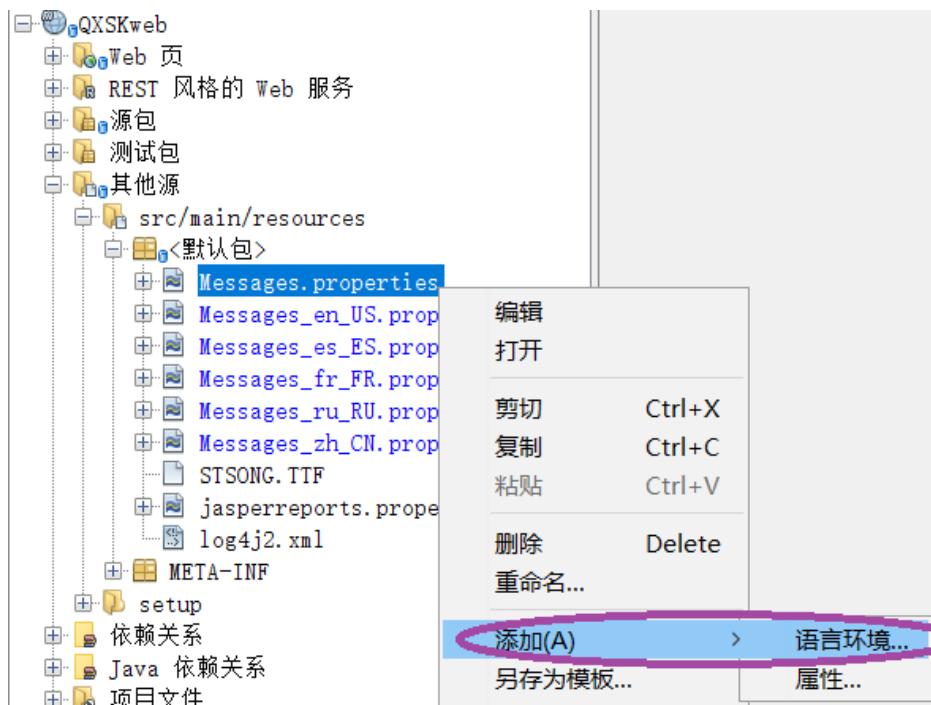
1) 编辑配置文件“WEB-INF\faces-config.xml”，设置本系统的本地化文件的基础名、和支持的语言列表：

```

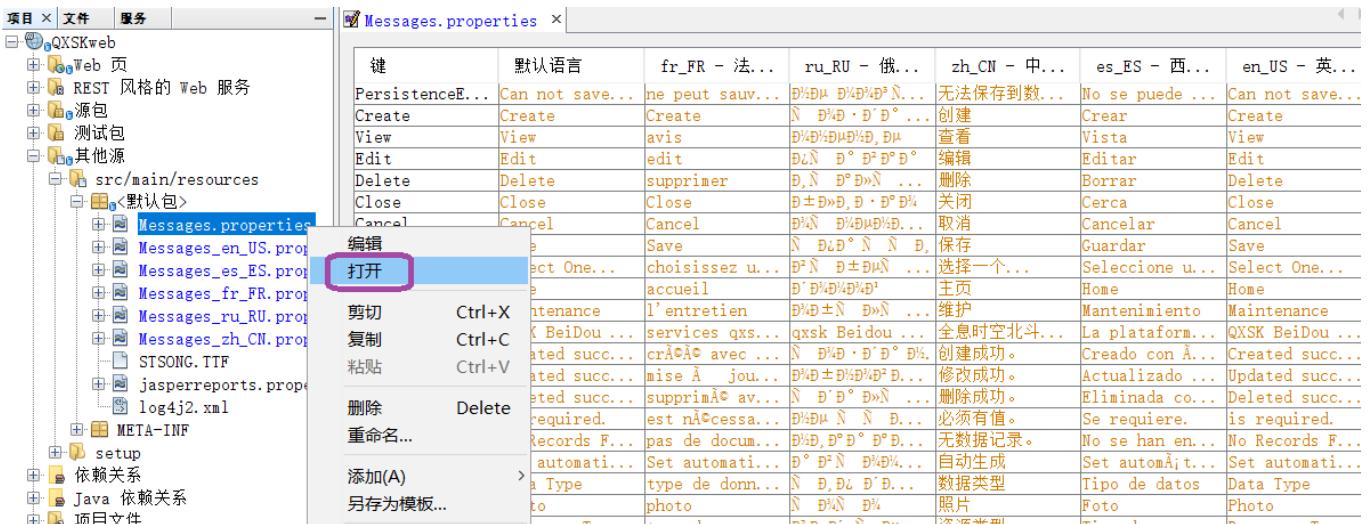
<?xml version='1.0' encoding='UTF-8'?>
<faces-config version="2.2"
  xmlns="http://xmlns.jcp.org/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
    faces-config_2_2.xsd">
  <application>
    <resource-bundle>
      <base-name>/Messages</base-name>
      <var>messages</var>
    </resource-bundle>
    <locale-config>
      <default-locale>en_US</default-locale>
      <supported-locale>zh_CN</supported-locale>
      <supported-locale>fr_FR</supported-locale>
      <supported-locale>es_ES</supported-locale>
      <supported-locale>ru_RU</supported-locale>
    </locale-config>
  </application>
  <managed-bean>
    <managed-bean-name>currentDate</managed-bean-name>
    <managed-bean-class>java.util.Date</managed-bean-class>
    <managed-bean-scope>request</managed-bean-scope>
  </managed-bean>
</faces-config>

```

2) 右键点击本地化文件“Messages.properties”、选择“添加”-“语言环境...”，以添加新的语言支持：



3) 右键点击本地化文件“Messages.properties”、选择“打开”，以同时编辑多种语言的翻译：



4) 页面语言由“com.qxsk9.bean.LocaleBean”设置：

```

public void setLanguage(String language) {
    try {
        logger.debug(language);
        switch (language.toLowerCase()) {
            case "zh":
            case "zh_cn":
                this.language = "zh_CN";
                break;
            case "en":
            case "en_us":
                this.language = "en_US";
                break;
            case "fr":
            case "fr_fr":
                this.language = "fr_FR";
                break;
            case "es":
            case "es_es":
                this.language = "es_ES";
                break;
            case "ru":
            case "ru_ru":
                this.language = "ru_RU";
                break;
            default:
                this.language = "zh_CN";
        }
        SessionTools.setSessionLanguage(this.language);
        locale = new Locale(this.language);
        FacesContext.getCurrentInstance().getViewRoot().setLocale(locale);
    } catch (Exception e) {
        this.language = "zh_CN";
    }
    loginBean.setLoginUserLanguage(this.language);
}

```

5) 界面语言的切换：模板文件 template.xhtml 中<f:view locale="#{localeBean.language}">。

3.9.9 定制数据导出器

PrimeFaces 支持页面导出数据：导出格式包括 xls、pdf、csv、xml；导出内容可以是：当前页、全表、选中的数据行。标签“p:dataExporter”应用起来很简洁实用，然而存在以下问题：

- 1) PDF 导出器由于硬代码而不支持中文数据：

<https://github.com/primefaces/primefaces/issues/3585>

- 2) XML 导出器由于硬代码不支持中文列名：

<https://github.com/primefaces/primefaces/issues/3588>

- 3) 没有 JSON 导出器。

查看 用户反馈

数据编号	357
用户反馈的对象	平台服务器
用户反馈类型	功能需求
标题	定制pdf和xml导出器
描述	primefaces的DataExporter对于pdf导出不支持中文（直接是西文字库的硬代码），对xml的导出不支持中文列名。
报告者	任珊虹
报告时间	2017-06-09 09:48:06
响应说明	定制的导出器可以灵活设置各种参数。
响应时间	2017-06-12 16:05:33
响应者	任珊虹
用户反馈状态	已解决
用户反馈的来源	网页客户端
解决的版本	-

所以我改写了 PrimeFaces 的导出器，使其支持中文的 PDF 导出、XML 导出、JSON 导出。所有扩展的导出器都在包 com.qxsk9.export 中。

以下是导出器的使用：XLS、CSV 的导出直接用的是 PrimeFaces 标签，而 PDF、XML、JSON 的导出用的是自己定制的 bean：

```

<p:commandLink value="pdf" ajax="false"
    onclick="PrimeFaces.monitorDownload(showStatus, hideStatus)">
    <p:fileDownload value="#{userFeedbackBean.exportFile('pdf')}" />
</p:commandLink>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
<p:commandLink value="xml" ajax="false"
    onclick="PrimeFaces.monitorDownload(showStatus, hideStatus)">
    <p:fileDownload value="#{userFeedbackBean.exportFile('xml')}" />
</p:commandLink>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
<p:commandLink value="json" ajax="false"
    onclick="PrimeFaces.monitorDownload(showStatus, hideStatus)">
    <p:fileDownload value="#{userFeedbackBean.exportFile('json')}" />
</p:commandLink>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
<p:commandLink value="xls" ajax="false"
    actionListener="#{userFeedbackBean.exporter.checkExportFilename2('TUserFeedback')}">
    <p:dataExporter type="xls" encoding="UTF-8"
        target="#:TUserFeedbackListForm:datable"
        pageOnly="#{userFeedbackBean.exporter.exportContent eq 'page'}"
        selectionOnly="#{userFeedbackBean.exporter.exportContent eq 'selected'}"
        fileName="#{userFeedbackBean.exporter.exportFilename}" />
</p:commandLink>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
<p:commandLink value="csv" ajax="false"
    actionListener="#{userFeedbackBean.exporter.checkExportFilename2('TUserFeedback')}">
    <p:dataExporter type="csv" encoding="UTF-8"
        target="#:TUserFeedbackListForm:datable"
        pageOnly="#{userFeedbackBean.exporter.exportContent eq 'page'}"
        selectionOnly="#{userFeedbackBean.exporter.exportContent eq 'selected'}"
        fileName="#{userFeedbackBean.exporter.exportFilename}" />
</p:commandLink>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

```

3.10 开发 RESTful Web Services

3.10.1 配置 Web Services 的环境

文件 com.qxsk9.webservice.resource.ApplicationConfig.java 定义了系统中所有网络服务的根地址。注意不要修改此类内的任何一行，系统会根据系统中网络服务的状态自动更新方法 addRestResourceClasses():

源 历史记录 |

```
16 @ApplicationPath("/ws") 系统所有网络服务的根地址
17 public class ApplicationConfig extends Application {
18
19     @Override
20     public Set<Class<?>> getClasses() {
21         Set<Class<?>> resources = new java.util.HashSet<>();
22         addRestResourceClasses(resources);
23         return resources;
24     }
25
26     /**
27      * Do not modify addRestResourceClasses() method. It is automatically
28      * populated with all resources defined in the project. If required, comment
29      * out calling this method in getClasses().
30     */
31     private void addRestResourceClasses(Set<Class<?>> resources) {
32         resources.add(com.qxsk9.webservice.resource.common.GeographyREST.class);
33         resources.add(com.qxsk9.webservice.resource.common.SystemREST.class);
34         resources.add(com.qxsk9.webservice.resource.terminal.DataReporter.class);
35         resources.add(com.qxsk9.webservice.resource.user.BraceletDataREST.class);
36     }
37 }
```

3.10.2 用 JAX-RS 注释定义 Web Services 的属性

以下是一个网络服务的示例：以 JAX-RS 注释来定义资源的调用方法、地址、响应的文件类型

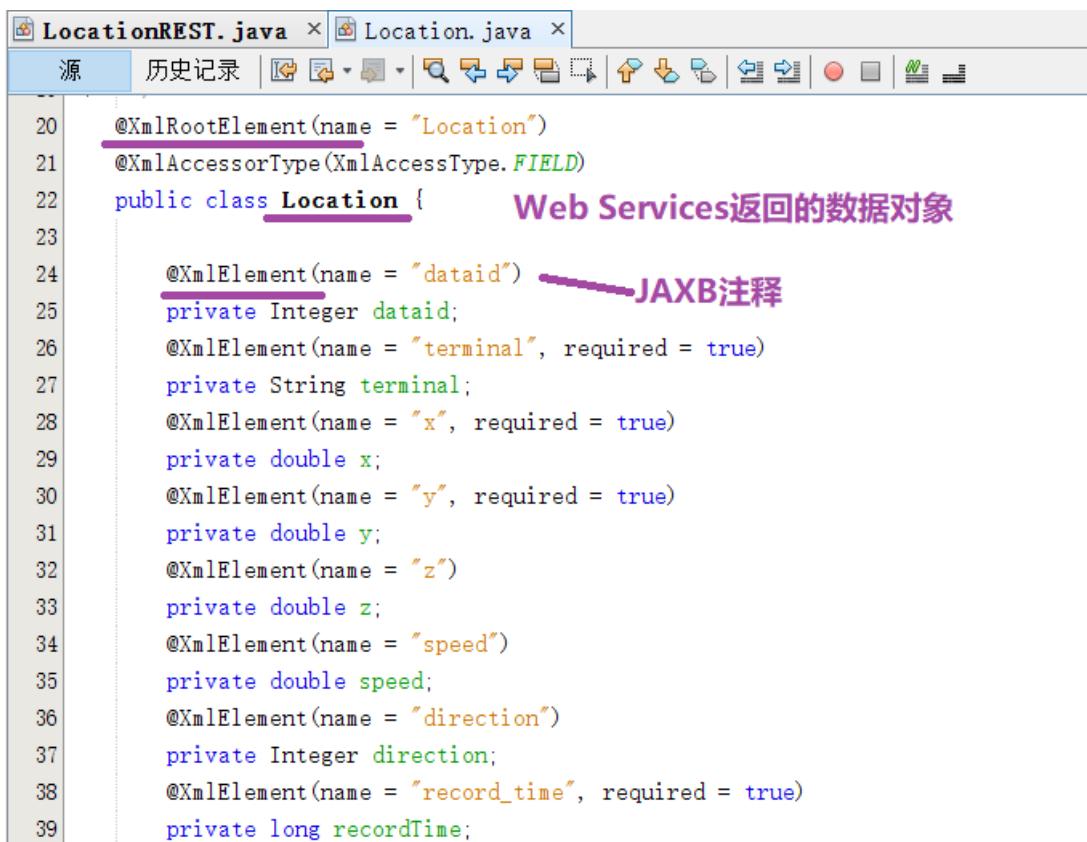
The screenshot shows a Java code editor with annotations highlighting specific parts of the code. Annotations include:

- 调用方法** (Call Method) points to the `@GET` annotation at line 61.
- 资源地址** (Resource Address) points to the `@Path("/gl")` annotation at line 62.
- 响应的数据类型** (Response Data Type) points to the `@Produces({MediaType.APPLICATION_JSON, MediaType.APPLICATION_XML})` annotation at line 63.
- 返回JAXB注释的对象** (Return JAXB annotated object) points to the `Location l = Location.buildLocation(theTerminal);` line at line 71.

```
61 @GET  
62 @Path("/gl")  
63 @Produces({MediaType.APPLICATION_JSON, MediaType.APPLICATION_XML})  
64 public Location getLast(@Context HttpServletRequest request,  
65     @QueryParam("t") String terminal) {  
66     if (terminal == null || terminal.isEmpty()) {  
67         return null;  
68     }  
69     String user = userValidator.getWsUser(request);  
70     TDataTerminal theTerminal = terminalEJB.getTerminal(user, terminal, "monitor");  
71     Location l = Location.buildLocation(theTerminal);  
72     return l;  
73 }
```

3.10.3 用 JAXB 注释定义 Web Services 的对象

以下是网络服务返回的数据对象：以 JAXB 注释定义属性



The screenshot shows a Java code editor with two tabs: "LocationREST.java" and "Location.java". The "Location.java" tab is active, displaying the following code:

```
20  @XmlRootElement(name = "Location")
21  @XmlAccessorType(XmlAccessType.FIELD)
22  public class Location {      Web Services返回的数据对象
23
24      @XmlElement(name = "dataid")      JAXB注释
25      private Integer dataid;
26      @XmlElement(name = "terminal", required = true)
27      private String terminal;
28      @XmlElement(name = "x", required = true)
29      private double x;
30      @XmlElement(name = "y", required = true)
31      private double y;
32      @XmlElement(name = "z")
33      private double z;
34      @XmlElement(name = "speed")
35      private double speed;
36      @XmlElement(name = "direction")
37      private Integer direction;
38      @XmlElement(name = "record_time", required = true)
39      private long recordTime;
```

A purple annotation "JAXB注释" points to the "@XmlElement" annotation at line 24.

3.11 开发 WebSocket 服务器端

3.11.1 基础类 DataSocket.java

本系统所有 WebSocket 类都继承于基础类 DataSocket.java，它实现以下通用功能：

- 1) 在 WebSocket 连接打开时检查账户合法性，若不合法则关闭连接。
- 2) 处理初始化参数：SessionLanguage（连接的语言）和 MonitoringTerminals（操作涉及的终端列表）。参见《全息时空运营平台-WebSocket 接口》。
- 3) 监视连接状态（心跳、异常、关闭）。

3.11.2 WebSocket 类

所有 WebSocket 类都在包 com.qxsk9.websocket 中，关注于以下功能：

- 1) 维护一个关于本地址的全局 session 列表，该表保存连接到本地址的所有客户端连接。
- 2) 定义向 session 发送数据的方法。此方法一般由 EJB 中写数据的方法调用，例如添加数据项时调用 WebSocket 的方法以向所有连接的 session 发送某类数据。

以下是一个 WebSocket 类的示例：

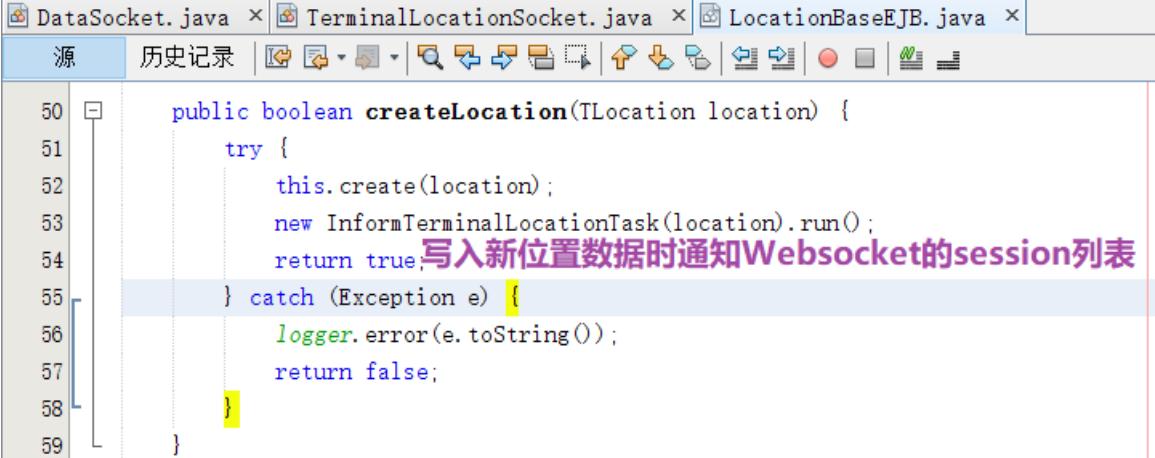
```

22  @ServerEndpoint("/terminalLocationSocket")
23  public class TerminalLocationSocket extends DataSocket {
24      public static Set<Session> TerminalLocationSocketSessions = null;
25
26      @Override
27      public void onOpenAction(Session session, EndpointConfig conf) {
28          synchronized (this) {
29              TerminalLocationSocketSessions = session.getOpenSessions();
30          }
31      }
32
33      @Override
34      public void onCloseAction(Session session, CloseReason closeReason) {
35          synchronized (this) {
36              TerminalLocationSocketSessions = session.getOpenSessions();
37          }
38      }
39
40      public static void informTerminalLocation(TLocation location) {
41          if (TerminalLocationSocketSessions == null || TerminalLocationSocketSessions.isEmpty() || location == null) {
42              return;
43          }
44          try {
45              String terminal = location.getTerminalid().getRegisterNumber();
46              for (Session session : TerminalLocationSocketSessions) {
47                  if (!session.isOpen()
48                      || !DataSocket.whetherInformSession(session, terminal)) {
49                      continue;
50                  }
51                  String jsonData = JsonTools.generateTerminalLocationJson(location);
52                  if (jsonData == null) {
53                      return;
54                  }
55                  session.getBasicRemote().sendObject(jsonData);
56              }
57          } catch (Exception e) {
58      }
59  }

```

3.11.3 向客户端发送数据

根据数据处理逻辑，系统在合适时机向连接到 WebSocket 的所有客户端发送数据。例如：在 EJB 里写入新位置数据时，在新线程中调用 WebSocket 类中的方法、向连接到 WebSocket 地址的所有客户端 session 发送数据：



The screenshot shows a Java code editor with three tabs: DataSocket.java, TerminalLocationSocket.java, and LocationBaseEJB.java. The LocationBaseEJB.java tab is active, displaying the following code:

```

50     public boolean createLocation(TLocation location) {
51         try {
52             this.create(location);
53             new InformTerminalLocationTask(location).run();
54             return true;
55         } catch (Exception e) {
56             logger.error(e.toString());
57             return false;
58         }
59     }

```

A purple annotation "写入新位置数据时通知Websocket的session列表" is overlaid on the code between lines 54 and 55.

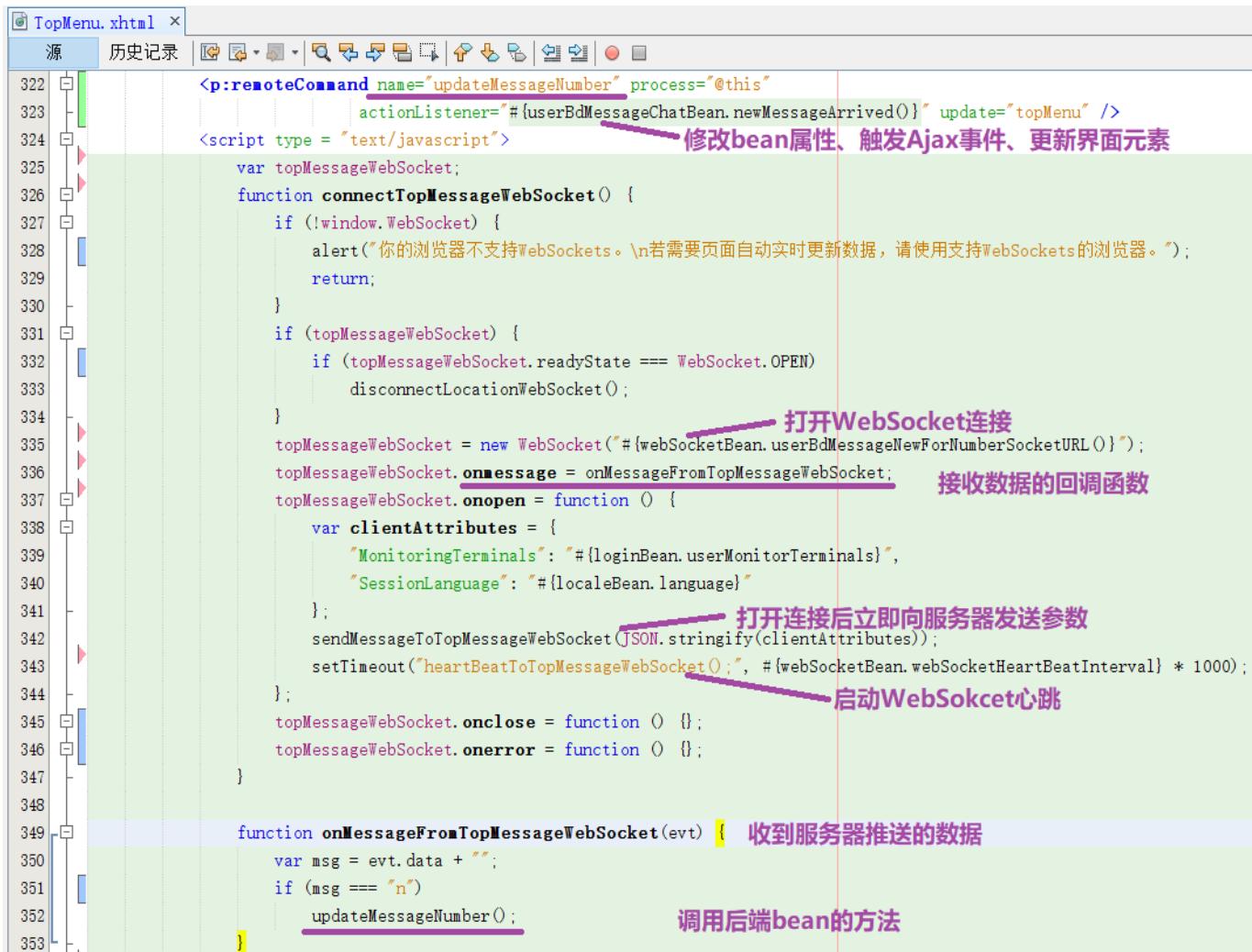


The screenshot shows the same Java code editor with the LocationBaseEJB.java tab active. The code is identical to the previous snippet, but a purple annotation "调用WebSocket类定义的方法" is overlaid on the line "TerminalLocationSocket.informTerminalLocation(location);".

3.12 开发 WebSocket 客户端-Javascript

以下是一个 WebSocket 客户端的 javascript 示例：在主菜单 TopMenu.xhtml 中

- 1) 利用对象 window.WebSocket 打开服务器提供的 WebSocket 地址。
- 2) 定义 onmessage 回调函数，以处理服务器发来的数据。
- 3) 定义 onopen 回调函数，以发送参数给服务器，并开启心跳。
- 4) 当接收到服务器发来的数据时，调用后端 bean 的方法以修改 bean 属性、触发 ajax 事件更新界面元素。



```

322 <p:remoteCommand name="updateMessageNumber" process="@this"
323   |   actionListener="#{userBdMessageChatBean.newMessageArrived()}" update="topMenu" />
324 <script type = "text/javascript">
325   |   var topMessageWebSocket;
326   |   function connectTopMessageWebSocket() {
327     |     if (!window.WebSocket) {
328       |       alert("你的浏览器不支持WebSockets。\\n若需要页面自动实时更新数据，请使用支持WebSockets的浏览器。");
329       |       return;
330     }
331     |     if (topMessageWebSocket) {
332       |       if (topMessageWebSocket.readyState === WebSocket.OPEN)
333         |         disconnectLocationWebSocket();
334     }
335     |     topMessageWebSocket = new WebSocket("#{webSocketBean.userBdMessageNewForNumberSocketURL()}");
336     |     topMessageWebSocket.onmessage = onMessageFromTopMessageWebSocket;
337     |     topMessageWebSocket.onopen = function () {
338       |       var clientAttributes = {
339         |         "MonitoringTerminals": "#{loginBean.userMonitorTerminals}",
340         |         "SessionLanguage": "#{localeBean.language}"
341       };
342       |       sendMessageToTopMessageWebSocket(JSON.stringify(clientAttributes));
343       |       setTimeout("heartBeatToTopMessageWebSocket()", #{webSocketBean.webSocketHeartBeatInterval} * 1000);
344     };
345     |     topMessageWebSocket.onclose = function () {};
346     |     topMessageWebSocket.onerror = function () {};
347   }
348
349   |   function onMessageFromTopMessageWebSocket(evt) { 收到服务器推送的数据
350     |     var msg = evt.data + "";
351     |     if (msg === "\n")
352       |       updateMessageNumber(); 调用后端bean的方法
353   }

```

The screenshot shows the Eclipse IDE interface with the TopMenu.xhtml file open. The code implements a WebSocket client. It uses a remoteCommand to trigger an update on the page. The script part handles the connection to the server, sending initial attributes, and setting up a heartbeat. It also defines an onmessage handler to receive messages from the server and update the UI by calling a backend bean method.

3.13 服务器的全局数据

3.13.1 系统常量

类 com.qxsk9.object.CommonValues 包含了所有的系统常量：

- 1) 在代码构建时就确定了值。
- 2) 不能被修改。

3.13.2 系统变量

类 com.qxsk9.system.SystemParameters 包含了所有的系统变量：

- 1) 在系统运行时动态修改。
- 2) 只能被代码修改。

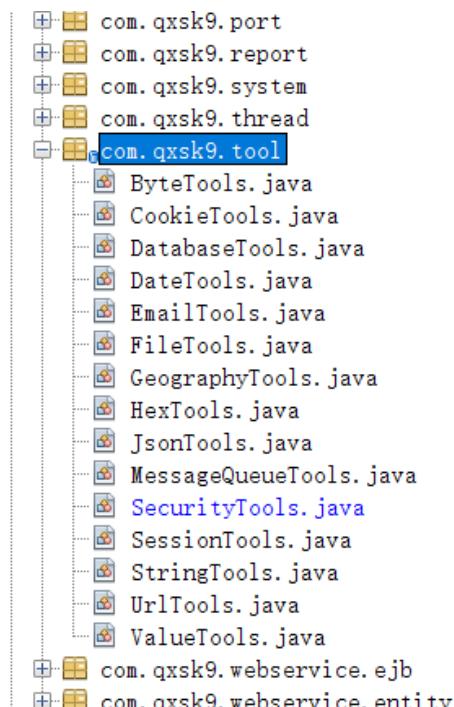
3.13.3 系统参数

类 com.qxsk9.system.SystemParameters 包含了所有的系统参数：

- 1) 对应于数据库表“t_system_parameter”的数据。
- 2) 由类 com.qxsk9.db.ejb.extended.SystemParameterEJB 在服务器初始化时读入内存。
- 3) 由系统管理员通过网页客户端进行增删改查。
- 4) 代码保证数据库表中的数据与内存中的数据始终保持一致。

3.13.4 系统工具箱

系统中通用的静态方法按用途定义在包 com.qxsk9.tool 的不同工具类中：



3.13.5 服务器的状态监听器

类 com.qxsk9.system.QXSKinit 继承了 ServletContextListener、成为 web 服务器的监听器：

```

41     @Override
42     public void contextInitialized(ServletContextEvent sce) {
43         logger.debug("初始化系统参数。。。");
44         systemParameterEJB.loadAllParameters();
45
46         logger.debug("初始化系统公共变量。。。");
47         SystemVariables.initData(sce);
48
49         logger.debug("初始化用户权限表。。。");
50         userOperateTerminalFinalEJB.loadData();
51
52         logger.debug("修正终端属性。。。");
53         terminalEJB.setLastMessages();
54
55         // 在子线程中启动各个线程以免阻塞系统启动
56         logger.debug("启动线程。。。");
57         new ThreadsTask().run();
58     }
59
60     @Override
61     public void contextDestroyed(ServletContextEvent sce) {
62         try {
63             if (GSM_port != null && GSM_port.isIsOpened()) {
64                 GSM_port.closePort();
65             }
66             if (BD_port != null && BD_port.isIsOpened()) {
67                 BD_port.closePort();
68             }
69
70             MQbroker.stop();
71             taskEJB.destroy();
72         } catch (Exception ex) {
73             logger.error(ex.toString());
74         }
75         System.out.println("系统停止。"); // logger在进入此方法之前似乎已被停止了
76     }

```

3.13.6 会话监听器

类 com.qxsk9.system.QXSKsessions 继承了 HttpSessionListener、成为服务器的会话监听器：在会话创建和销毁时给出处理。

```

@Override
public void sessionCreated(HttpSessionEvent event) {... 28 行}

@Override
public void sessionDestroyed(HttpSessionEvent event) {... 36 行}

```

3.14 服务器的初始化

3.14.1 服务器的启动步骤

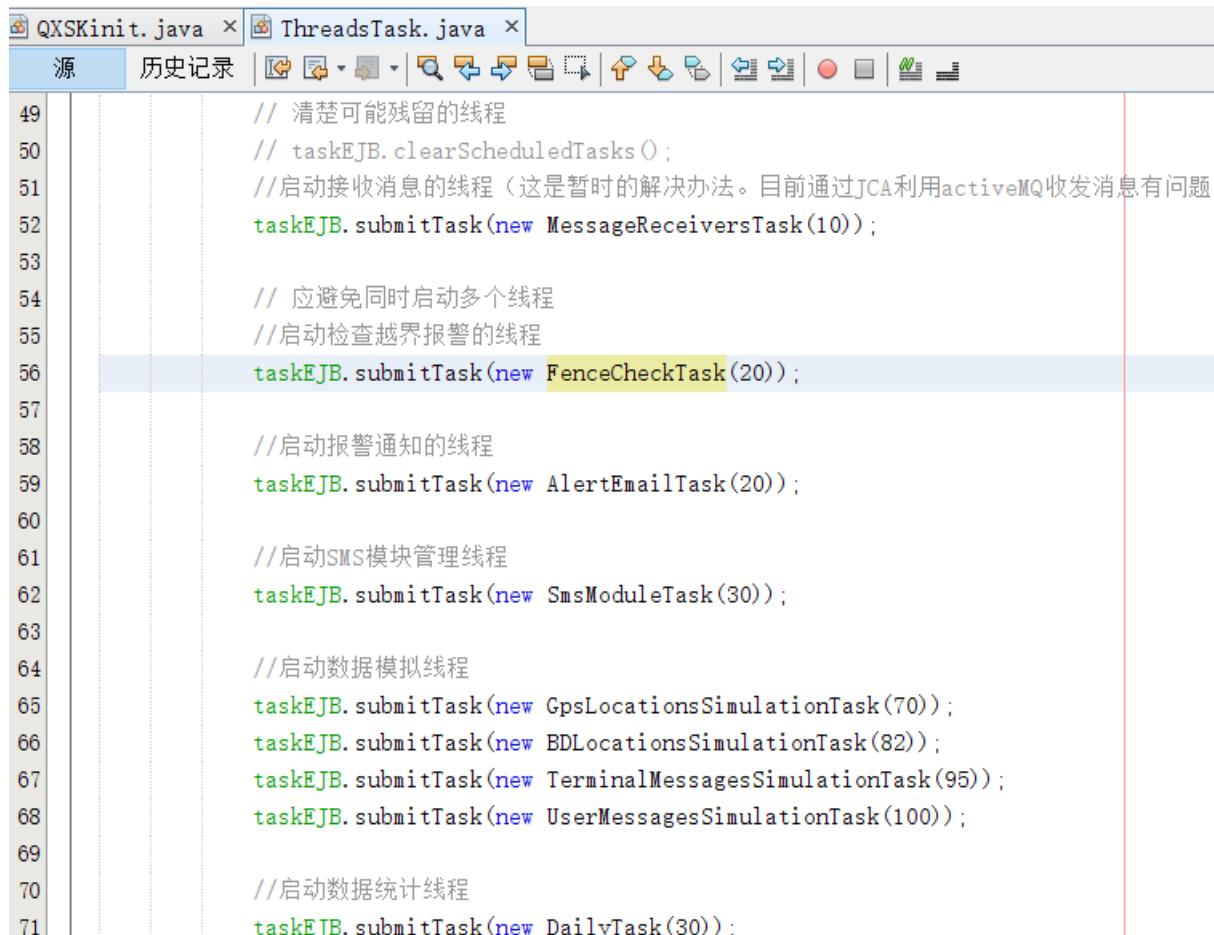
类 com.qxsk9.system.QXSKinit 在方法 contextInitialized 中定义了系统初始化的过程:

- 1) 从数据库中把系统参数读入内存。
- 2) 初始化系统变量:
 - (1) 实例化 GSM 端口和北斗端口
 - (2) 启动 ActiveMQ 代理
 - (3) 对与环境相关的全局变量赋值
- 3) 调用存储过程、产生初始的“最终用户数据权限表”。
- 4) 最后运行调度线程 ThreadsTask, 来启动耗时的初始化子任务, 以避免阻塞服务器的启动。

3.14.2 初始化任务的调度线程

调度线程 com.qxsk9.thread.ThreadTask:

- 1) 逐个启动初始化任务的子线程。
- 2) 每个初始化任务的子线程都被定义了延迟启动秒数, 例如 FenceCheckTask 将在 20 秒以后被启动。
- 3) 各个初始化任务的子线程的延迟启动秒数应当合理设计, 避免同时启动造成系统高负载。



```

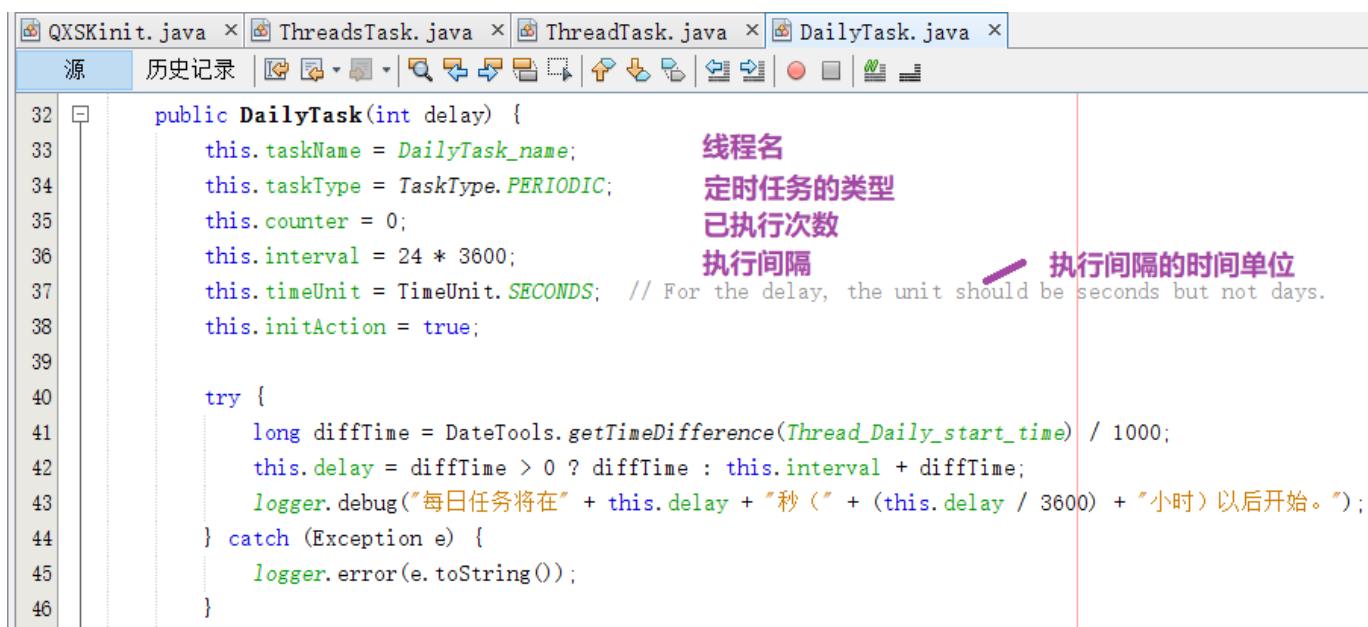
 49 // 清楚可能残留的线程
50 // taskEJB.clearScheduledTasks();
51 //启动接收消息的线程（这是暂时的解决办法。目前通过JCA利用activemq收发消息有问题
52 taskEJB.submitTask(new MessageReceiversTask(10));
53
54 // 应避免同时启动多个线程
55 //启动检查越界报警的线程
56 taskEJB.submitTask(new FenceCheckTask(20));
57
58 //启动报警通知的线程
59 taskEJB.submitTask(new AlertEmailTask(20));
60
61 //启动SMS模块管理线程
62 taskEJB.submitTask(new SmsModuleTask(30));
63
64 //启动数据模拟线程
65 taskEJB.submitTask(new GpsLocationsSimulationTask(70));
66 taskEJB.submitTask(new BDLocationsSimulationTask(82));
67 taskEJB.submitTask(new TerminalMessagesSimulationTask(95));
68 taskEJB.submitTask(new UserMessagesSimulationTask(100));
69
70 //启动数据统计线程
71 taskEJB.submitTask(new DailyTask(30));

```

3.14.3 初始化任务的子线程

初始化任务的子线程：

- 1) 在包 com.qxsk9.thread 中
- 2) 继承于 ThreadTask，包含以下属性：线程名、定时任务的类型、已执行次数、执行间隔、执行间隔的时间单位

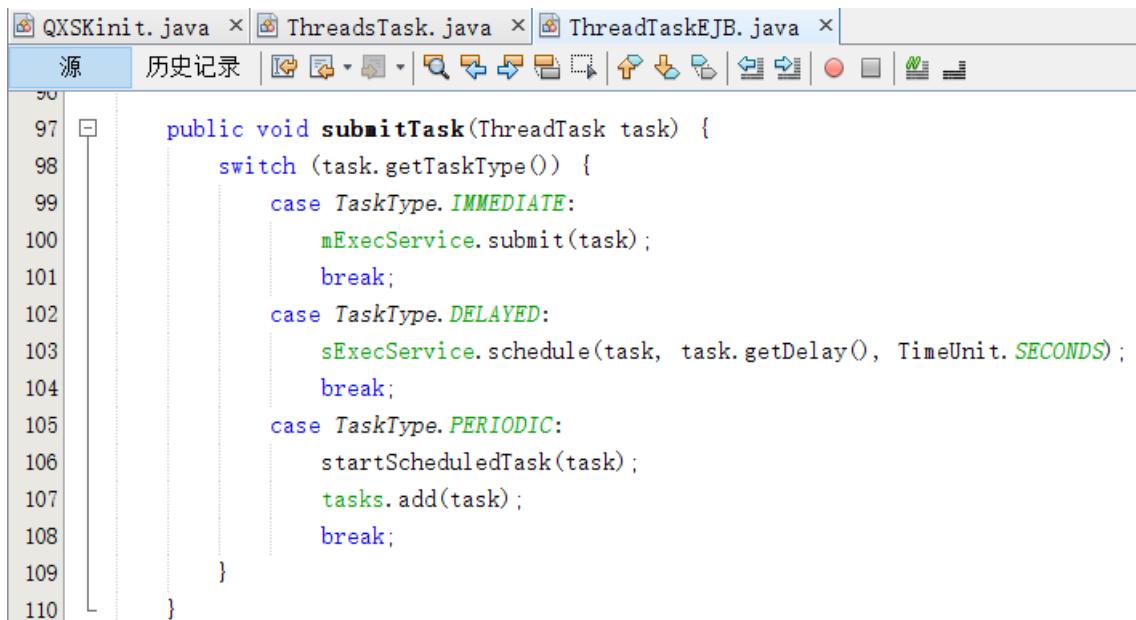


```

32  public DailyTask(int delay) {
33      this.taskName = DailyTask_name;
34      this.taskType = TaskType.PERIODIC;
35      this.counter = 0;
36      this.interval = 24 * 3600;
37      this.timeUnit = TimeUnit.SECONDS; // For the delay, the unit should be seconds but not days.
38      this.initAction = true;
39
40      try {
41          long diffTime = DateTools.getTimeDifference(Thread_Daily_start_time) / 1000;
42          this.delay = diffTime > 0 ? diffTime : this.interval + diffTime;
43          logger.debug("每日任务将在" + this.delay + "秒 (" + (this.delay / 3600) + "小时) 以后开始。");
44      } catch (Exception e) {
45          logger.error(e.toString());
46      }
}

```

- 3) 都是定时线程（TimerTask），分为三类：立即执行、延迟执行、周期执行：



```

97  public void submitTask(ThreadTask task) {
98      switch (task.getTaskType()) {
99          case TaskType.IMMEDIATE:
100              mExecService.submit(task);
101              break;
102          case TaskType.DELAYED:
103              sExecService.schedule(task, task.getDelay(), TimeUnit.SECONDS);
104              break;
105          case TaskType.PERIODIC:
106              startScheduledTask(task);
107              tasks.add(task);
108              break;
109      }
110  }

```

- 4) 立即执行和延迟执行的初始化任务只执行一次即结束，而周期执行的子线程成为系统的常驻子线程。

3.14.4 周期执行的子线程

1) 系统包括以下周期执行的子线程:

类名	作用	间隔值的系统参数	间隔的缺省值
FenceCheckTask	检查越界报警	Thread_Fence_checking_interval	3600 秒
AlertEmailTask	检查未发送的报警邮件	Thread_Alert_checking_interval	3600 秒
SmsModuleTask	删除 SIM 卡中所有短信	Thread_Sms_Module_interval	3600 秒
AlertMessageTask	检查未发送的报警短信	Thread_Alert_checking_interval	3600 秒
GpsLocationsSimulationTask	模拟新的 GPS 位置数据		10 秒
BdLocationsSimulationTask	模拟新的北斗位置数据		60 秒
TerminalMessagesSimulationTask	模拟终端发给用户的消息		20 秒
UserMessagesSimulationTask	模拟用户发给消息的消息		20 秒
DailyTask	刷新统计报告; 清除特定目录		1 天

2) 注意: 子线程 SmsModuleTask 不是系统初始化启动的, 而是在短信模块端口打开时启动的。

3) 系统管理员可以查看和启停系统以上周期执行的子线程。

3.14.5 打开端口

目前的策略是: 系统启动时不自动打开端口, 而是需要系统管理员登录系统手工打开北斗端口和短信端口。

查看 用户反馈	
数据编号	53
用户反馈的对象	平台服务器
用户反馈类型	问题报告
标题	端口有问题时系统奔溃无法正确发布应用
描述	目前, 在发布应用的同时就自动启动端口初始化工作, 这样若是端口有问题时系统直接退出。可以改为发布以后再由管理员初始化端口, 保障发布的顺利进行。
报告者	任珊虹
报告时间	2016-02-25 00:15:09
响应说明	现在在系统发布时不再打开端口, 而是由管理员手动启动端口。端口崩溃的问题是由于RXTXcomm驱动版本问题, 也已解决。
响应时间	2016-02-27 19:53:38
响应者	任珊虹
用户反馈状态	已解决
用户反馈的来源	网页客户端
解决的版本	-

3.15 服务器的安全

3.15.1 加密算法

类 com.qxsk9.tool.SecurityTools 以静态方法提供加密/解密功能：

- 1) 服务器对于网页登录、WebService 登录、WebSocket 登录分别提供不同的加密种子，以达到安全隔离的目的。
 - 2) 系统既支持 AES 加密算法也支持 PBE 加密算法，由客户端决定使用何种加密算法。
 - 3) PBE 加密的参数（盐的长度、迭代次数）也在此类中定义，开发者可以根据实际项目的特性修改参数。

```
public static byte[] getSalt() { ...10 行 }

public static SecretKey getSaltKey(String key) { ...12 行 }

public static String encryptPBE(byte[] salt, String key, String clearText) { ...12 行 }

public static String encryptAndroidPBE(String clearText) { ...3 行 }

public static String decryptPBE(byte[] salt, String key, String ciperText) { ...12 行 }

public static String decryptPBE(String key, String ciperText) { ...10 行 }

public static String decryptAndroidPBE(String clearText) { ...3 行 }

public static String serverEncryptAES(String content) { ...3 行 }

public static String websocketEncryptAES(String content) { ...3 行 }

public static String encryptAES(String content, String seed) { ...24 行 }

public static String serverDecryptAES(String content) { ...3 行 }

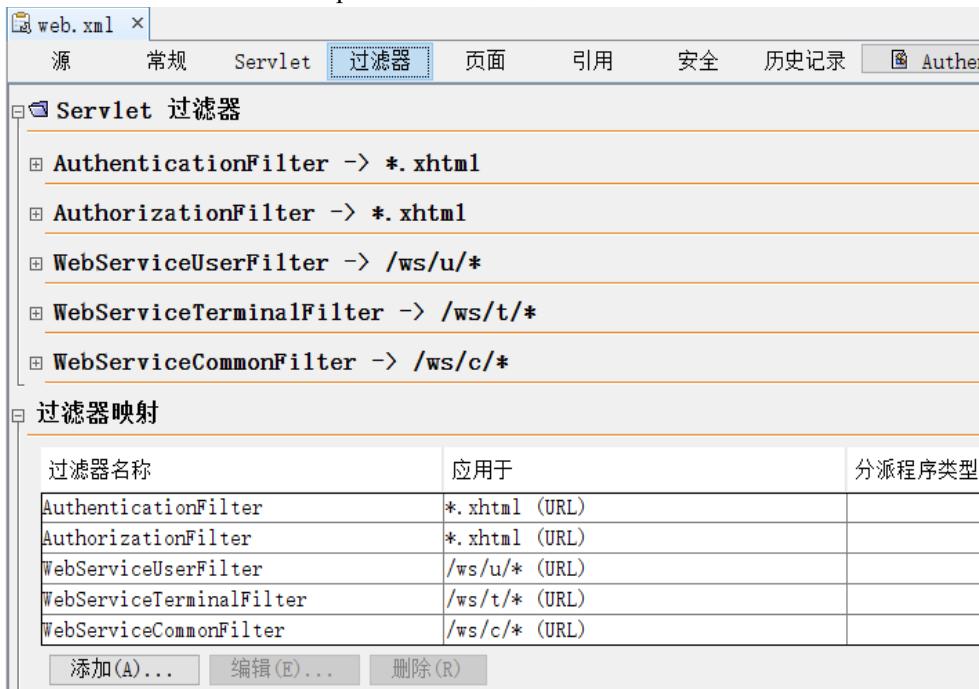
public static String websocketDecryptAES(String content) { ...3 行 }

public static String decryptAES(String content, String seed) { ...22 行 }
```

查看 用户反馈	
数据编号	439
用户反馈的对象	安卓客户端
用户反馈类型	功能需求
标题	对称的数据加密算法
描述	新版Android不再支持基于Crypto provider的加密算法。我们需要对称加密算法，用来保存敏感数据，而不是用来验证密码。因此需要找到新版android支持的双向加密算法。
报告者	任珊虹
报告时间	2017-12-08 10:44:05
响应说明	利用PBE实现对称加密。
响应时间	2018-02-28 19:56:13
响应者	任珊虹
用户反馈状态	已解决
用户反馈的来源	网页客户端
解决的版本	安卓客户端-2.100

3.15.2 资源的过滤器

系统中所有的资源过滤器都在包 com.qxsk9.filters 中，并且在 WEB-INF\web.xml 给出约束范围：



3.15.3 XHTML 的认证过滤器

类 com.qxsk9.filters.AuthenticationFilter 作用于所有 XHTML 网页：

- 1) 若网页地址在 passURLs 内，则不做检查、直接通过
- 2) 对于不在 passURLs 内的网页地址，若当前 session 已登录，则通过检查。
- 3) 若网页地址不在 passURLs 内，还未登录或者会话已超时，则网页重定向到登录界面。
- 4) 若网页请求是 ajax 请求，则重定向的代码需要特殊处理。

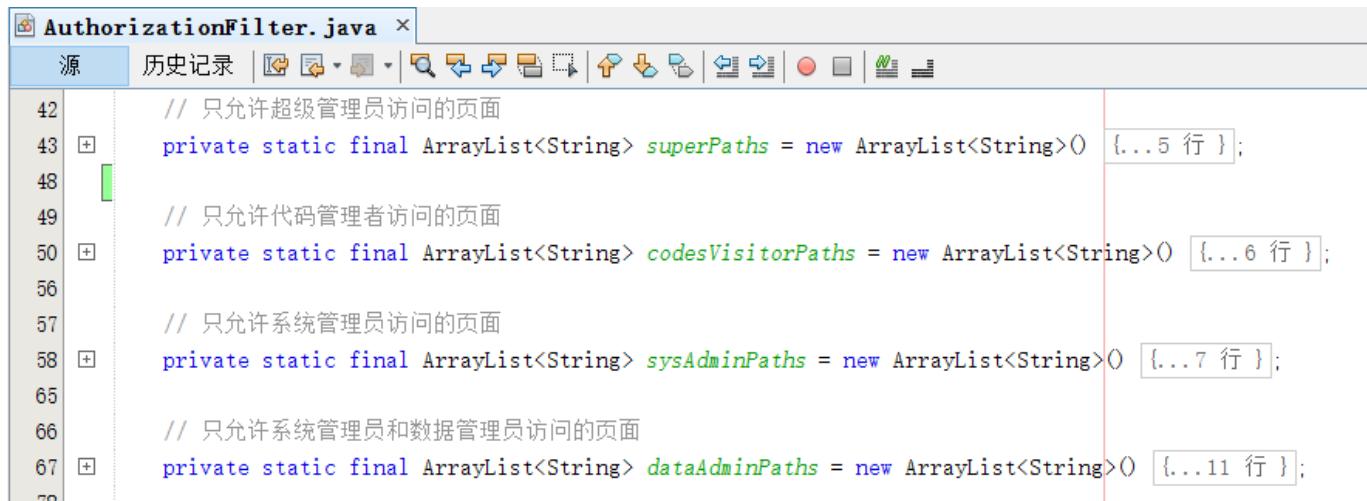
```

70     if (isPassURL(request) || currentUser != null) {
71         chain.doFilter(request, response);
72     } else { // 未登陆或会话超时
73         String redirectURL = httpRequest.getContextPath() + loginURL;
74         // 区分普通http请求还是ajax请求
75         if (httpRequest.getHeader("x-requested-with") != null &&
76             httpRequest.getHeader("x-requested-with").equals("XMLHttpRequest")) {
77             StringBuilder sb = new StringBuilder();
78             sb.append("<?xml version='1.0\' encoding='UTF-8\'?><partial-response><redirect url='");
79             append(redirectURL).append("\'></redirect></partial-response>");
80             httpResponse.setCharacterEncoding("UTF-8");
81             httpResponse.setContentType("text/xml");
82             PrintWriter pw = response.getWriter();
83             pw.println(sb.toString());
84             pw.flush();
85         } else {
86             httpResponse.sendRedirect(redirectURL);
87         }
88     }

```

3.15.4 xhtml 的授权过滤器

类 com.qxsk9.filters.AuthorizationFilter 作用于所有 xhtml 网页：根据用户的角色检查其可访问网页的范围



```

AuthorizationFilter.java
源 历史记录 | 搜索 | 打开 | 新建 | 保存 | 取消 | 重置 | 全屏 | 关闭
42 // 只允许超级管理员访问的页面
43 private static final ArrayList<String> superPaths = new ArrayList<String>() {... 5 行 };
48
49 // 只允许代码管理者访问的页面
50 private static final ArrayList<String> codesVisitorPaths = new ArrayList<String>() {... 6 行 };
56
57 // 只允许系统管理员访问的页面
58 private static final ArrayList<String> sysAdminPaths = new ArrayList<String>() {... 7 行 };
65
66 // 只允许系统管理员和数据管理员访问的页面
67 private static final ArrayList<String> dataAdminPaths = new ArrayList<String>() {... 11 行 };
70
    
```

3.15.5 公共 WebService 的认证过滤器

类 com.qxsk9.filters.WebServiceCommonFilter:

- 1) 作用于所有公共 WebService，即地址如 ws/c/* 的网络服务：
- 2) 这个过滤器目前不做任何检查，直接通过。

3.15.6 面向用户的 WebService 的认证过滤器

类 com.qxsk9.filters.WebServiceUserFilter:

- 1) 作用于所有面向用户的 WebService，即地址如 ws/u/* 的网络服务：
- 2) 地址中必须包含合法用户账号的参数
- 3) 既可以用明文密码，即“u=用户名&p=密码”，也可以用 PBE 密码，即“u=用户名&p=PBE 密码”
- 4) 当认证不通过时，网络服务返回错误码 403（禁止访问）面向用户的 WebService 的认证过滤器

注意：注册用户的 WebService 地址为 ws/ru，不受这个过滤器影响。

3.15.7 面向终端的 WebService 的认证过滤器

类 com.qxsk9.filters.WebServiceTerminalFilter:

- 1) 作用于所有面向终端的 WebService，即地址如 ws/t/* 的网络服务：
- 2) 地址中必须包含合法终端报告码的参数，即“t=终端名&k=报告码”
- 3) 当认证不通过时，网络服务返回错误码 403（禁止访问）

3.15.8 安全链接 SSL

参见《运行环境的安装与配置手册》。

3.16 读写分离

读写分离（Read/Write Splitting）：让主数据库（master）处理事务性增、改、删操作，而从数据库（slave）处理查询操作。

3.16.1 方案的选择

实现读写分离可以采用 Mysql Proxy：

<http://jayluns.iteye.com/blog/2275690>

使用代理的好处是对代码透明：不必修改代码、由代理判断数据读写应该在哪一个服务器上实现，缺点是代理可能又成了性能瓶颈。

本系统采用另一个方案：数据持久层实现读写分离。理由是：少许配置和代码改动即可在代码层面实现读写分离，无需代理：涉及读写的操作，连接实时库；单纯读的操作，连接历史库

3.16.2 定义持久层单元

编辑配置文件“resources\META-INF\persistence.xml”，为主从数据库分别定义持久层单元：

```

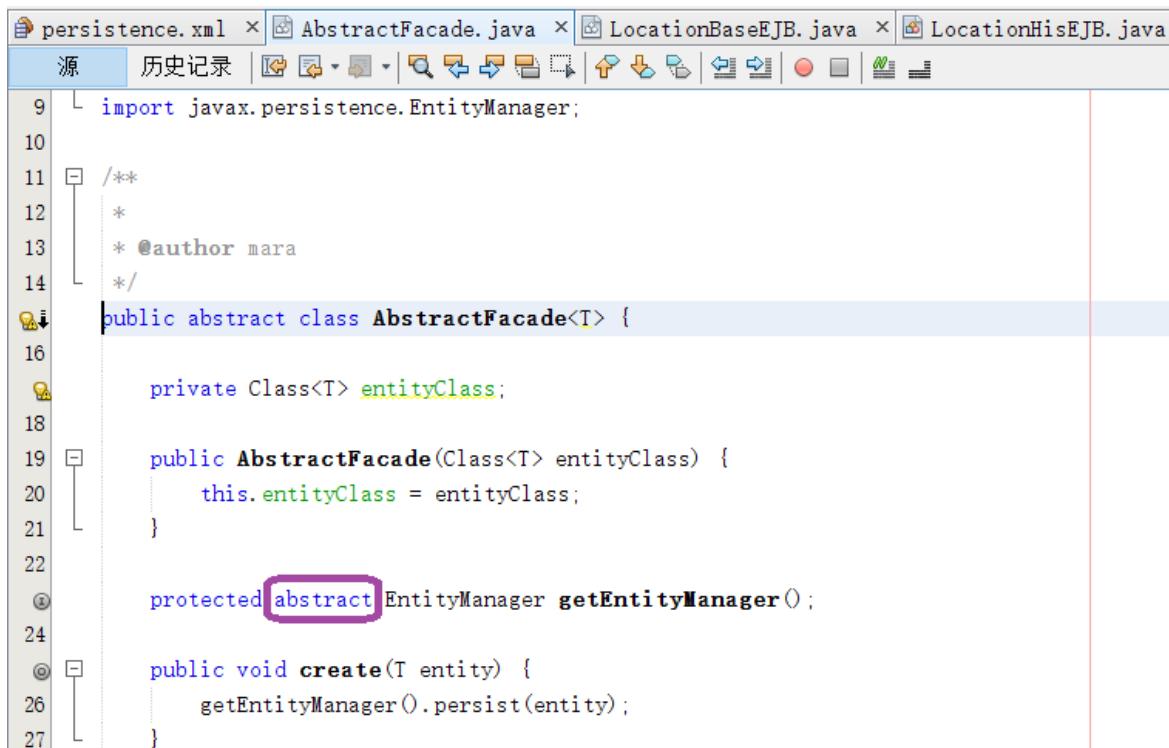
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.1" xmlns="http://xmlns.jcp.org/xml/ns/persistence" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd">
    <persistence-unit name="qxsk9_db" transaction-type="JTA">
        <provider>org.eclipse.persistence.jpa.PersistenceProvider</provider>
        <jta-data-source>jdbc/beidou</jta-data-source>
        <exclude-unlisted-classes>false</exclude-unlisted-classes>
        <!-- <properties>
            <property name="eclipselink.logging.level" value="FINE"/>
            <property name="eclipselink.logging.logger" value="DefaultLogger"/>
        </properties>-->
    </persistence-unit>
    <persistence-unit name="qxsk9_db_repl" transaction-type="JTA">
        <provider>org.eclipse.persistence.jpa.PersistenceProvider</provider>
        <jta-data-source>jdbc/beidouRepl</jta-data-source>
        <exclude-unlisted-classes>false</exclude-unlisted-classes>
        <!-- <properties>
            <property name="eclipselink.logging.level" value="FINE"/>
            <property name="eclipselink.logging.logger" value="DefaultLogger"/>
        </properties>-->
    </persistence-unit>
    <persistence-unit name="qxsk9_db_his" transaction-type="JTA">
        <provider>org.eclipse.persistence.jpa.PersistenceProvider</provider>
        <jta-data-source>jdbc/beidouHis</jta-data-source>
        <exclude-unlisted-classes>false</exclude-unlisted-classes>
        <properties>
            <!-- <property name="eclipselink.logging.level" value="FINE"/>
            <property name="eclipselink.logging.logger" value="DefaultLogger"/>-->
            <!--<property name="eclipselink.cache.shared.default" value="false"/>-->
        </properties>
        <!--<shared-cache-mode>NONE</shared-cache-mode>-->
    </persistence-unit>
</persistence>

```

3.16.3 定义“基础 EJB”

改造需要读写分离的 EJB，使其成为“基础 EJB”：

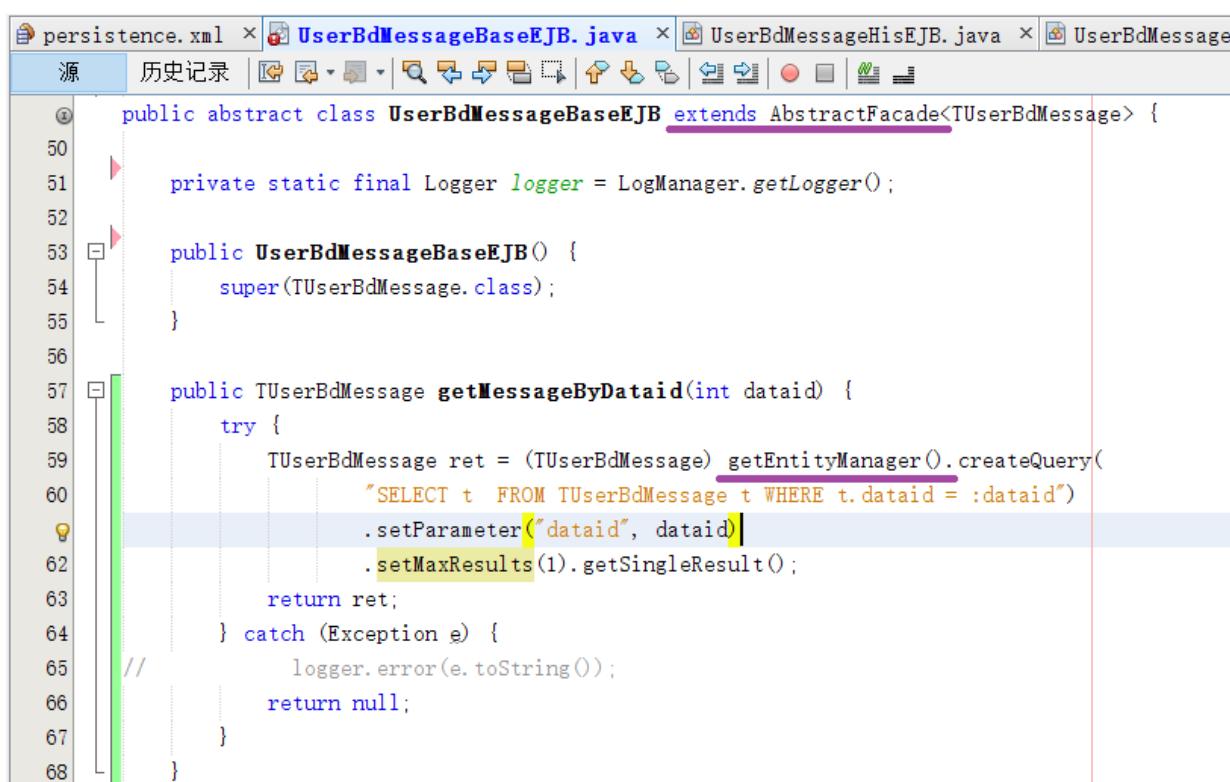
- 1) 将它自己改为抽象类型。（抽象工厂模式）
- 2) 将 EntityManager 定义为抽象类型，将所有引用 EntityManager 的地方改为“getEntityManager()”
- 3) 所有数据处理逻辑仍然保留在基础 EJB 中。
- 4) 所有读写分离的基础 EJB 都在包 com.qxsk9.db.ejb.base 中：



```

9 import javax.persistence.EntityManager;
10
11 /**
12 *
13 * @author mara
14 */
15 public abstract class AbstractFacade<T> {
16
17     private Class<T> entityClass;
18
19     public AbstractFacade(Class<T> entityClass) {
20         this.entityClass = entityClass;
21     }
22
23     protected abstract EntityManager getEntityManager();
24
25     public void create(T entity) {
26         getEntityManager().persist(entity);
27     }

```



```

49
50     private static final Logger logger = LogManager.getLogger();
51
52
53     public UserBdMessageBaseEJB() {
54         super(TUserBdMessage.class);
55     }
56
57     public TUserBdMessage getMessageByDataid(int dataid) {
58         try {
59             TUserBdMessage ret = (TUserBdMessage) getEntityManager().createQuery(
60                 "SELECT t FROM TUserBdMessage t WHERE t.dataid = :dataid")
61                 .setParameter("dataid", dataid)
62                 .setMaxResults(1).getSingleResult();
63             return ret;
64         } catch (Exception e) {
65             logger.error(e.toString());
66             return null;
67         }
68     }

```

3.16.4 定义“持久层单元的 EJB”

针对不同的持久层单元、分别定义 EJB:

- 1) 继承基础 EJB。
- 2) 注入持久层、实现 EntityManager()
- 3) 注入其它读写分离的 EJB、实现这些 EJB 的 get 方法。
- 4) 所有连接主库 qxsk9_db 的主从分离的 EJB 都在包 com.qxsk9.db.ejb.intime 中:

```

1  ...
2  package com.qxsk9.db.ejb.intime;
3
4  import com.qxsk9.db.ejb.base.UserBdMessageBaseEJB;
5  import javax.ejb.Stateless;
6  import javax.persistence.EntityManager;
7  import javax.persistence.PersistenceContext;
8
9  /**
10  *
11  * @author mara
12  */
13  @Stateless
14  public class UserBdMessageIntimeEJB extends UserBdMessageBaseEJB {
15
16      @PersistenceContext(unitName = "qxsk9_db")
17      private EntityManager em;
18
19
20      @Override
21      protected EntityManager getEntityManager() {
22          return em;
23      }
24
25  }

```

- 4) 所有连接历史库 qxsk9_db_his 的主从分离的 EJB 都在包 com.qxsk9.db.ejb.his 中:

```

17  @Stateless
18  public class UserBdMessageHisEJB extends UserBdMessageBaseEJB {
19
20      @PersistenceContext(unitName = "qxsk9_db_his")
21      private EntityManager em_his;
22
23      @Override
24      protected EntityManager getEntityManager() {
25          //      em_his.getEntityManagerFactory().getCache().evictAll();
26          return em_his;
27      }
28
29      public UserBdMessageHisEJB() {
30
31  }
32

```

3.17 “Stale Read”问题

3.17.1 问题是什么？

主从架构的读写分离会产生“Stale Read”（过期读）问题：由于某种原因（延迟、缓存），从库的查询并未同步主库对数据的修改，返回的是过期数据。

以下网页讨论了不同类型数据库中因为延迟而产生的过期读问题：

<https://brandur.org/postgres-reads>

<https://aphyr.com/posts/322-jepsen-mongodb-stale-reads>

<https://kristiannielsen.livejournal.com/18308.html>

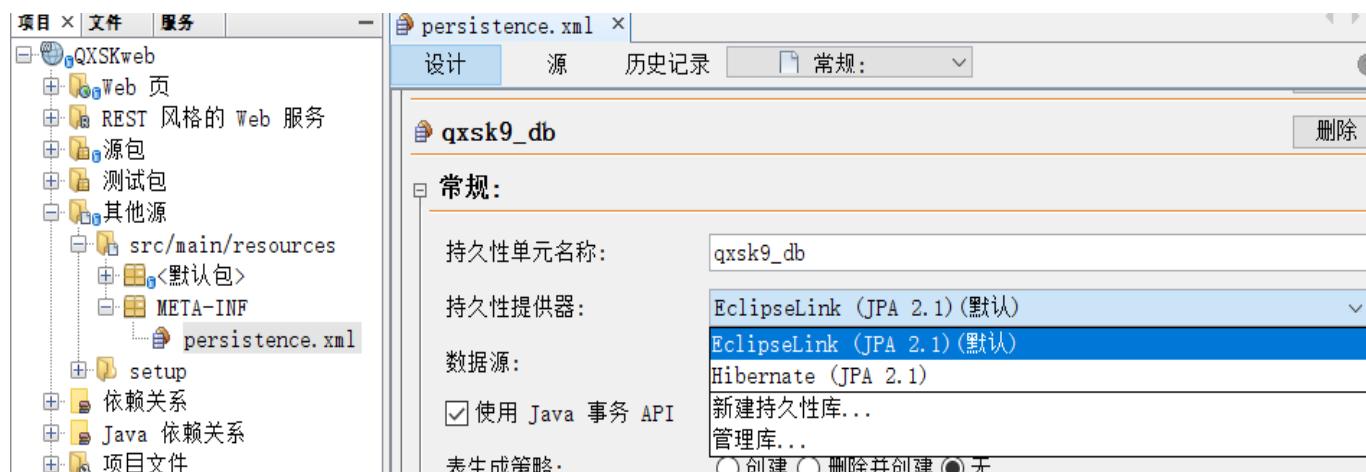
以下网页谈论了 JPA 缓存导致的过期读问题：

https://en.wikibooks.org/wiki/Java_Persistence/Caching

<http://wiki.eclipse.org/EclipseLink/Examples/JPA/Caching>

3.17.2 观察到的现象

本系统采用的是缺省的 JPA provider，支持二级缓存：



在我的实践里，碰到了因 JPA 缓存导致的过期读：在 GlassFish 中通过 JPA 配置的 mysql 主从数据库，在所有参数为缺省值时，则 Stale Read 问题是常态，即数据修改后、查询结果通常是过期的，而且并不会等待一段时间后就能获取实时数据。

一个实例：

- 1) 在网页上查询一条数据记录，这是从库的 JPA 查询操作。
- 2) 在网页的数据列表中修改这条记录，这是主库的 JPA 写操作。
- 3) 直接检查数据库内容，看到主库和从库里这条记录都被更新了。说明主库的 JPA 缓存同步了主库，并且主库与从库也做到了同步。
- 4) 但是网页查询这条记录仍然是修改前的内容。说明从库的 JPA 缓存未能与从库同步。

问题出在：**从库的 JPA 缓存不能感知从库中数据的修改，因为从库的修改来源于外界（主从同步）而不是从库的 JPA 写操作。**

同时还能观察到：主库 JPA 的增删操作，总是能被从库 JPA 感知到。即增删数据不会发生 stale read 问题。

3.17.3 官方的解决办法

解决 Stale Read 的官方方法可以是如下之一（如上面网页所诉）：

- 1) 修改 persistence.xml 来关闭二级缓存：

<https://stackoverflow.com/questions/2809275/disable-caching-in-jpa-eclipselink>

<https://stackoverflow.com/questions/22149130/jpa-eclipselink-database-change-notification-does-not-invalidate-cache-entry>

但是在我的环境中，设置 persistence.xml 没有用。

- 2) 刷新 JPA 缓存，问题可以得到解决（参见《The Java EE Tutorial Release 7》第 44 章）：

`getEntityManager().getEntityManagerFactory().getCache().evictAll();`

刷新 JPA 缓存不是好的解决办法：缓存没了存在意义。

- 3) “优化锁”也不是好办法，那需要大量修改代码并且降低性能。

- 4) 对表单独设置是否缓存。这个办法也需要大量修改代码而且不如直接关闭缓存。

3.17.4 我的解决办法

有没有办法既能受益于高速缓存、又能避免 stale read 问题？

可以观察到：“如果一种数据只增删不更改，则这类数据不存在过期读的问题”。非常幸运，在很多实时数据流系统（包括定位系统）中，持续累计的数据都是只增删不修改的。

所以，我的解决方案是：

- 1) 对系统中的数据加以分类，分析哪些数据只增删而从不修改。
- 2) 对于只增删不修改的数据做读写分离，并且开启 JPA 缓存。
- 3) 对会发生更改的数据要么只在主库上操作、要么关闭 JPA 缓存。

在本系统中，随时间大量积累的 6 个大表都是只增删不修改，可以做到开启 JPA 缓存同时避免 stale read 问题。

查看 用户反馈

数据编号	434
用户反馈的对象	平台服务器
用户反馈类型	性能优化
标题	读写分离、实时同步、缓存、与“陈旧数据”的权衡
描述	经过实践得出结论，适合读写分离的数据表应当：只增或删，从不修改。对这样的数据做读写分离，既能获益于数据实时同步、缓存、又能避免“陈旧数据”（stale read）。在大多数实时监控系统中，大量累积的数据恰恰都是这样的特征：只增删不修改。
报告者	任珊虹
报告时间	2017-11-24 19:29:38
响应说明	在我们的系统中，随时间大量累增的五个数据表都是只增不改，因此都可以读写分离、实时同步、开启缓存、同时避免“陈旧数据”。
响应时间	2017-12-22 16:42:19
响应者	任珊虹
用户反馈状态	已解决
用户反馈的来源	网页客户端
解决的版本	平台服务器-5.000

3.18 事务

3.18.1 事务界定

本系统中所有 EJB 都采用缺省的策略，即容器托管的事务界定（container-managed transaction demarcation）：

- 1) 当 EJB 方法开始时容器立即开始一个事务，当方法退出时提交这个事务。
- 2) 一个方法可以与一个事务相关，不允许嵌套事务或多个事务。

查看 用户反馈	
数据编号	421
用户反馈的对象	平台服务器
用户反馈类型	改进建议
标题	避免同一事务涉及多个数据库连接
描述	目前的驱动是非分布式的（non-XA），若事务涉及写，则其读也应当连接实时库。
报告者	任珊虹
报告时间	2017-11-16 13:24:27
响应说明	
响应时间	2017-11-16 16:18:36
响应者	任珊虹
用户反馈状态	已解决
用户反馈的来源	网页客户端
解决的版本	平台服务器-4.600

3.18.2 事务属性

本系统中绝大多数 EJB 都没有定义事务属性，即采用缺省的事务属性 Required:

- 1) 如果客户端运行在一个事务中，当它调用一个 EJB 的方法时，则此方法在该客户端的事务中执行。
- 2) 如果客户端无事务，当它调用一个 EJB 的方法时，容器在运行方法之前开始一个新事务。

只有少数几个 EJB 在创建新数据时加了事务属性 RequiredNew:

- 1) 如果客户端运行在一个事务中，当它调用一个 EJB 的方法时，容器执行以下步骤：
 - (1) 挂起客户端的事务
 - (2) 开始新的事务
 - (3) 代理对该方法的调用
 - (4) 在方法结束时恢复客户端的事务
- 2) 如果客户端无事务，当它调用一个 EJB 的方法时，容器在运行方法之前开始一个新事务。

这些 EJB 采用事务属性 RequiredNew 的原因是：需要新创建的数据立即被提交生效、以便后面的步骤引用。例如，包含自增关键字的数据，后续操作需要引用它的关键字，则需要提交事务才能获得关键字的值。

```

64
65     @TransactionAttribute(REQUIRES_NEW)      方法结束时立即提交事务
66     public boolean createAndSubmit(CTerminalFence tf) {
67         try {
68             this.create(tf);
69             return true;
70         } catch (Exception e) {
71             logger.error(e.toString());
72             return false;
73         }
74     }

```

查看 用户反馈

数据编号	241
用户反馈的对象	平台服务器
用户反馈类型	问题报告
标题	终端发给用户的消息被记录了两次
描述	这个问题好像以前出现过，又出现了。。。
报告者	任珊虹
报告时间	2016-11-05 09:31:25
响应说明	原因是创建时数据实体在缓存里没有写入数据库。解决办法是先flush数据再refresh实体。
响应时间	2016-11-06 12:56:47
响应者	任珊虹
用户反馈状态	已解决
用户反馈的来源	网页客户端
解决的版本	-

3.19 报表设计环境 Jaspersoft Studio

3.19.1 资源地址

下载地址:

<https://community.jaspersoft.com/project/jaspersoft-studio>

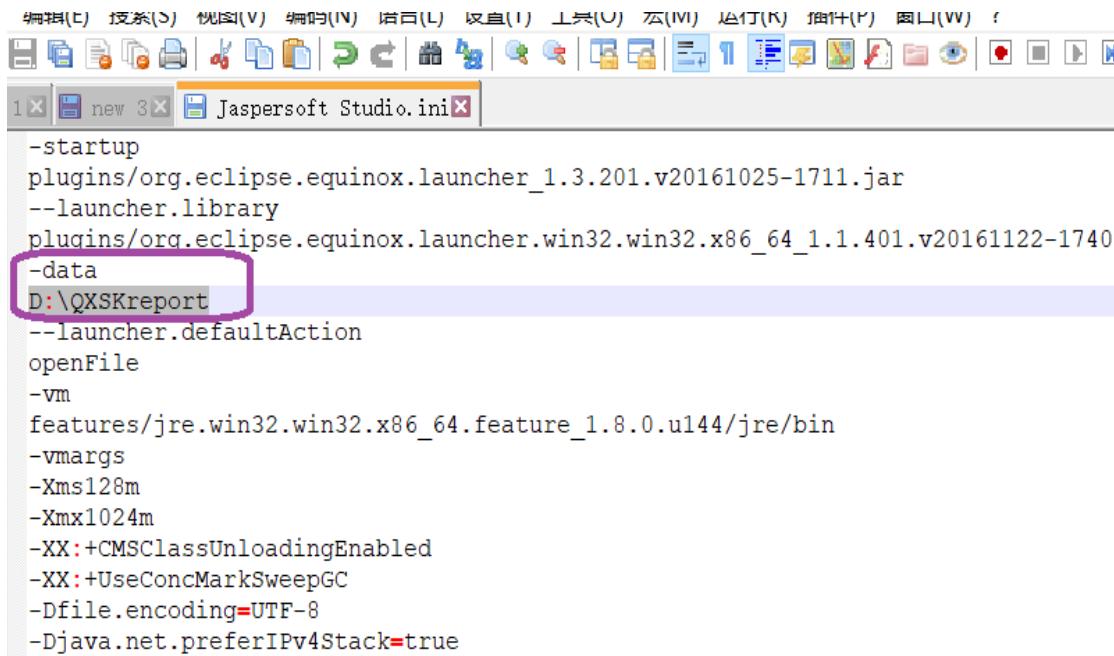
3.19.2 修改 workspace 路径

参见:

<https://community.jaspersoft.com/wiki/tibco-jaspersoft-studio-how-configure-non-default-path-jaspersoft-studio-workspace>

即编辑“安装目录/Jaspersoft Studio.ini”，修改为：

```
-data
D:\QXSKreport
```

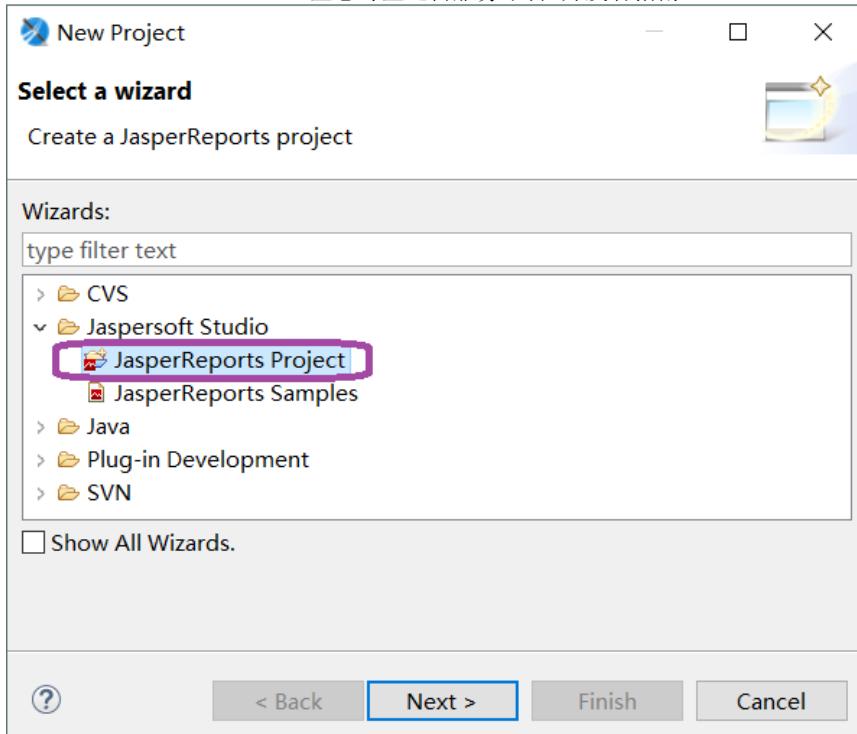


```
-startup
plugins/org.eclipse.equinox.launcher_1.3.201.v20161025-1711.jar
--launcher.library
plugins/org.eclipse.equinox.launcher.win32.win32.x86_64_1.1.401.v20161122-1740
-data
D:\QXSKreport
--launcher.defaultAction
openFile
--vm
features/jre.win32.win32.x86_64.feature_1.8.0.u144/jre/bin
--vmargs
-Xms128m
-Xmx1024m
-XX:+CMSClassUnloadingEnabled
-XX:+UseConcMarkSweepGC
-Dfile.encoding=UTF-8
-Djava.net.preferIPv4Stack=true
```

然后重启 Studio。

3.19.3 导入报表模板文件

- 1) 新建项目。选择菜单项“File → New → Project”。
- 2) 选择“JasperReports Project”:



3) 项目创建后，确保三类系统库文件已显示在项目名下。

4) 将模板文件（扩展名为 jrxml）的目录整体复制进项目下：

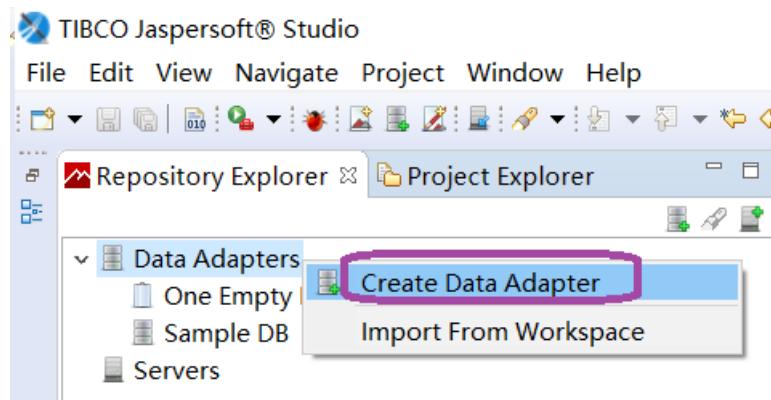
The screenshot shows the TIBCO Jaspersoft Studio interface. The top menu bar includes File, Edit, View, Navigate, Project, Window, and Help. The toolbar below has various icons for file operations. The left side features the 'Project Explorer' panel, which shows a project structure with a 'QXSKreport' folder containing 'JRE System Library [JavaSE-1.8]', 'JasperReports Library', 'Jaspersoft Server Library', and a 'report' folder. Inside 'report', there are several files: 'coffee_stain.png', 'coffee.jpg', 'flower1.png', 'leaf_banner_green.png', 'qxsk9_bar_values.jrxml', 'qxsk9_line_values.jrxml', 'qxsk9_list_data_monthly.jrxml', 'qxsk9_list_fee_timely.jrxml', 'qxsk9_list_fee.jrxml', 'qxsk9_list_monthly_data.jrxml', 'qxsk9_list_monthly_fee.jrxml', 'qxsk9_list_price.jrxml', 'qxsk9_list_timely_fee.jrxml', 'qxsk9_list_timely_value.jrxml', 'qxsk9_list_yearly_fee.jrxml', 'qxsk9_pie_fee.jrxml', 'qxsk9_pie_value.jrxml', 'qxsk9_stackbar_h.jrxml', 'qxsk9_stackbar.jrxml', and 'wood.jpg'. The right side is the 'Main Report' panel, where a report is being designed. It includes a header section with '\$P {QXSK}', a main body section with '\$P {BeidouServicePlatform}', and a footer section with '\$P {report_title}'. There are also tables and other report elements. A green leaf image is placed on the page.

3.19.4 配置数据源

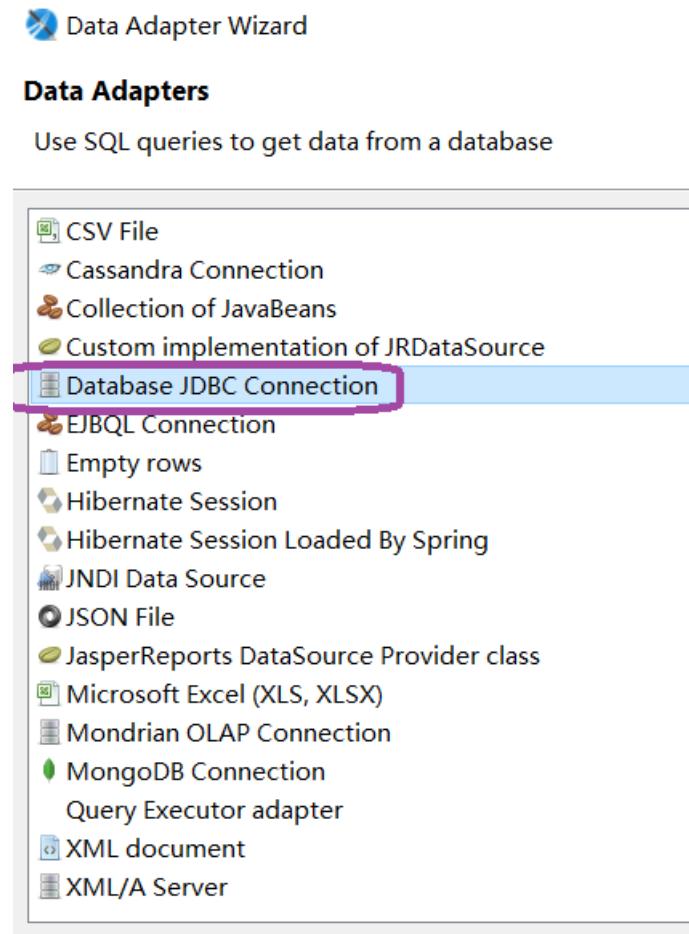
1) 下载 mysql 的 jdbc 驱动程序:

<https://dev.mysql.com/downloads/connector/j/>

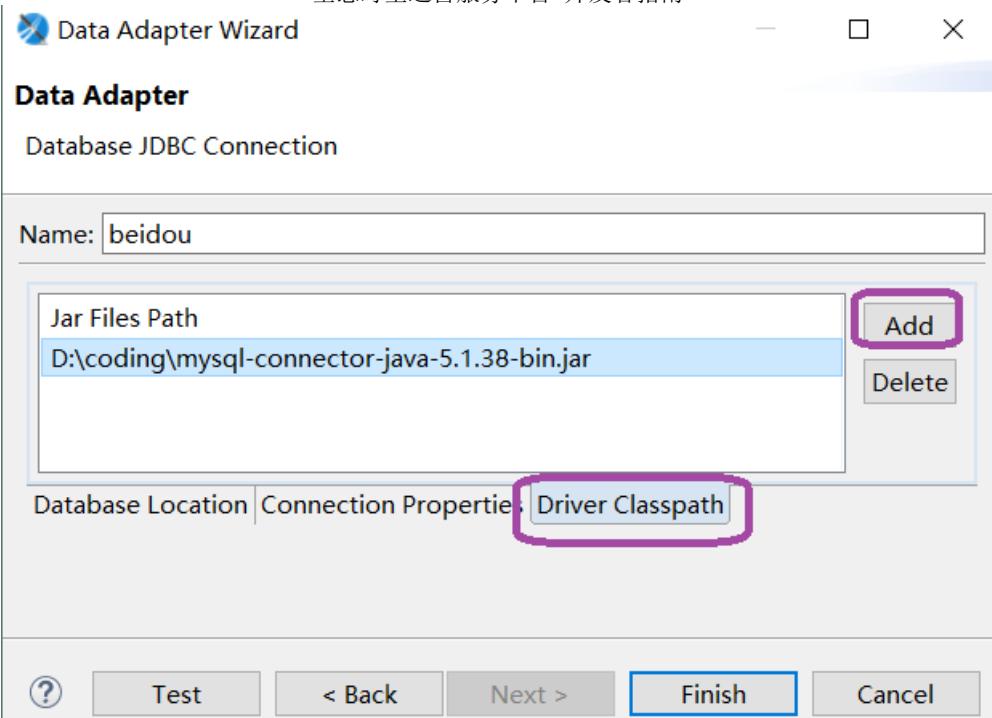
2) 选择菜单项 “File → New → Data Adapters”，或者在 Repository 浏览器视图中右键点击 “Data Adapters” 选择 “Create Data Adapter”。



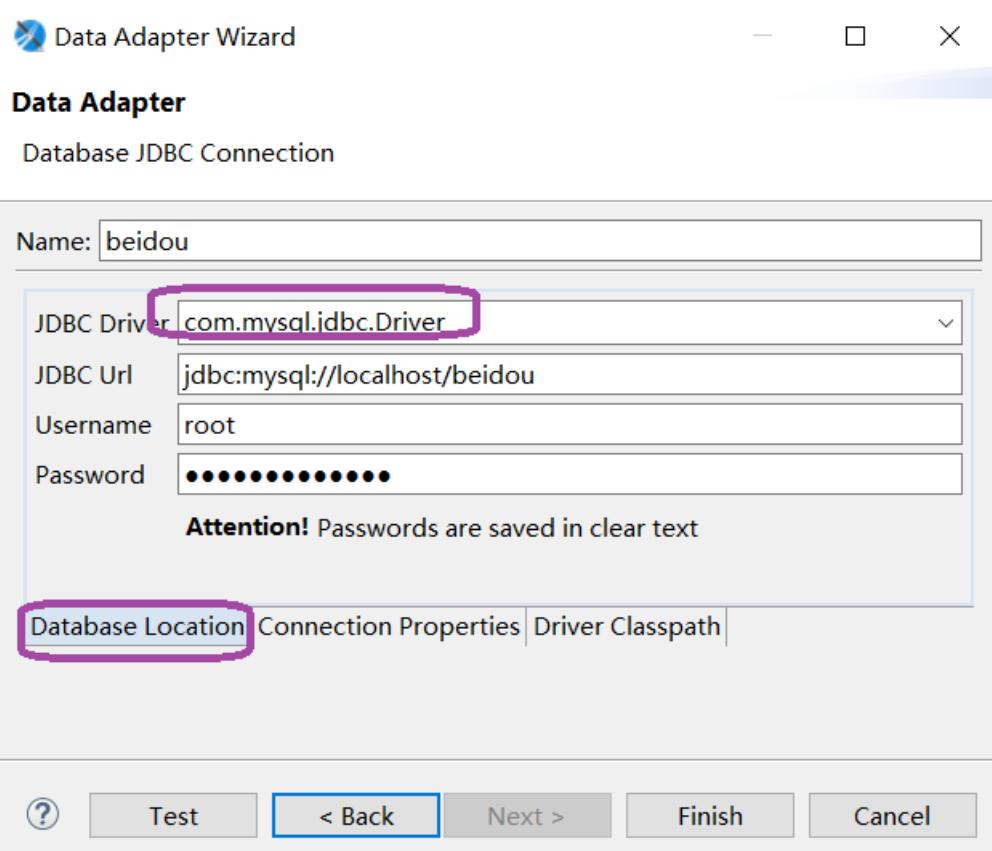
3) 选择 “Database JDBC Connection”:



4) 点击页签 “Driver Classpath”，把下载的 mysql 的 Jdbc 驱动程序加到路径里：



5) 点击页签“Database Location”，选择驱动程序“com.mysql.jdbc.Driver”，填写正确的数据库名和账户信息。



6) 点击按钮“Test”，若弹出成功窗口，则参数正确，点击按钮“Finish”以保存设置。

3.19.5 编辑报表模板源文件 jrxml

- 1) 双击模板源文件(jrxml),在编辑区打开
- 2) 在“Source”模式下直接编辑报表的参数、查询语句、数据域、和变量:

```

<parameter name="page_label" class="java.lang.String" isForPrompting="false">
<defaultValueExpression><![CDATA[“费用的单位为“人民币”。费用的单价请参考“价格表”]]></defaultValueExpression>
<parameter name="query_condition" class="java.lang.String" isForPrompting="false">
<defaultValueExpression><![CDATA[["locale='zh_CN' AND key3 = '费用' AND key4 = '费用' AND time_type = '年' AND key1 >= '2018-01-01' AND key1 <= '2018-12-31']]></defaultValueExpression>
<parameter name="queryString" language="SQL">
<![CDATA[SELECT key1, key2,
DATE_FORMAT(time_value,'%Y') as timeValue, statistic_type, count_int,count_double
FROM beidou.t_statistic_data_locales
WHERE $P{query_condition}
ORDER BY timeValue , key1, key2,statistic_type]]>
</queryString>
<field name="key1" class="java.lang.String"/>
<field name="key2" class="java.lang.String"/>
<field name="timeValue" class="java.lang.String"/>
<field name="statistic_type" class="java.lang.String"/>
<field name="count_int" class="java.lang.Integer"/>
<field name="count_double" class="java.lang.Double"/>
<variable name="total_count" class="java.lang.Integer" calculation="Sum">
<variableExpression><![CDATA[$F{count_int}]]></variableExpression>
<initialValueExpression><![CDATA[0]]></initialValueExpression>
</variable>
<variable name="time_count_total" class="java.lang.Integer" resetType="Group" resetGroup="GroupTime" calculation="Sum">
<variableExpression><![CDATA[$F{count_int}]]></variableExpression>
<initialValueExpression><![CDATA[0]]></initialValueExpression>
</variable>
<variable name="time_fee_total" class="java.lang.Double" resetType="Group" resetGroup="GroupTime" calculation="Sum">
<variableExpression><![CDATA[$F{count_double}]]></variableExpression>
<initialValueExpression><![CDATA[0.0]]></initialValueExpression>
</variable>

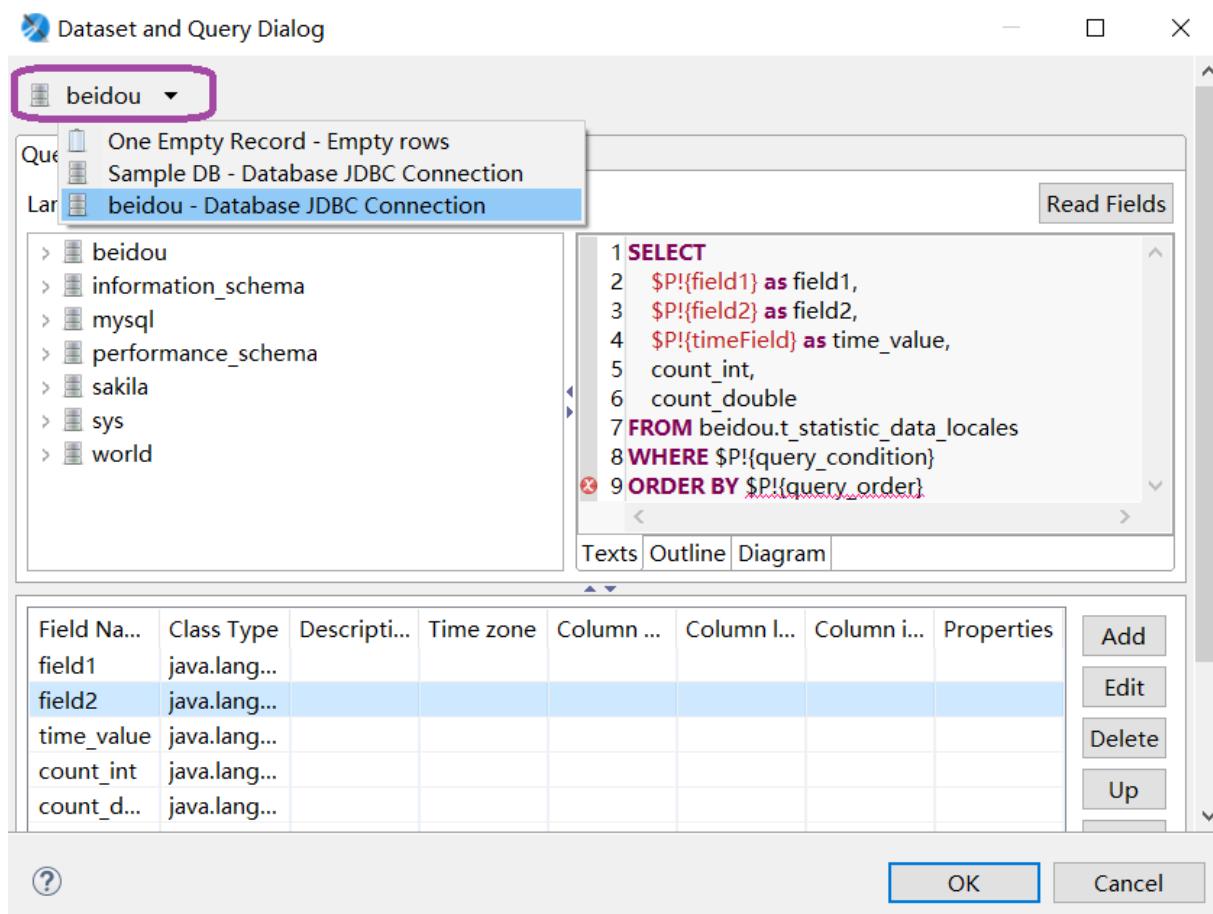
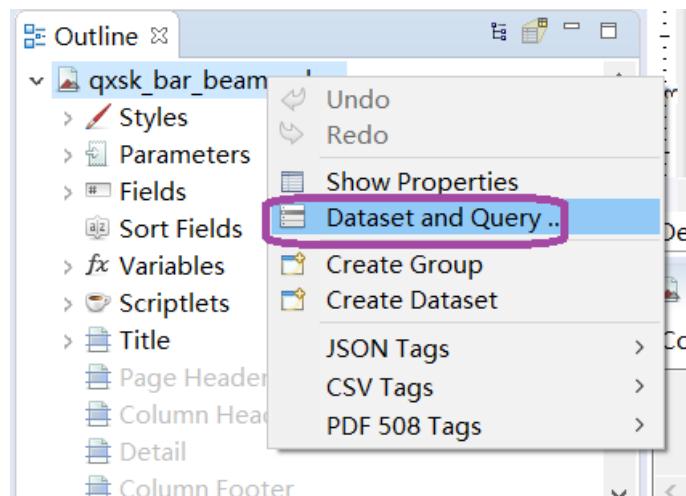
```

Design Source Preview

3) 在“Design”模式下插入、拖动、调整报表元素

The screenshot shows the Jaspersoft Studio interface in Design mode. The left sidebar displays the project structure with files such as 'qxsk9_list_charging_yearly.jrxml', 'qxsk9_pie_fee.jasper', and 'qxsk9_pie_fee.jrxml'. The main workspace shows the report layout for 'qxsk9_list_yearly_fee.jrxml'. The layout includes a title section featuring an image of coffee beans, the text '\$P{QXSK}', and '\$P{report_title}'. Below the title is a group header section with various fields and formulas. The bottom navigation bar shows tabs for 'Design', 'Source', and 'Preview'.

4) 在“Outline”视图（左下角）里右键点击报表名，选择“Dataset and Query”，为报表选择刚创建的数据源。编辑报表的查询语句，并且增删改数据域。



5) 在“Preview”模式下选择数据源，运行报表，则数据库中的数据被读入、生成报表实例：

全息时空
北斗服务运营平台

用户收发北斗消息的月统计报告

用户	月份	操作	数目	费用
所有用户				
	2015-11	用户向终端发送普通北斗消息	5	0.00
	2015-12	用户向终端发送普通北斗消息	3	0.00
		用户向终端发送紧急北斗消息	1	0.00

Design Source Preview

Report State

Console Errors (0) Statistics

Compilation Time 0.011 sec
Filling Time 0.495 sec
Report Execution Time 0.679 sec

3.19.6 编译报表模板文件 jasper

选择菜单项“Project → Build All”，则所有模板源文件 jrxml 被编译成模板文件 jasper

TIBCO Jaspersoft® Studio

File Edit View Navigate Project Window Help

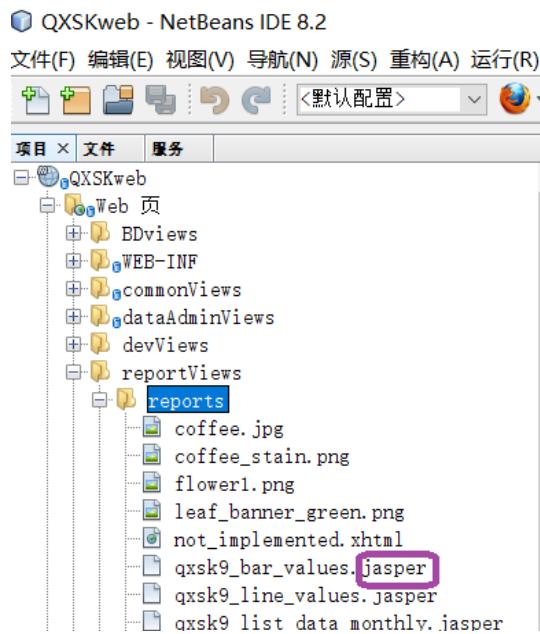
Repository Explorer

- QXSReport
- JRE System Library
- JasperReports Library
- Jaspersoft Server Library
- report
- coffee_stain.png
- coffee.jpg
- flower1.png
- leaf_banner_green.png
- qxsk9_bar_value.jasper
- qxsk9_bar_values.jrxml
- qxsk9_line_values.jasper
- qxsk9_line_values.jrxml
- qxsk9_list_data_monthly.jasper
- qxsk9_list_data_monthly.jrxml
- qxsk9_list_fee_timely.jasper
- qxsk9_list_fee_timely.jrxml

3.20 生成统计报表

3.20.1 引用报表模板 Jasper 文件

将编译好的 Jasper 文件复制到 NetBeans 项目相应的资源目录下，注意报表引用的图片文件也要复制在同一目录下：



3.20.2 初始化报表对象

1) 类 com.qxsk9.object.StatisticReport 定义了报表对象的属性：

```

15  public class StatisticReport {
16
17      private String locale;
18      private String name, type, source_file, out_file;
19      private Map<String, Object> parameters = new HashMap();

```

2) 类 com.qxsk9.object.StatisticReports 的方法 defineAllReports 初始化系统所有的报表对象：

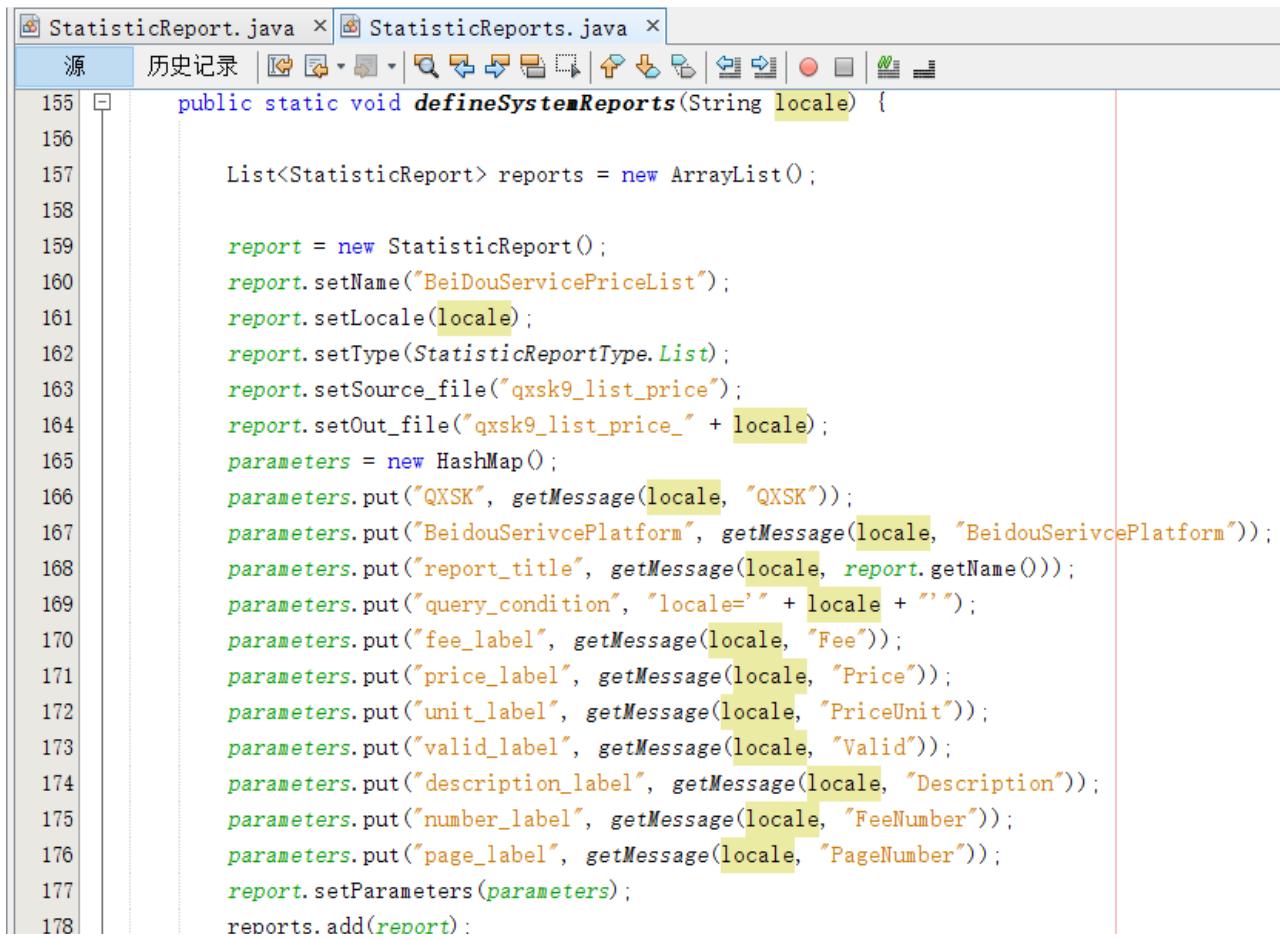
```

143         for (String locale : CommonValues.SupportedLanguages) {
144             // logger.debug(locale);
145             defineSystemReports(locale);
146             defineTerminalReports(locale);
147             defineUserReports(locale);
148             definePhoneReports(locale);
149             defineBeamReports(locale);
150             defineVersionReports(locale);
151             defineUserfeedReports(locale);
152         }

```

(这个方法整个应用只执行一次)

3) 初始化报表对象的示例如下，报表对象的属性、和报表模板的参数都被赋值，并且字串都是本地化的：



```

155 public static void defineSystemReports(String locale) {
156
157     List<StatisticReport> reports = new ArrayList();
158
159     report = new StatisticReport();
160     report.setName("BeiDouServicePriceList");
161     report.setLocale(locale);
162     report.setType(StatisticReportType.List);
163     report.setSource_file("qxsk9_list_price");
164     report.setOut_file("qxsk9_list_price_" + locale);
165     parameters = new HashMap();
166     parameters.put("QXSK", getMessage(locale, "QXSK"));
167     parameters.put("BeidouSerivcePlatform", getMessage(locale, "BeidouSerivcePlatform"));
168     parameters.put("report_title", getMessage(locale, report.getName()));
169     parameters.put("query_condition", "locale=" + locale + "'");
170     parameters.put("fee_label", getMessage(locale, "Fee"));
171     parameters.put("price_label", getMessage(locale, "Price"));
172     parameters.put("unit_label", getMessage(locale, "PriceUnit"));
173     parameters.put("valid_label", getMessage(locale, "Valid"));
174     parameters.put("description_label", getMessage(locale, "Description"));
175     parameters.put("number_label", getMessage(locale, "FeeNumber"));
176     parameters.put("page_label", getMessage(locale, "PageNumber"));
177     report.setParameters(parameters);
178     reports.add(report);

```

3.20.3 生成统计数据

报表模板中的查询语句都针对系统的统计数据表，因此在生成报表之前需要生成统计数据：

1) 类 com.qxsk9.db.ejb.extented.StatisticDataEJB 的方法 calculateAllData 重新生成所有统计数据：



```

875 public void calculateAllData() {
876     try {
877         logger.debug("正在计算所有统计数据。。。。");
878         clearAllDataExceptFence();
879         countAllChargingData();
880         countBeamsData();
881         countUserFeedbackData();
882         countVersionsData();
883
884         summarizeData();
885
886         statisticDataLocalesEJB.generateData();
887         chargingSettingLocalesEJB.generateData();
888         logger.debug("所有统计数据已生成。");
889     } catch (Exception ex) {
890         logger.error(ex.toString());
891     }
892 }

```

2) 以下是生成统计数据的示例，把基本数据表的统计数据写入统计表：

```

413     public void countBeamsData() {
414         try {
415             String timeType = StatisticTimeType.Month;
416             String dateFormat = getTimeFormat(timeType);
417             String currentTime = DateTools.getNowChinaDatetime();
418
419             String sql = "REPLACE INTO t_statistic_data (statistic_type, record_time, key1, key2, key3,
420                     + " SELECT "
421                     + " '" + StatisticType.BdBeam + "', "
422                     + " '" + currentTime + "', "
423                     + " '" + BdBeamType.BdBeamTime + "' as key1, time_beam as key2, '', '', '', '' ,
424                     + " '" + timeType + "' ,
425                     + " DATE(DATE_FORMAT(record_time,'" + dateFormat + "')) as time_value, count(dataid)
426                     + " FROM t_bd_beam "
427                     + " group by time_value, key2";
428
429             // logger.debug(sql);
430             em.createNativeQuery(sql).executeUpdate();

```

3) 对统计表的数据进行降维统计，生成更大粒度的统计数据：

```

829     public void summarizeData() {
830
831         try {
832             String sql;
833             String currentTime = DateTools.getNowChinaDatetime();
834
835             // 由月数据产生年数据
836             sql = "REPLACE INTO t_statistic_data (statistic_type, record_time, key1, key2, key3, key4, key5, key6, time_value, sum(count_int), sum(count_double))
837                     + " select "
838                     + " statistic_type , "
839                     + " '" + currentTime + "' , "
840                     + " key1, key2, key3, key4, key5, key6, '" + StatisticTimeType.Year +
841                     + " DATE(DATE_FORMAT(time_value,'%Y-01-01')) as timeValue, sum(count_int),
842                     + " FROM t_statistic_data "
843                     + " WHERE time_type = '" + StatisticTimeType.Month + "' "
844                     + " group by timeValue, statistic_type, key1, key2, key3, key4, key5
845
846             // logger.debug(sql);
847             em.createNativeQuery(sql).executeUpdate();
848
849             // 由年数据产生所有数据
850             sql = "REPLACE INTO t_statistic_data (statistic_type, record_time, key1, key2, key3, key4, key5, key6, time_value, sum(count_int), sum(count_double))
851                     + " select "
852                     + " statistic_type , "
853                     + " '" + currentTime + "' , "
854                     + " key1, key2, key3, key4, key5, key6, '" + StatisticTimeType.AllTime +
855                     + " '2000-01-01', sum(count_int), sum(count_double) "
856                     + " FROM t_statistic_data "
857                     + " WHERE time_type = '" + StatisticTimeType.Year + "' "
858                     + " group by statistic_type, key1, key2, key3, key4, key5, key6 ";
859
860             // logger.debug(sql);
861             em.createNativeQuery(sql).executeUpdate();

```

4) 类 com.qxsk9.db.ejb.extented.StatisticDataLocalesEJB 的方法 generateData 将统计数据写成不同语言的版本：（报表中引用的部分统计数据与语言有关，例如，Year 在中文报表中应该显示为“年”）

```

88     List<TStatisticData> allData = statisticDataEJB.getAllData(); 读取统计表所有数据行
89     for (String language : CommonValues.SupportedLanguages) {
90         for (TStatisticData statisticData : allData) { 对不同语言进行数据本地化
91             TStatisticDataLocales localeData = new TStatisticDataLocales();
92             localeData.setTStatisticDataLocalesPK(new TStatisticDataLocalesPK());
93             localeData.getTStatisticDataLocalesPK().setLocale(language);
94             localeData.getTStatisticDataLocalesPK().setStatisticType(SystemVariables.getMessage(localeData.getTStatisticDataLocalesPK().setKey1(SystemVariables.getMessage(language,
95             localeData.getTStatisticDataLocalesPK().setKey2(SystemVariables.getMessage(language,
96             localeData.getTStatisticDataLocalesPK().setKey3(SystemVariables.getMessage(language,
97             localeData.getTStatisticDataLocalesPK().setKey4(SystemVariables.getMessage(language,
98             localeData.getTStatisticDataLocalesPK().setKey5(SystemVariables.getMessage(language,
99             localeData.getTStatisticDataLocalesPK().setKey6(SystemVariables.getMessage(language,
100            localeData.getTStatisticDataLocalesPK().setTimeType(SystemVariables.getMessage(language,
101            localeData.getTStatisticDataLocalesPK().setTimeValue(statisticData.getTStatisticDataLocalesPK().setCountInt(statisticData.getCountInt()));
102            localeData.setCountDouble(statisticData.getCountDouble());
103            localeData.setCountLong(statisticData.getCountLong());
104            localeData.setRecordTime(statisticData.getRecordTime());
105            this.edit(localeData);
106        }
107    }
108 }
109 }
```

3.20.4 生成报表

类 com.qxsk9.object.StatisticReports 的方法 generateReports 生成一个报表对象的各种格式的报表文件：

```

2572     public static void generateReports(StatisticReport report) {
2573         try {
2574             JasperPrint jasperPrint = generatePrint(report); 生成报表数据对象
2575             if (jasperPrint == null) {
2576                 return;
2577             }
2578             String reportname = report.getOut_file();
2579             generateHtmlReport(jasperPrint, reportname);
2580             generatePdfReport(jasperPrint, reportname);
2581             generateXlsReport(jasperPrint, reportname);
2582         } catch (Exception e) {
2583             logger.error(e.toString());
2584         }
2585     }
2586
2587     public static JasperPrint generatePrint(StatisticReport report) {
2588         try {
2589             String infile = reportsPath + "/" + report.getSource_file() + ".jasper";
2590             JasperPrint jasperPrint = JasperFillManager.fillReport(infile, report.getParameters(), connection);
2591             return jasperPrint;
2592         } catch (Exception e) {
2593             logger.error(e.toString());
2594         }
2595     }

```

类 com.qxsk9.object.StatisticReports 的方法 generateAllReports 生成所有报表文件:

The screenshot shows a Java code editor with two tabs open: "StatisticReport.java" and "StatisticReports.java". The "StatisticReport.java" tab is active, displaying a method that iterates through various report categories (SupportedLanguages) and generates reports for each item in those categories. The code uses several static imports from CommonValues, such as SupportedLanguages, systemReports, terminalReports, userReports, phoneReports, beamReports, versionReports, and userfeedReports.

```
for (String locale : CommonValues.SupportedLanguages) {  
    for (StatisticReport item : systemReports.get(locale)) {  
        generateReports(item);  
    }  
    for (StatisticReport item : terminalReports.get(locale)) {  
        generateReports(item);  
    }  
    for (StatisticReport item : userReports.get(locale)) {  
        generateReports(item);  
    }  
    for (StatisticReport item : phoneReports.get(locale)) {  
        generateReports(item);  
    }  
    for (StatisticReport item : beamReports.get(locale)) {  
        generateReports(item);  
    }  
    for (StatisticReport item : versionReports.get(locale)) {  
        generateReports(item);  
    }  
    for (StatisticReport item : userfeedReports.get(locale)) {  
        generateReports(item);  
    }  
}
```

3.20.5 实时报表还是定时报表？

统计数据和报表的生成是一个耗时的过程，因此本系统不支持普通用户实时生成报表。目前的策略是：

- 1) 在每日子线程（DailyTask）中自动生成统计数据和所有统计报告。
 - 2) 为用户提供已生成的统计报表的连接。
 - 3) 系统管理员可以实时生成统计数据和统计报表。

3.21 应用百度地图

3.21.1 申请浏览器端的 AK

申请百度地图应用 AK 的地址如下，需要为每一种客户端申请：

<http://lbsyun.baidu.com/apiconsole/key>

The screenshot shows a list of applications in the Baidu Map API console. The columns are: Application ID, Application Name, Access Application (AK), Category, Remarks (Double-click to edit), and Configuration.

应用编号	应用名称	访问应用 (AK)	应用类别	备注信息 (双击更改)	应用配置
8937077	全息时空		浏览器端		设置 删除
8268654	北斗视野安卓版		Android端		设置 删除
8129113	北斗视野		iOS端		设置 删除

您当前创建了 3 个应用

目前的应用 AK 是我的百度账户中申请的，可以以其它账户申请新的应用 AK，然后修改程序中相应的引用。

申请浏览器端的 AK 只需要填写地址的白名单：

应用AK:

应用名称: 全息时空

应用类型: 浏览器端

启用服务:

<input checked="" type="checkbox"/> 云检索	<input checked="" type="checkbox"/> Javascript API	<input checked="" type="checkbox"/> 地点检索
<input checked="" type="checkbox"/> 正逆地理编码	<input checked="" type="checkbox"/> 普通IP定位	<input checked="" type="checkbox"/> 静态图
<input checked="" type="checkbox"/> 全景静态图	<input checked="" type="checkbox"/> 坐标转换	<input checked="" type="checkbox"/> 鹰眼轨迹
<input checked="" type="checkbox"/> 全景URL API	<input checked="" type="checkbox"/> 云逆地理编码	<input checked="" type="checkbox"/> 云地理编码
<input checked="" type="checkbox"/> 天气查询API	<input checked="" type="checkbox"/> 路线规划	<input checked="" type="checkbox"/> 批量算路
<input checked="" type="checkbox"/> 时区		

qxsk9.com

Referer白名单:

只有该白名单中的网站才能成功发起调用

格式: *.mysite.com*, *myapp.com* 多个域名之间请用英文半角逗号隔开

如果不想对任何域名做限制，设置为英文半角星号*(谨慎使用，前端代码容易泄露AK，被其他网站非法调用，上线前可以用作Debug，线上正式ak请设置合理的Referer白名单)

新申请的Mobile类型的ak与新申请的Browser类型的ak不再支持云存储接口的访问，如要使用云存储，请申请server类型ak

3.21.2 在网页中显示地图

以下是一个显示百度地图的网页示例：

```

<style type="text/css">
    html {height:100%}
    body {height:100%;margin:0px;padding:0px;font-size: 10pt}
    #themap {height:93%;margin:0px;padding:0px;font-size: 9pt} ————— 地图容器的属性
</style>
<script type="text/javascript" src="#{locationMapBean.mapScript}">/>
<h:form ... 25 行 />
<div id="themap"></div> ————— 地图容器
<script type = "text/javascript">

```

```

var map = new BMap.Map("themap"); // 创建地图实例
map.centerAndZoom(new BMap.Point(116.404, 39.915), #{locationMapBean.mapSize});
map.addControl(new BMap.NavigationControl()); // 缩放地图的控件，默认在左上角
map.addControl(new BMap.OverviewMapControl()); // 地图的缩略图的控件，默认在右下方；
map.addControl(new BMap.MapTypeControl()); // 地图类型控件，默认在右上方；
var opts = {offset: new BMap.Size(10, 25)};
map.addControl(new BMap.ScaleControl(opts)); // 地图显示比例的控件，默认在左下方；
map.enableScrollWheelZoom(); //开启滚轮缩放功能
map.disableDoubleClickZoom(); //关闭双击放大功能
map.enableKeyboard(); //开启键盘操作功能

```

3.21.3 在 https 链接的网页中显示百度地图

对于 https 链接的网页，引用百度地图 API 地址需要添加一个参数“&s=1”，否则地图显示为空白：

查看 用户反馈	
数据编号	249
用户反馈的对象	网页客户端
用户反馈类型	问题报告
标题	Https地址无法显示地图
描述	当使用安全链接时，地图显示为空白
报告者	任珊虹
报告时间	2016-11-23 08:52:19
响应说明	改写js地址，改用2.0接口
响应时间	2016-11-23 12:17:02
响应者	任珊虹
用户反馈状态	已解决
用户反馈的来源	网页客户端
解决的版本	-

```

112     public String getMapScript() {
113         HttpServletRequest request = (HttpServletRequest) FacesContext.getCurrentInstance().getExternalContext();
114         mapScript = request.getScheme() + "://api.map.baidu.com/api?v=2.0&ak=..."; // 替换为你的ak
115         if (mapScript.startsWith("https")) {
116             mapScript += "&s=1"; // 对于https需要添加一个参数
117         }
118         return mapScript; // 网页中引用的百度地图API地址
119     }

```

3.21.4 调用地址转换服务

需要把非百度坐标转换为百度坐标才能使位置正确显示在百度地图上：

- 1) 可以在网页上利用 javascript API 转换坐标：

http://lbsyun.baidu.com/jsdemo.htm#a5_2

- 2) 也可以在后端 bean 中利用百度的网络服务转换坐标：

<http://lbsyun.baidu.com/index.php?title=webapi/guide/changeposition>

- 3) 结论是，在后端 bean 中利用百度的网络服务转换坐标：

查看 用户反馈

数据编号	225
用户反馈的对象	网页客户端
用户反馈类型	改进建议
标题	网页上的坐标转换可以用javascript接口
描述	目前的坐标转换是服务器调用百度地图服务
报告者	任珊虹
报告时间	2016-09-09 18:15:55
响应说明	javascript接口仍然是调用了百度地图服务，而且只能一次转换10个。不如直接调用服务，一次可以转换100个。
响应时间	2016-09-09 18:17:37
响应者	任珊虹
用户反馈状态	无需解决
用户反馈的来源	网页客户端
解决的版本	-

服务器的地址转换方法在类 com.qxsk9.tool.GeographyTools 中：

```

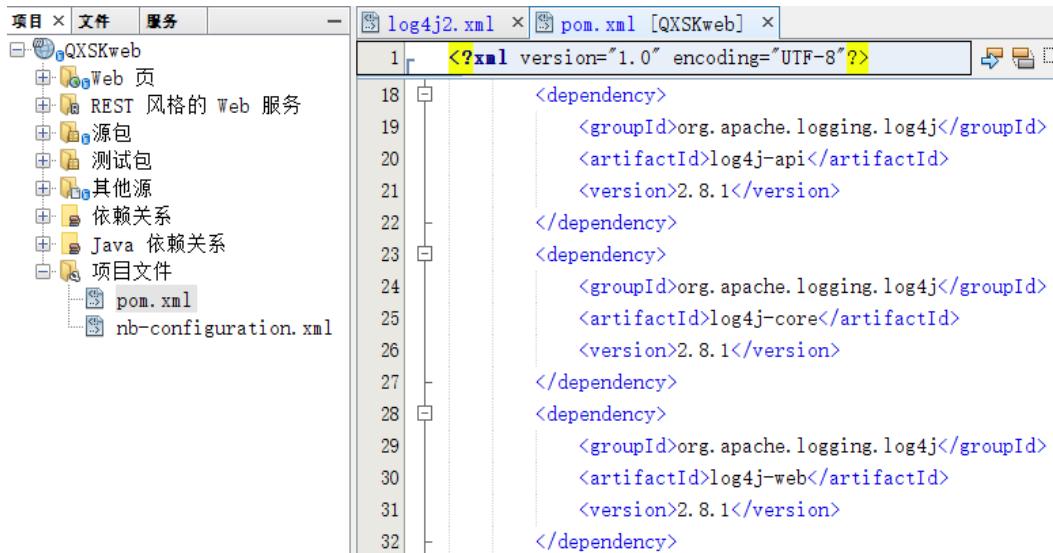
GeographyTools.java x
源 历史记录 | 保存 | 取消 | 搜索 | 全屏 | 复制 | 剪切 | 粘贴 | 重命名 | 删除 | 新建 | 重新组织 | 关闭 | 其他
516 // http://api.map.baidu.com/geoconv/v1/?coords=114.21892734521,29.575429778924;114.21892734521,29.575429778924
517 // 调用地址转换服务的一个例子
518 // {"status":0,"result":[{"x":114.23075195314,"y":29.579085915581}, {"x":114.23075059936,"y":29.579088067364}]}
519 public static List<Coordinate> calculateCoordinateOfBD09(List<Coordinate> xys) {
520     List<Coordinate> results = new ArrayList();
521     if (xys == null || xys.isEmpty()) {
522         return results;
523     }
524     String lists = "";
525     for (Coordinate xy : xys) {
526         if (lists.isEmpty()) {
527             lists = xy.getX() + "," + xy.getY();
528         } else {
529             lists = lists + ";" + xy.getX() + "," + xy.getY(); —— 把输入的坐标列表转变成符合格式的字符串
530         }
531     }
532     String baiduURL = "http://api.map.baidu.com/geoconv/v1/?coords=" + lists + "&ak=E4805d16520de693a3fe707cdc962045&output=json";
533     Coordinate xy = new Coordinate();
534     try {
535         JsonParser parser = Json.createParser(new URL(baiduURL).openStream()); —— 解析返回的json数据
536         String key = "none";
537         while (parser.hasNext()) {
538             Event event = parser.next();

```

3.22 系统的日志

3.22.1 代码记录的 log4j2 日志

1) 在 pom.xml 中加入 log4j2 的依赖:

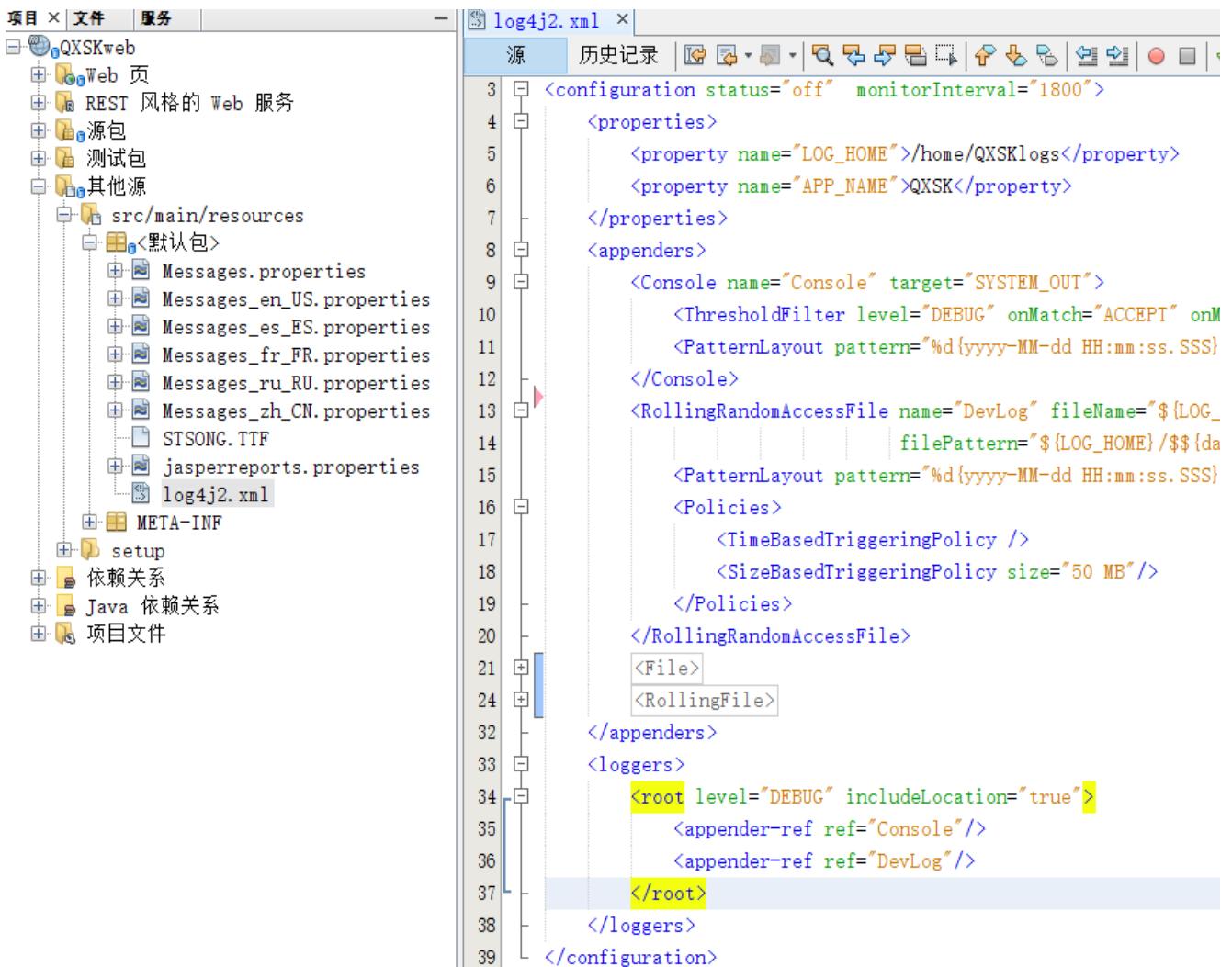


```

<dependency>
    <groupId>org.apache.logging.log4j</groupId>
    <artifactId>log4j-api</artifactId>
    <version>2.8.1</version>
</dependency>
<dependency>
    <groupId>org.apache.logging.log4j</groupId>
    <artifactId>log4j-core</artifactId>
    <version>2.8.1</version>
</dependency>
<dependency>
    <groupId>org.apache.logging.log4j</groupId>
    <artifactId>log4j-web</artifactId>
    <version>2.8.1</version>
</dependency>

```

2) 编辑配置文件 log4j2.xml, 使得调试级别的日志信息同时写入控制台和日志文件:



```

<configuration status="off" monitorInterval="1800">
    <properties>
        <property name="LOG_HOME">/home/QXSKlogs</property>
        <property name="APP_NAME">QXSK</property>
    </properties>
    <appenders>
        <Console name="Console" target="SYSTEM_OUT">
            <ThresholdFilter level="DEBUG" onMatch="ACCEPT" onOff="OFF" onMatch="ACCEPT" onOff="OFF" />
            <PatternLayout pattern="%d{yyyy-MM-dd HH:mm:ss.SSS} %msg%n" />
        </Console>
        <RollingRandomAccessFile name="DevLog" fileName="${LOG_HOME}/$${date}-%${random:0:5}.log" filePattern="${LOG_HOME}/$$${date}-%${random:0:5}.log" />
        <PatternLayout pattern="%d{yyyy-MM-dd HH:mm:ss.SSS} %msg%n" />
        <Policies>
            <TimeBasedTriggeringPolicy />
            <SizeBasedTriggeringPolicy size="50 MB" />
        </Policies>
    </RollingRandomAccessFile>
    <File>
        <RollingFile>
            <PatternLayout pattern="%d{yyyy-MM-dd HH:mm:ss.SSS} %msg%n" />
            <Policies>
                <TimeBasedTriggeringPolicy />
                <SizeBasedTriggeringPolicy size="50 MB" />
            </Policies>
        </RollingFile>
    </File>
    </appenders>
    <loggers>
        <root level="DEBUG" includeLocation="true">
            <appender-ref ref="Console"/>
            <appender-ref ref="DevLog"/>
        </root>
    </loggers>
</configuration>

```

3) 代码中写日志信息的示例:

```

33     private static final Logger logger = LogManager.getLogger();
34
35     @Override
36     public void onOpenAction(Session session, EndpointConfig conf) {
37         try {
38             synchronized (this) {
39                 LogSocketSessions = session.getOpenSessions();
40                 logger.debug("LogSocketSessions size:" + LogSocketSessions.size()); 输出调试信息
41             }
42         } catch (Exception e) {
43             logger.error(e.toString()); 输出错误信息
44         }
45     }

```

4) 以下是控制台显示的 log4j2 日志信息示例，系统初始化的日志:

代码中记录的log4j2日志

```

信息: Monitoring jndi:/server/QXSKweb/WEB-INF/faces-config.xml for modifications
信息: Running on PrimeFaces 5.0
信息: 2018-04-18 19:24:50.316 [admin-listener(2)] DEBUG com.qxsk9.system.QXSKinit contextInitialized() 44 初始化系统参数。。
信息: 2018-04-18 19:24:50.339 [admin-listener(2)] DEBUG com.qxsk9.system.QXSKinit contextInitialized() 47 初始化系统公共变量。。
信息: INFO | Loaded the Bouncy Castle security provider.
信息: WARN | Memory Usage for the Broker (1024mb) is more than the maximum available for the JVM: 455 mb - resetting to 70% of n
信息: INFO | JMX consoles can connect to service:jmx:rmi:///jndi/rmi://localhost:1099/jmxrmi
信息: INFO | Using Persistence Adapter: KahaDBPersistenceAdapter[D:\glassfish-4.1.2\glassfish\domains\qxsk9\config\activemq-data]
信息: INFO | KahaDB is version 6
信息: INFO | PListStore:[D:\glassfish-4.1.2\glassfish\domains\qxsk9\config\activemq-data\localhost\tmp_storage] started
信息: INFO | Apache ActiveMQ 5.15.2 (localhost, ID:Mara-PC-61433-1524050690857-0:1) is starting
信息: INFO | Listening for connections at: tcp://127.0.0.1:61616
信息: INFO | Connector tcp://localhost:61616 started
信息: INFO | Apache ActiveMQ 5.15.2 (localhost, ID:Mara-PC-61433-1524050690857-0:1) started
信息: INFO | For help or more information please see: http://activemq.apache.org
信息: 2018-04-18 19:24:51.246 [admin-listener(2)] DEBUG com.qxsk9.system.QXSKinit contextInitialized() 50 初始化用户权限表。。
信息: 2018-04-18 19:24:51.249 [admin-listener(2)] DEBUG com.qxsk9.db.ejb.extended.UserOperateTerminalFinalEJB loadData() 52 ref
信息: 2018-04-18 19:24:51.421 [admin-listener(2)] DEBUG com.qxsk9.system.QXSKinit contextInitialized() 53 修正终端属性。。
信息: 2018-04-18 19:24:51.505 [admin-listener(2)] DEBUG com.qxsk9.system.QXSKinit contextInitialized() 57 启动线程。。
信息: 2018-04-18 19:24:51.514 [ThreadsTask] DEBUG com.qxsk9.thread.ThreadsTask run() 47 线程ThreadsTask开始工作
信息: 2018-04-18 19:24:51.550 [ThreadsTask] DEBUG com.qxsk9.thread.DailyTask <init>() 43 每日任务将在27309秒 (7小时) 以后开始。
信息: 2018-04-18 19:24:51.551 [ThreadsTask] DEBUG com.qxsk9.thread.ThreadsTask run() 73 线程ThreadsTask结束工作。
信息: Loading application [QXSKweb] at [/QXSKweb]
信息: QXSKweb was successfully deployed in 10.393 milliseconds.

```

5) 以下是 log4j2 日志文件的示例:

名称	QXSK_dev.log - 记事本
2017-12	2018-04-18 17:44:45.096 [RunLevelControllerThread-1524044663385]
2018-01	2018-04-18 17:44:46.183 [RunLevelControllerThread-1524044663385]
2018-02	2018-04-18 17:44:47.375 [RunLevelControllerThread-1524044663385]
2018-03	2018-04-18 17:44:47.379 [RunLevelControllerThread-1524044663385]
2018-04	2018-04-18 17:44:48.088 [RunLevelControllerThread-1524044663385]
QXSK.log	2018-04-18 17:44:49.858 [RunLevelControllerThread-1524044663385]
QXSK_dev.log	2018-04-18 17:44:49.867 [ThreadsTask] DEBUG com.qxsk9.thread.Thre
QXSK_rolling.log	2018-04-18 17:44:50.026 [ThreadsTask] DEBUG com.qxsk9.thread.Dail
	2018-04-18 17:44:50.026 [ThreadsTask] DEBUG com.qxsk9.thread.Thre
	2018-04-18 17:45:03.305 [admin-listener(4)] DEBUG com.qxsk9.sys

3.22.2 服务器记录的日志

在服务器的控制台上可以查看服务器记录的日志，选择“Server”->“View Log Files”：

The screenshot shows the GlassFish Server Open Source Edition control console at localhost:4848/common/index.jsf. The left sidebar shows a tree view of server components. The 'Domain' section is expanded, and 'server (Admin Server)' is selected, highlighted with a red box. The top navigation bar has tabs: General, Resources, Properties, Monitor, Batch, and JMS Physical. Below the tabs, there are buttons: Stop, Restart, View Log Files (which is also highlighted with a red box), View Raw Log, Rotate Log, and Recover Transaction... A sub-menu window titled 'Log Viewer - Mozilla Firefox' is open over the 'View Log Files' button. It shows the URL localhost:4848/common/logViewer/logViewer.jsf?instanceName=server&loglevel=INFO&viewRes. It includes fields for 'Instance:' (set to 'server') and 'Log File:' (set to 'server.log'). The main area displays a table titled 'Log Viewer Results (40)'. The table has columns: Record Number, Log Level, Message, and Logger. Two log entries are shown:

Record Number	Log Level	Message	Logger
4789	WARNING	java.io.IOException: Connection is closed at org.glassfish.grizzly.nio.NIOConnection.assertOpen(NI... (details)	javax.enterprise.resource.webcontainer.jsf.app
4788	WARNING	JSF1064: 无法从库 javax.faces 中找到或提供资源 jsf.js。 (details)	javax.enterprise.resource.webcontainer.jsf.app

3.22.3 系统的审计日志

本系统还定义了审计日志：

- 1) 数据表 `t_system_log` 保存系统的审计日志，包含：操作类型、用户、客户端地址、时间、其它数据。
- 2) 系统执行以下操作时，都会记录审计日志：
 - (1) 收到任何客户端的登录请求
 - (2) 监听到网页客户端 session 的登入和退出。
 - (3) 收到面向用户的 WebService 请求。
 - (4) 收到 WebSocket 的连接请求。
 - (5) 发送完毕邮件。
 - (6) 发送完毕短信。
- 3) 普通用户可以通过网页客户端查看与自己有关的审计日志，点击菜单“帮助 → 系统日志”：
- 4) 管理员可以通过网页客户端查看所有审计日志：

查看 系统日志

数据编号	350
记录时间	2018-04-18 00:22:09
客户端	49.80.109.245 江苏省
用户	试用账户
终端	
来源	Web_Client
目标	Web_Server
数据途径	Web_Page
数据地址	QXSKsessions
操作	Session_Destroyed
重要性	3
字符串值 A	36b61fa9f59e653391d0d48f3481
整型值	
布尔值	<input type="checkbox"/>
字符串值 B	
字符串值 C	
字符串值 D	
长整型值	16675
时间	2018-04-17 19:44:13
返回码	

关闭

轨迹 警报 消息 999+ 开发

统计 帮助 全息时空 HOLOGRAPHY SPACE

用户	终端	来源	目标
任珊虹		Web_Client	Web_S
任珊虹		Web_Client	Data_S
任珊虹		Web_Client	Data_S
任珊虹		Web_Client	Web_S
		Web_Client	Web_S
		Web_Client	Web_S
试用账户		Web_Client	Web_S
试用账户		Web_Client	Web_S
试用账户		Web_Client	Data_S
试用账户		Web_Client	Data_S
试用账户		Web_Client	Web_S
试用账户		Web_Client	Web_S

16007070号-1

3.23 自动化测试

3.23.1 工具 selenium

3.23.1.1 资源地址

selenium 既可以用来做测试、也可以用来爬资源，是白黑两道都趁手的工具：

<http://www.51testing.com/zhuanti/selenium.html>

<https://www.cnblogs.com/zhaof/p/6953241.html>

selenium 官网

<http://www.seleniumhq.org/>

<https://github.com/SeleniumHQ>

<https://www.seleniumhq.org/download/>

selenium 文档

<https://github.com/SeleniumHQ/docs>

<http://selenium-python.readthedocs.io/index.html>

Selenium 发展史

<http://www.cnblogs.com/fnng/p/7426928.html>

3.23.1.2 关于 Selenium IDE

Selenium IDE 可以自动录制用户的操作、然后保存为自动脚本。然而我已经放弃了使用 Selenium IDE，因为我感觉直接编写脚本更顺手。

如果要在 FireFox 中使用 Selenium IDE，则会碰到麻烦：火狐 48 以后的版本支持 Selenium3，火狐 48 之前的版本支持 selenium2，而火狐 55 以后的新版本不支持 SeleniumIDE！以下是一个网友的描述：

Firefox V48.0.2（V56.0 也是 OK 的亲试）可实现完美兼容，需保证浏览器不自动更新。步骤如下：

(1) 火狐浏览器选择自定义安装（不要选择默认安装），取消勾选“安装维护服务”；

(2) 安装完成后，工具——选项——firefox 更新——勾选“不检查更新”即可

如果你已经安装了高版本的 Firefox 或者更新设置屡试不爽，可试试上面的方法！

也就是退回到旧版本。以下是旧版火狐的下载地址：

<http://ftp.mozilla.org/pub/firefox/releases/54.0/>

以下是来自官网的信息，开发者说：正在基于 SideeX 重写 Selenium IDE，SideeX 很慷慨地贡献了技术。

<https://github.com/SeleniumHQ/selenium-ide>

The screenshot shows the GitHub repository page for 'Selenium IDE'. The page title is 'Selenium IDE · build passing'. Below the title, it says '[WIP] An integrated development environment for Selenium scripts'. To the right, there's a large blue logo with the letters 'Se' and a video camera icon. The main content area discusses the project as a work in progress, a complete rewrite of the old Selenium IDE, which was a Firefox extension relying heavily on APIs no longer supported by newer versions of Firefox. It mentions that the IDE is developed as a modern browser extension supporting both Chrome and Firefox and other modern browsers (in theory). It also notes that as this is an early stage, many things aren't clear yet. A purple horizontal bar at the bottom states: 'We are using SideeX as a start point. The SideeX team was kind enough to let us use their work.'

以下是来自 SideeX 的描述：我们尊重和感谢 Selenium 开发了伟大实用的工具，现在由我们添砖加瓦了！

<http://sideex.org/>

The screenshot shows the SideeX website homepage. The URL in the address bar is 'sideex.org'. The main heading is 'SideeX - Smart Record-Playback Browser Automation'. Below it, it says 'Open Source · 100% Free' and 'Over 60,000 web testers around the world run SideeX code'. A button labeled 'Download SideeX 2 for Chrome/Firefox' is visible. The background features a colorful cartoon illustration of various characters.

What is SideeX

- In a word, SideeX [sɪdɛks] is a free and open source smart record-playback automated web application testing tool
- SideeX presents as an extended version of Selenium IDE with a number of superior expanded automation capabilities

sideex.org 的数据...

Smart Record and Playback Automations

- Smart auto-wait for playing every command. No more need for manually adding any `_AndWait` commands
- Smart auto-record/play commands for unnamed popup windows
- Smart auto-record/play commands for mouseover, scrolling, drag&drop, dropdown, rich-text editor, etc

Software Powered by SideeX

- We respect and thank Selenium Core and IDE contributors for developing the great and useful [Selenium IDE](#) tool
- Now, SideeX is used to empower the following software:
 - [The next generation of official Selenium IDE](#)
 - [Katalon Recorder](#)

来自开源世界的惺惺相惜真是让人羡慕啊！

3.23.1.3 关于 WebDriver

WebDriver 是 W3C 的一个标准，由 Selenium 主持：

<https://w3c.github.io/webdriver/webdriver-spec.html>

WebDriver 之所以能够实现与浏览器进行交互，是因为浏览器实现了这些协议：

<https://www.cnblogs.com/augus007/articles/7338195.html>

以下网址归纳了各个浏览器驱动器的下载地址：

<https://www.seleniumhq.org/download/>

firefox driver 官方地址：

<https://github.com/mozilla/geckodriver/>

<https://github.com/mozilla/geckodriver/releases/>

chrome driver 官方地址及国内镜像：

<https://sites.google.com/a/chromium.org/chromedriver/home>

<http://npm.taobao.org/mirrors/chromedriver/>

Microsoft Web Driver 官方地址：

<http://go.microsoft.com/fwlink/?LinkId=619687>

3.23.2 编程环境 pycharm

3.23.2.1 资源地址

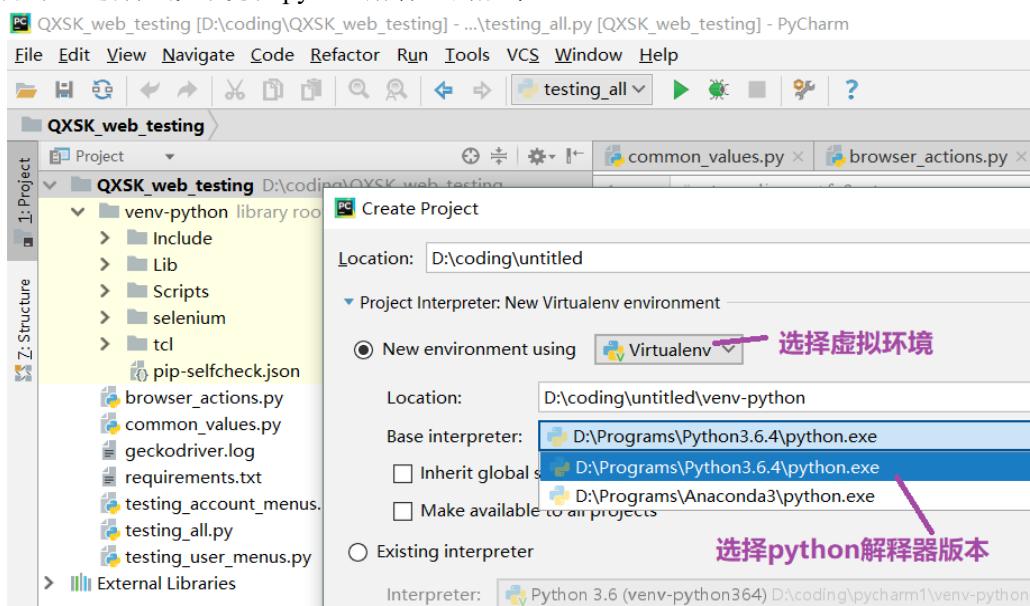
如前所述，我选择了强大的 IDE 工具 pycharm。免费开源的社区版够用，下载地址：

<https://www.jetbrains.com/pycharm/?fromMenu>

pycharm 的界面风格与 IntelliJ IDEA 一致，用过 Android Studio 的人打开 pycharm 后会感觉似乎在用 AS。

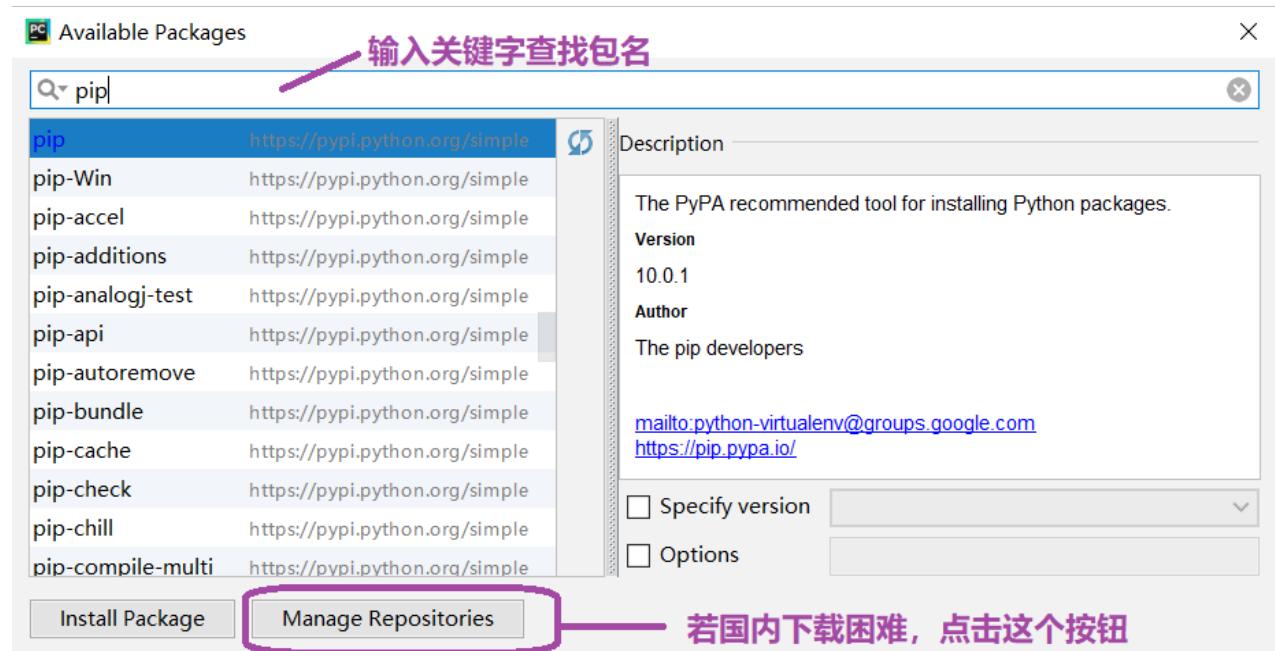
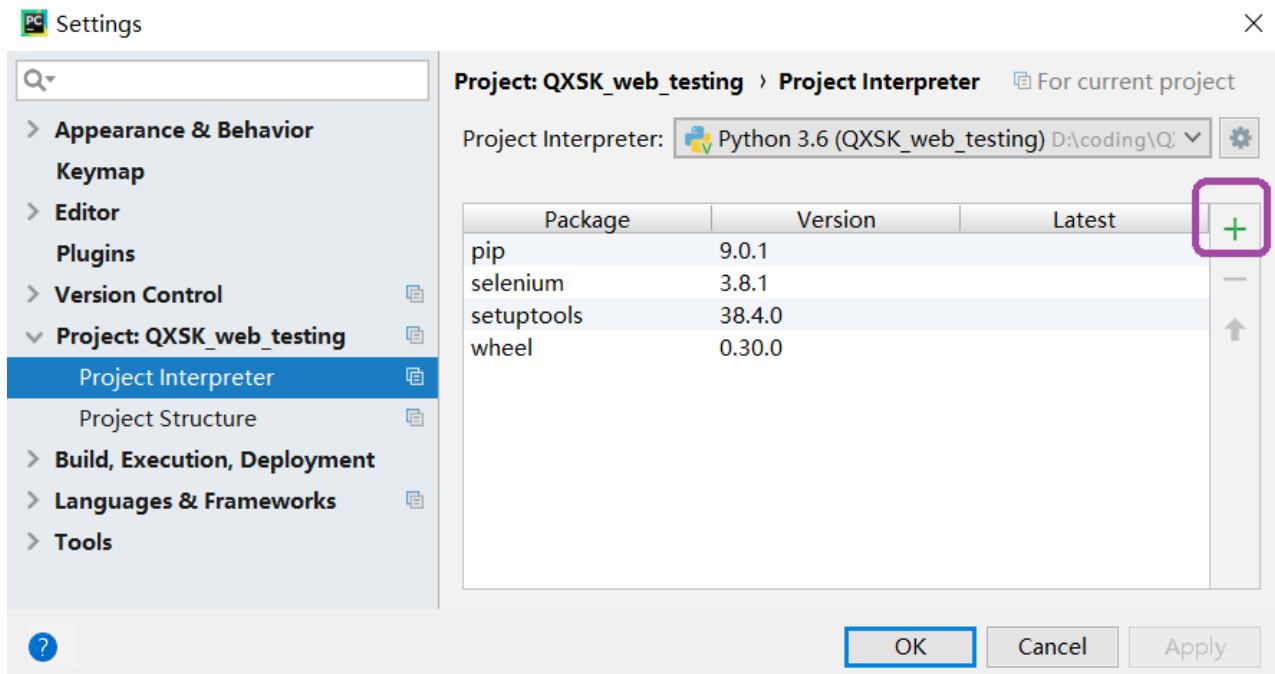
3.23.2.2 虚拟环境和 python 解释器的版本

创建项目时，选择虚拟环境和 python 解释器的版本。

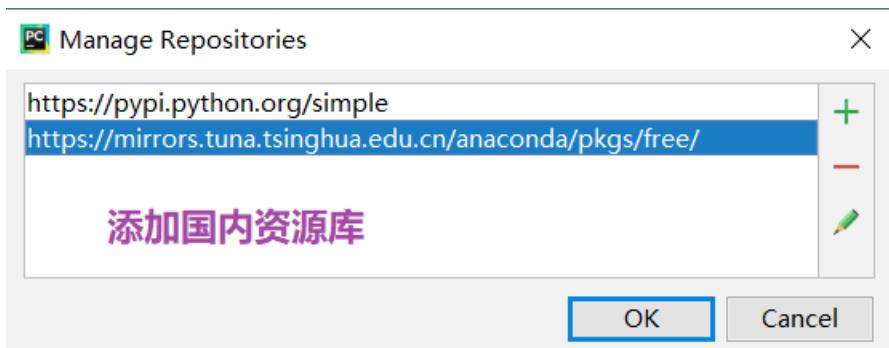


3.23.2.3 管理 Python 包和 Python 库

选择菜单项“File → Settings”，点击 Project Interpreter，若要添加 python 包则点击按钮“+”：



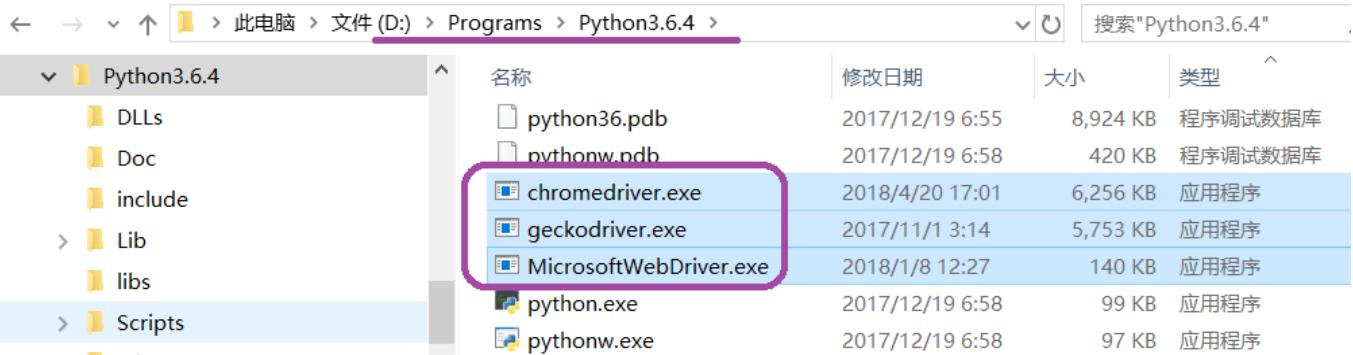
添加国内的资源库地址：<https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/free/>



3.23.3 Selenium 编程

3.23.3.1 配置 Python 环境

- 1) 确保 selenium 包已添加项目的虚拟环境中
- 2) 下载所需的 WebDriver，并复制到在系统 PATH 中的任何目录下，例如把它们放在 python 解释器目录下：



3.23.3.2 示例代码

以下示例：打开 firefox、访问网址、填写账号、点击按钮登录、查找网页菜单项、点击菜单项、退出：

```

1 test.py x
2
3 from selenium import webdriver
4 from time import sleep
5 from selenium.webdriver.common.action_chains import ActionChains
6 import common_values
7
8
9 def click_sub_menu(the_browser, main_item, sub_item):
10     main_menu = the_browser.find_element_by_id("topMenuForm:" + main_item)
11     ActionChains(the_browser).move_to_element(main_menu).perform()
12     sleep(common_values.PAGE_WAIT_SHOW_SECONDS)
13
14     sub_menu = the_browser.find_element_by_id("topMenuForm:" + sub_item)
15     ActionChains(the_browser).move_to_element(sub_menu).perform()
16     sleep(common_values.PAGE_WAIT_SHOW_SECONDS)
17     sub_menu.click()
18     sleep(common_values.PAGE_WAIT_SHOW_SECONDS)
19
20 def logout(the_browser):
21     click_sub_menu(the_browser, "AccountMenu", "LogoutMenu")
22
23 if __name__ == '__main__':
24     browser = webdriver.Firefox()
25     browser.get("http://localhost:8080/QXSKweb/faces/login.xhtml")
26     browser.implicitly_wait(10)
27     browser.maximize_window()
28
29     browser.find_element_by_id("logForm:user").send_keys("测试-普通用户")
30     browser.find_element_by_id("logForm:Pass").send_keys("TesterCommon")
31     browser.find_element_by_id("logForm:loginButton").click()
32
33     click_sub_menu(browser, "UserMenu", "UserSubMenu")
34
35     logout(browser)
36
37
38
39
40
41

```

导入必要的包

查找特定的主菜单项

鼠标移到目标主菜单项上

查找特定子菜单项

鼠标移到目标子菜单项上

点击目标子菜单项

打开Firefox浏览器

访问网址

等待页面加载，最多等待10秒

填写账户信息

点击登录按钮

点击指定主菜单项下指定子菜单项

注销

3.23.3.3 常用 selenium 代码

以下是一些常用的 selenium 代码：

```
# 打开 firefox 浏览器
```

```
browser = webdriver.Firefox()
```

```
# 访问地址
```

```
browser.get(loginUrl)
```

```
# 设置浏览器窗口
```

```
browser.maximize_window()
```

```
browser.set_window_size(1000, 800)
```

```
browser.set_window_position(22, 33)
```

```
# 操作页面
```

```
browser.implicitly_wait(2)
```

```
time.sleep(5)
```

```
browser.back()
```

```
browser.forward()
```

```
# 定位元素
```

```
browser.find_element_by_id('kw1').send_keys("selenium")
```

```
browser.find_element_by_xpath("//input[@id='kw1']").send_keys("selenium")
```

```
browser.find_element_by_link_text("贴 吧").click()
```

```
browser.find_element_by_partial_link_text("贴").click()
```

```
browser.find_element_by_css_selector("#kw1").send_keys("selenium")
```

```
browser.find_element_by_class_name("btn").click()
```

```
# 操作元素
```

```
browser.find_element_by_id("kw1").clear()
```

```
browser.find_element_by_id('su1').click()
```

```
browser.find_element_by_id('su1').submit()
```

```
browser.find_element_by_xpath("//input[@id='kw1']").send_keys("selenium")
```

```
browser.find_element_by_id('kw1').send_keys(Keys.ENTER)
```

```
browser.find_element_by_id('kw1').send_keys(Keys.CONTROL, 'a')
```

```
pswd = driver.find_element_by_id("password")
```

```
# 鼠标双击事件
```

```
doubleclick(element)
```

```
# 右键单击鼠标
rightclick = ActionChains(driver)
rightclick.context_click(pswd).perform()

# 鼠标拖放事件
drag_and_drop(element, target) // element ,target 分别为原位置和目标位置
```

查看 用户反馈	
数据编号	456
用户反馈的对象	网页客户端
用户反馈类型	改进建议
标题	页面元素应定义id，以便于自动化功能测试
描述	自动化功能测试需要id来定位页面元素
报告者	任珊虹
报告时间	2018-01-08 15:10:43
响应说明	部分实现
响应时间	2018-02-27 09:57:35
响应者	任珊虹
用户反馈状态	正在处理
用户反馈的来源	网页客户端
解决的版本	-

3.23.3.4 selenium 三种等待方式

网友的描述：

<https://blog.csdn.net/ping523/article/details/53419622>

<https://www.cnblogs.com/imyalost/p/7420924.html>

4 安卓应用的开发

4.1 Android Studio

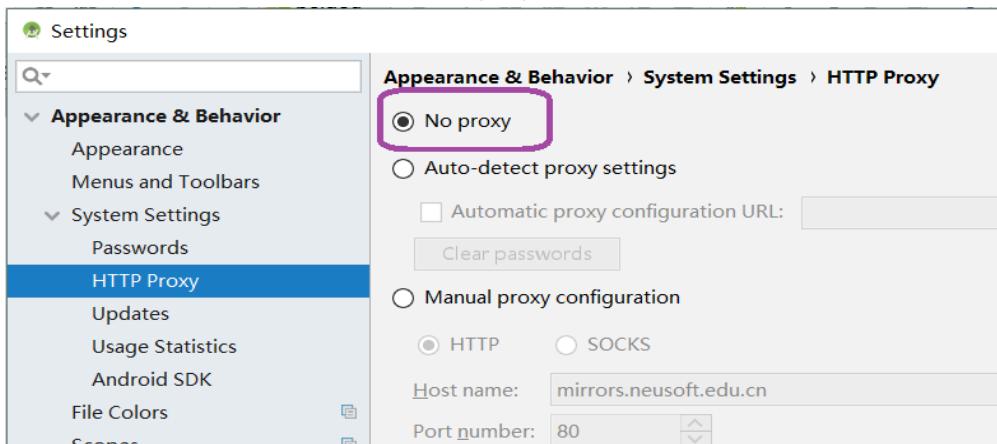
4.1.1 资源地址

下载地址：

<https://developer.android.google.cn/studio/index.html#win-bundle>

4.1.2 HTTP proxy

由于无法从谷歌官方网址更新 AS，我曾经按照网上的方法设置代理，代理曾经能帮助更新 SDK 但不能帮助更新 Studio，所以我的 AS 一直版本较低。然而最近几个月，似乎情况有了变化：代理反而会带来麻烦，当设置为 No Proxy，SDK 和 Studio 竟然可以自动更新了：(google.cn 真的可以访问了)



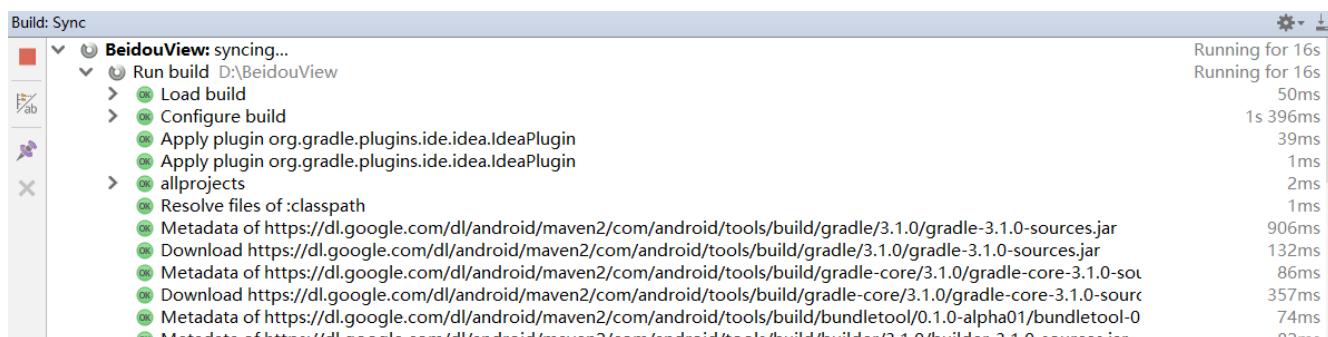
4.1.3 更新 Gradle

“AndroidStudio、gradle、buildToolsVersion 关系”：

<https://blog.csdn.net/lixin88/article/details/61196274>

由于不能从谷歌资源站自动下载，更新 Gradle 曾经是一件痛苦的事。现在 AS 的资源更新已经很智能了：按照它的提示点击链接、或者修改配置就好了。

以下是我的 AS 环境的截图：它正在从谷歌资源站下载东西。。。 (我的网络实际不能访问谷歌资源)



4.2 应用的全局数据

4.2.1 应用的常量

类 com.qxsk9.beidouview.object.CommonValues 包含了所有的系统常量：

- 1) 在代码构建时就确定了值。
- 2) 不能被修改。

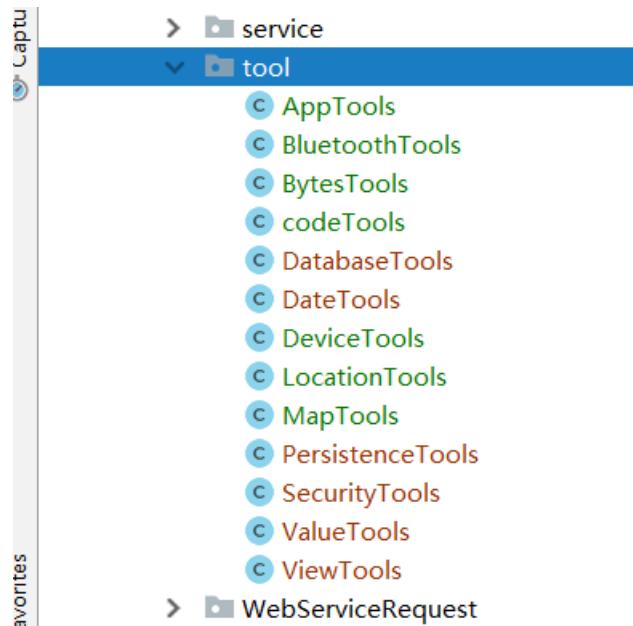
4.2.2 应用的参数

类 com.qxsk9.system.SystemValues 包含了所有的系统参数：

- 1) 对应于数据库表 TableAppAttribute 的数据。
- 2) 由 LoginActivity 在服务器初始化时读入内存。
- 3) 由用户在界面上操作时更改。
- 4) 代码保证数据库表中的数据与内存中的数据始终保持一致。

4.2.3 应用的工具箱

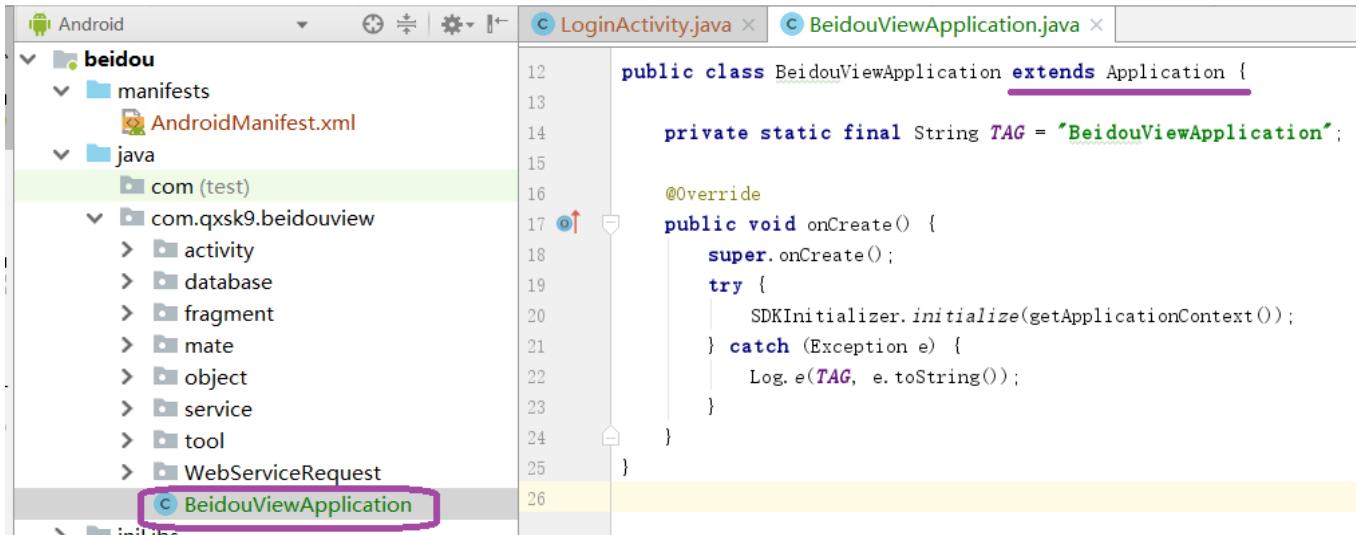
系统中通用的静态方法按用途定义在包 com.qxsk9.beidouview.tool 的不同工具类中：



4.3 应用的初始化

4.3.1 Application 类

类 com.qxsk9.beidouview.BeidouViewApplication 继承了 Application，调用了百度地图的初始化方法：



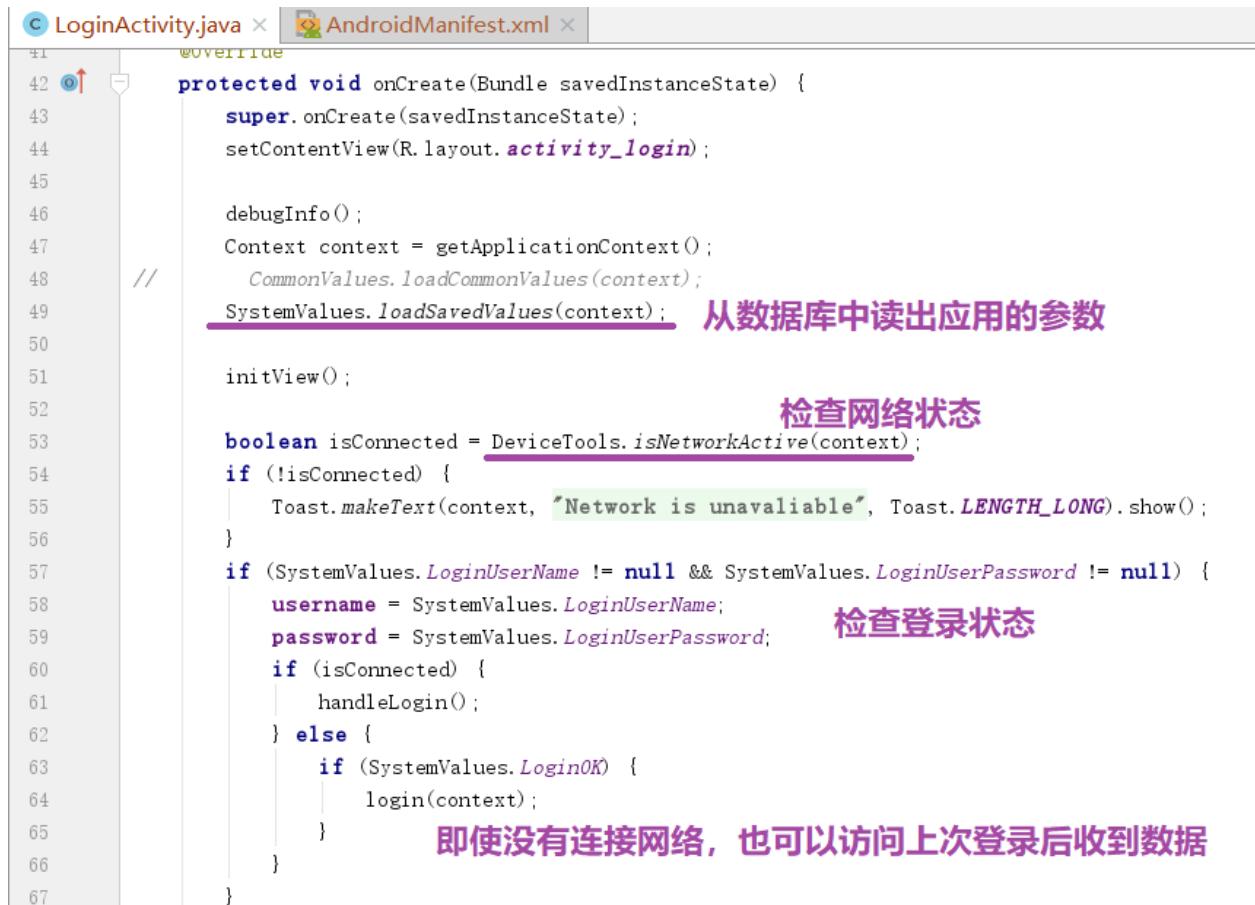
```

public class BeidouViewApplication extends Application {
    private static final String TAG = "BeidouViewApplication";
    @Override
    public void onCreate() {
        super.onCreate();
        try {
            SDKInitializer.initialize(getApplicationContext());
        } catch (Exception e) {
            Log.e(TAG, e.toString());
        }
    }
}

```

4.3.2 应用的封面

类 com.qxsk9.beidouview.activity.LoginActivity 是应用的启动界面：



```

protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_login);

    debugInfo();
    Context context = getApplicationContext();
    CommonValues.loadCommonValues(context);
    SystemValues.loadSavedValues(context); 从数据库中读出应用的参数

    initView();

    boolean isConnected = DeviceTools.isNetworkActive(context);
    if (!isConnected) {
        Toast.makeText(context, "Network is unavailable", Toast.LENGTH_LONG).show();
    }
    if (SystemValues.LoginUserName != null && SystemValues.LoginUserPassword != null) {
        username = SystemValues.LoginUserName;
        password = SystemValues.LoginUserPassword;
        if (isConnected) {
            handleLogin();
        } else {
            if (SystemValues.LoginOK) {
                login(context);
            }
        }
    }
}

```

4.4 应用百度地图

4.4.1 申请 Android 端的 AK

百度地图 AK 管理地址:

<http://lbsyun.baidu.com/apiconsole/key>

按以下网页介绍的方式找到开发环境的 SHA1 :

<http://lbsyun.baidu.com/index.php?title=androidsdk/guide/key>

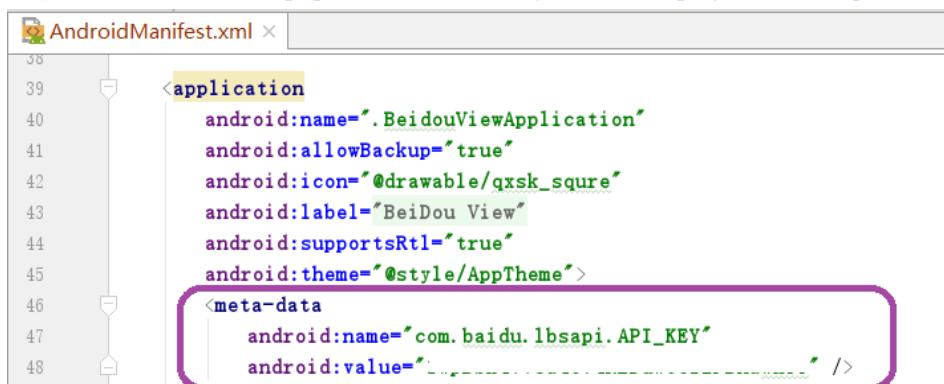
填写包名和 SHA 值以申请 Android 端的 AK:

应用AK:
 应用名称: 北斗视野安卓版
 应用类型: Android端
 启用服务: 云检索 正逆地理编码 Android地图SDK
 Android定位SDK Android导航离线SDK Android导航SDK
 静态图 全景静态图 坐标转换
 鹰眼轨迹 全景URL API Android导航 HUD SDK
 云逆地理编码 云地理编码 推荐上车点
 Javascript API 地点检索 普通IP定位
 天气查询API 路线规划
 *发布版SHA1:
 开发版SHA1:
 *包名: com.qxsk9.beidouview
 安全码:
 eidouview

 eidouview
 Android SDK安全码组成: SHA1+包名。[\(查看详细配置方法\)](#) 新申请的Mobile与Browser类型的ak不再支持云存储接口的访问, 如要使用云存储, 请申请Server类型ak。

4.4.2 在 AndroidManifest 中添加开发密钥

<http://lbsyun.baidu.com/index.php?title=androidsdk/guide/create-project/hellomap>



4.4.3 百度地图的 Android SDK

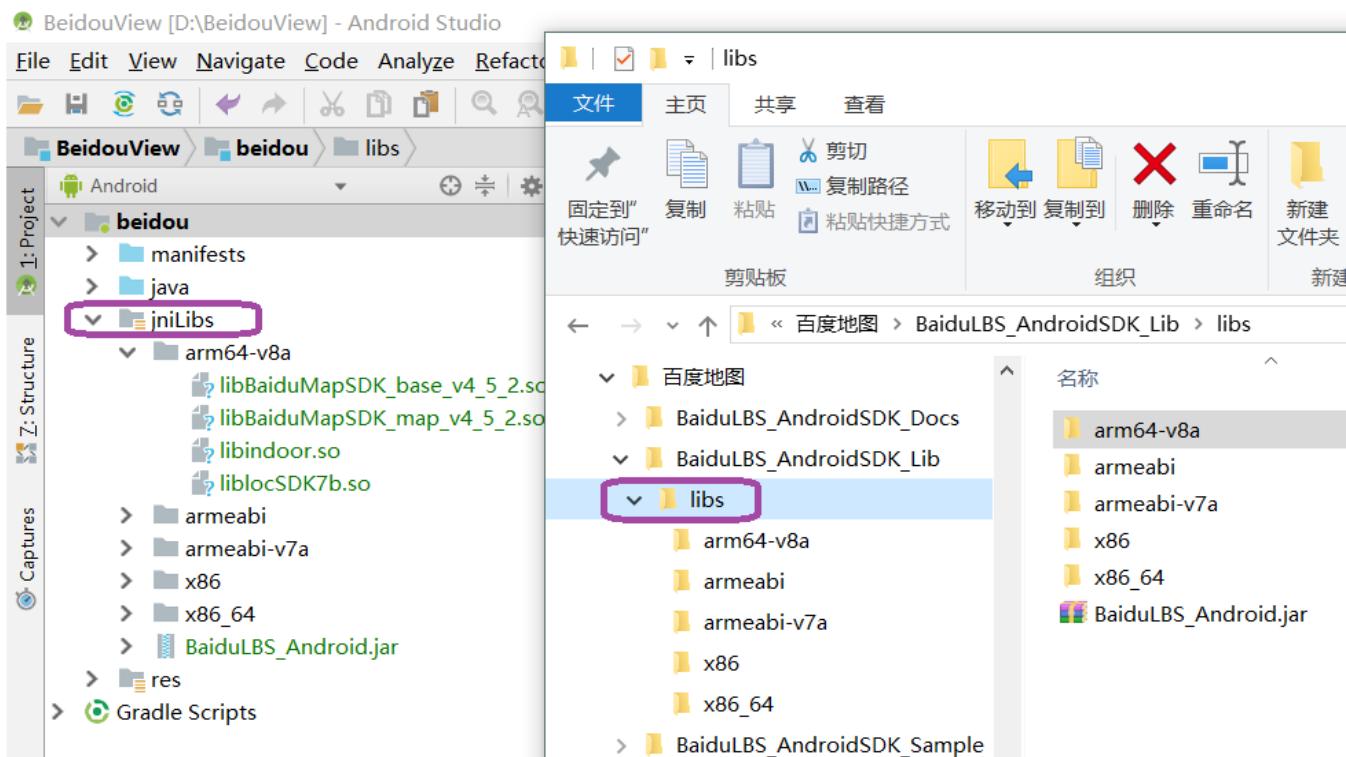
百度地图 Android SDK 的资源地址:

<http://lbsyun.baidu.com/index.php?title=androidsdk>

4.4.4 导入 SDK

将下载下来的 SDK 包中 lib 下的所有文件复制到项目的目录 jniLibs 下:

<http://lbsyun.baidu.com/index.php?title=androidsdk/guide/create-project/androidstudio>



4.4.5 初始化 SDK

在系统的初始化方法中调用 SDK 即可:

```
c BeidouViewApplication.java x
12  public class BeidouViewApplication extends Application {
13
14      private static final String TAG = "BeidouViewApplication";
15
16      @Override
17      public void onCreate() {
18          super.onCreate();
19          try {
20              SDKInitializer.initialize(getApplicationContext());
21          } catch (Exception e) {
22              Log.e(TAG, e.toString());
23          }
24      }
25 }
```

4.4.6 在布局中添加地图

在布局中加入类 com.baidu.mapapi.map.MapView 即可。同一 FrameLayout 下元素将被显示在地图上，用这种方法可以在地图上显示字串或者控件：

```

<FrameLayout
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="1">

    <com.baidu.mapapi.map.MapView
        android:id="@+id/location_map_view"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:clickable="true" />

    <LinearLayout...>

    <TextView
        android:id="@+id/location_load_time"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="bottom|right"
        android:textColor="@color/darkBlue" />

```

4.4.7 在代码中控制地图

以下是引用和操作地图的示例：

```

private void initView() {
    mMapView = (MapView) fragmentView.findViewById(R.id.location_map_view);
    mMapView.showZoomControls(b: true); //设置启用内置的缩放控件
    mMapView.removeViewAt(index: 1); //隐藏地图上百度地图logo图标
    mMapView.showScaleControl(b: true); //隐藏地图上比例尺
    mBaiduMap = mMapView.getMap();
    mUiSettings = mBaiduMap.getUiSettings();
    mUiSettings.setCompassEnabled(true); //设置是否允许指南针
    mUiSettings.setScrollGesturesEnabled(true); //设置是否允许拖拽手势
    mUiSettings.setZoomGesturesEnabled(true); //设置是否允许缩放手势

    mBaiduMap.setMapStatus(MapStatusUpdateFactory.zoomTo(v: 15));
    mBaiduMap.setOnMarkerClickListener(marker) -> { return popInfo(marker); });
    mBaiduMap.setOnMapClickListener(new BaiduMap.OnMapClickListener() {
        @Override
        public boolean onMapPoiClick(MapPoi arg0) { return false; }

        @Override
        public void onMapClick(LatLng arg0) { mBaiduMap.hideInfoWindow(); }
    });
}

```

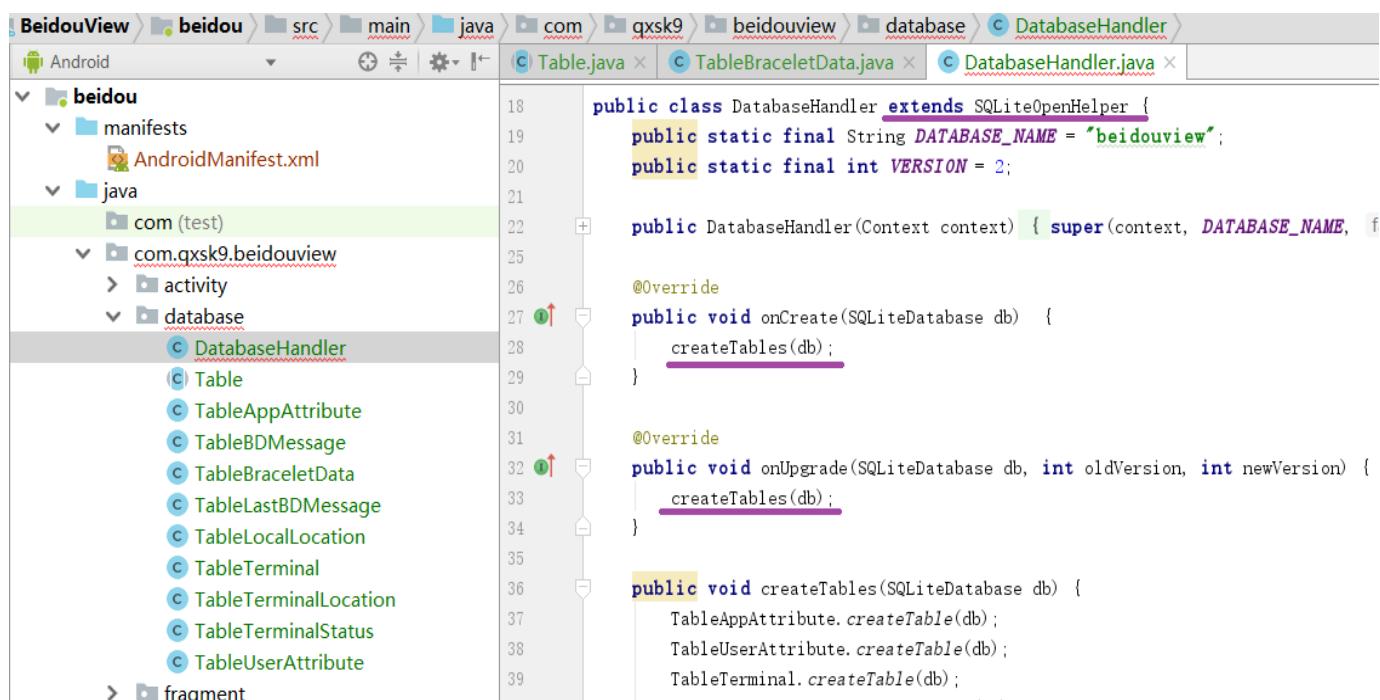
4.5 SQLite3

本地数据都存放在 SQLite3 数据库中，数据库相关的类都在包 com.qxsk9.beidouview.database 中。

查看 用户反馈	
数据编号	430
用户反馈的对象	安卓客户端
用户反馈类型	功能需求
标题	本地数据全部存入Sqlite
描述	现在本地数据存为SharedPreferences。而且没有保存查询数据的快照。
报告者	任珊瑚
报告时间	2017-11-23 10:19:00
响应说明	本地数据包括：app全局变量、用户变量、查询数据的快照。
响应时间	2017-12-22 16:43:43
响应者	任珊瑚
用户反馈状态	已解决
用户反馈的来源	网页客户端
解决的版本	安卓客户端-2.000

4.5.1 数据库辅助类 DatabaseHandler

类 DatabaseHandler 继承 SQLiteOpenHelper，用于对数据库的操作：



The screenshot shows the Android Studio interface with the following details:

- Project Structure:** The project is named "BeidouView". Inside the "beidou" package, there is a "database" folder containing several Java files: DatabaseHandler.java, Table.java, TableBraceletData.java, and others.
- Code Editor:** The code for DatabaseHandler.java is displayed. It extends SQLiteOpenHelper and defines constants for the database name ("beidouview") and version (2). It overrides the onCreate and onUpgrade methods to create tables if they don't exist. The createTables method calls three other static methods: TableAppAttribute.createTable, TableUserAttribute.createTable, and TableTerminal.createTable.
- Code View:** The code is annotated with line numbers (18-39) and some lines are highlighted in purple, likely indicating code snippets or specific sections of interest.

```

public class DatabaseHandler extends SQLiteOpenHelper {
    public static final String DATABASE_NAME = "beidouview";
    public static final int VERSION = 2;

    public DatabaseHandler(Context context) { super(context, DATABASE_NAME, null, VERSION); }

    @Override
    public void onCreate(SQLiteDatabase db) {
        createTables(db);
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        createTables(db);
    }

    public void createTables(SQLiteDatabase db) {
        TableAppAttribute.createTable(db);
        TableUserAttribute.createTable(db);
        TableTerminal.createTable(db);
    }
}

```

4.5.2 数据表的模板 Table

抽象类 Table 定义了数据表的模板和通用的数据处理方法：

```
Table.java x

22 ① public abstract class Table {
23
24      protected static final String TAG = "Table";
25
26 ②     protected abstract String getTableName();
27
28 ③     protected abstract String[] getTableColumns();
29
30 ④     protected abstract String[] getTableColumnsTypes();
31
32 ⑤     protected abstract String[] getTableColumnsNotNulls();
33
34 ⑥     protected abstract String getPrimaryKey();
35
36 ⑦     protected abstract String[] getIndices();
37
38 ⑧     protected abstract String getTag();

39
40
41     public static void dropTable(SQLiteDatabase db, String table) {
42         try {
43             db.execSQL("DROP TABLE IF EXISTS " + table);
44         } catch (Exception e) {
45
46         }
47     }
48
49     @ public static void createTable(SQLiteDatabase db, Table table) {
50         createTable(db, table.getTableName(),
51                     table.getTableColumns(), table.getTableColumnsTypes(),
52                     table.getTableColumnsNotNulls(), table.getPrimaryKey(), table.getIndices());
53     }


```

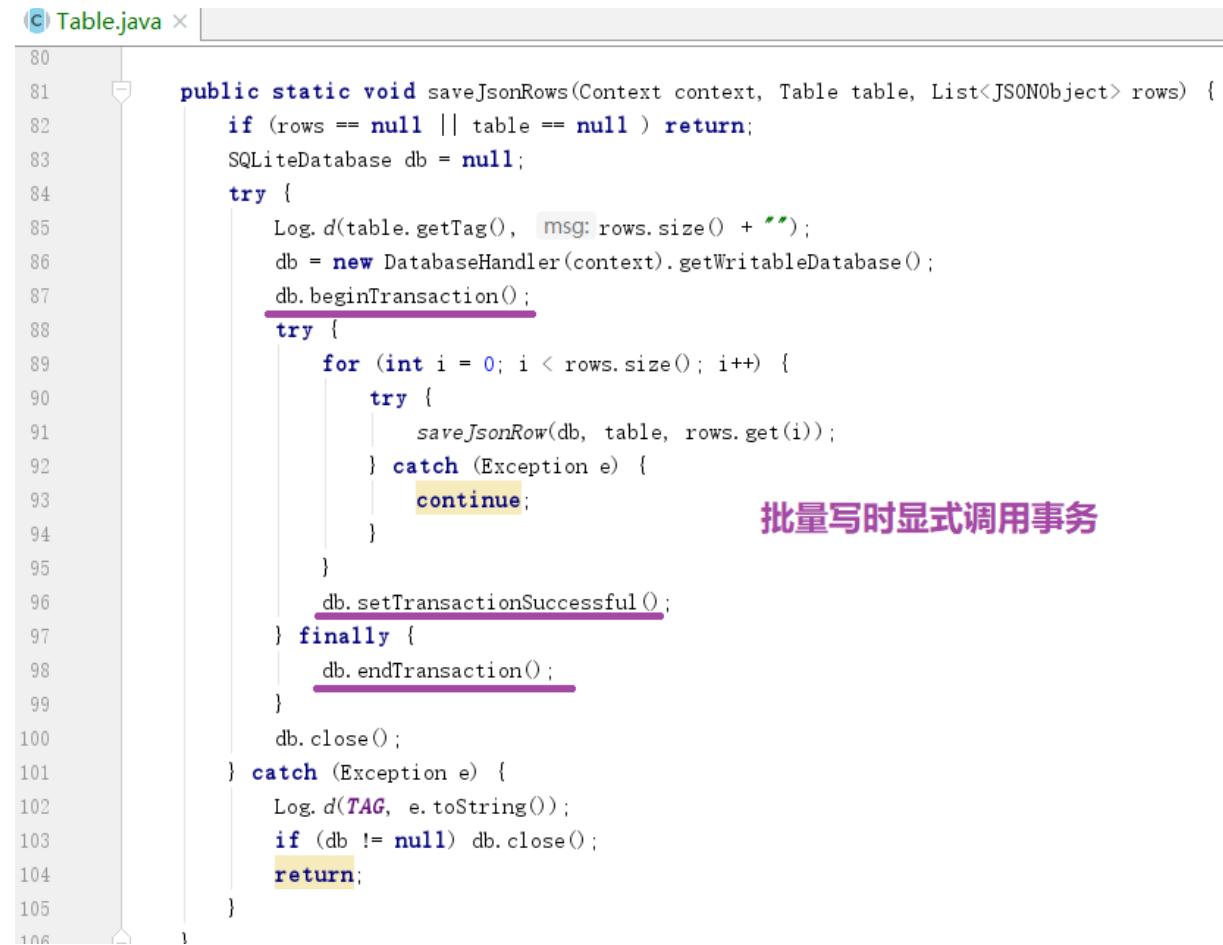
4.5.3 数据表

数据表继承类 Table，具体定义表的属性和特定的数据处理方法，以下是一个数据表的示例：

```
18 public class TableLocalLocation extends Table {
19
20     public static final String NAME = "TableLocalLocation";
21     public static final String[] COLUMNS = {"id",
22         "x", "y", "z",
23         "speed", "direction", "record_time", "accuracy", "provider", "coordinate"};
24     public static final String[] COLUMN_TYPES = {"INTEGER",
25         "DOUBLE", "DOUBLE", "DOUBLE",
26         "DOUBLE", "DOUBLE", "DATETIME", "DOUBLE", "TEXT", "TEXT"}];
27     public static final String[] COLUMNS_NOT_NULLS = {"NOT NULL",
28         "NOT NULL", "", "",
29         "", "", "", "", ""};
30     public static final String PRIMARY_KEY = "id";
31     public static final String[] INDICES = {"record_time"};
32
33     private static final String TAG = "TableLocalLocation";
34
35     protected String getTableName() { return NAME; }
```

4.5.4 显式调用事务

不写事务语句，则每条写操作语句都是提交一次事务。在批量写数据时若不显式调用事务，则性能影响很大。



```

80
81     public static void saveJsonRows(Context context, Table table, List<JSONObject> rows) {
82         if (rows == null || table == null) return;
83         SQLiteDatabase db = null;
84         try {
85             Log.d(table.getTag(), "msg: " + rows.size());
86             db = new DatabaseHandler(context).getWritableDatabase();
87             db.beginTransaction();
88             try {
89                 for (int i = 0; i < rows.size(); i++) {
90                     try {
91                         saveJsonRow(db, table, rows.get(i));
92                     } catch (Exception e) {
93                         continue;
94                     }
95                 }
96                 db.setTransactionSuccessful();
97             } finally {
98                 db.endTransaction();
99             }
100            db.close();
101        } catch (Exception e) {
102            Log.d(TAG, e.toString());
103            if (db != null) db.close();
104        }
105    }
106

```

批量写时显式调用事务

查看 用户反馈

数据编号	471
用户反馈的对象	安卓客户端
用户反馈类型	性能优化
标题	sqlite批量写应该显式应用事务
描述	不写事务语句，则每条写操作语句都是提交一次事务。
报告者	任珊虹
报告时间	2018-02-28 12:42:42
响应说明	不写事务语句，则3134条数据消耗9144毫秒，平均每条消耗2.92毫秒，还造成4次死锁。而写了事务语句，则3134条数据消耗360毫秒，平均每条消耗0.115毫秒，没有死锁。即，性能相差25倍多！9秒多的操作必须开启后台进程，而不到400毫秒的操作则可以不考虑界面阻塞了。
响应时间	2018-04-22 14:27:37
响应者	任珊虹
用户反馈状态	已解决
用户反馈的来源	网页客户端
解决的版本	安卓客户端-3.000

4.5.5 写数据库的时机

按照谷歌官方文档，最好在 onStop 中写数据库：



```

9
0     @Override
1         public void onStop() {
2             super.onStop();
3
4             if (serverOK) {
5                 Table.clearData(getContext(), terminalTable);
6                 Table.saveJsonRows(getContext(), terminalTable, terminalAttributes);
7             }
}

```

界面切换时在数据库中保存数据

查看 用户反馈

数据编号	473
用户反馈的对象	安卓客户端
用户反馈类型	性能优化
标题	评估数据保存的时机
描述	应避免频繁保存数据、同时避免界面切换的阻塞问题。
报告者	任珊虹
报告时间	2018-02-28 12:52:53
响应说明	经过测试，性能优化前1万条数据的写大约耗时30秒，而性能优化后1万条数据的写大约耗时1秒。目前方案是：在onStop里保存数据。
响应时间	2018-03-06 15:28:24
响应者	任珊虹
用户反馈状态	已解决
用户反馈的来源	网页客户端
解决的版本	安卓客户端-3.000

4.6 调用网络服务

本应用与服务平台的数据交换方式是网络服务和 WebSocket，网络服务用于一次性数据调用，WebSocket 用于接收实时数据。

所有网络服务相关的类都在包 com.qxsk9.beidouview.WebServiceRequest 中。

4.6.1 通用的网络服务对象 WebServiceRequestObject

类 WebServiceRequestObject 定义了网络服务的属性和通用方法：

The screenshot shows the Android Studio interface with the project structure on the left and the code editor on the right.

Project Structure:

- beidou
 - manifests
 - java
 - com (test)
 - com.qxsk9.beidouview
 - activity
 - database
 - fragment
 - mate
 - object
 - service
 - tool
 - WebServiceRequest
 - WebServiceRequestForFence
 - WebServiceRequestForLocation
 - WebServiceRequestForMessages
 - WebServiceRequestForStatus
 - WebServiceRequestForSystem
 - WebServiceRequestForTerminals
 - WebServiceRequestForUser
 - WebServiceRequestObject
 - BeidouViewApplication
 - jniLibs
 - res
 - Gradle Scripts

Code Editor (WebServiceRequestObject.java):

```

private String theURL;
private HashMap<String, String> theParameters = new HashMap();
private String finalURL = null;
private int returnType;
private HttpURLConnection httpURLConnection;
private Object returnData = null;
private boolean called = false;
private int resultCode;
private boolean success = false;
private String errorString = "";
public Object requestData() {
    if (called) return returnData;
    try {
        switch (returnType) {
            case WebServicReturnType.INT:
                returnData = requestIntData();
                break;
            case WebServicReturnType.BOOLEAN:
                returnData = requestBooleanData();
                break;
            case WebServicReturnType.STRING:
                returnData = requestStringData();
                break;
            case WebServicReturnType.JSONARRAY:
                returnData = requestJSONArrayData();
                break;
            case WebServicReturnType.JSON:
                returnData = requestjsonData();
                break;
            case WebServicReturnType.XML:
                returnData = requestXMLData();
                break;
            case WebServicReturnType.DOUBLE:
                returnData = requestDoubleData();
                break;
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
    return returnData;
}

```

Annotations:

- 网络服务的属性**: Labels the private fields: theURL, theParameters, finalURL, returnType, httpURLConnection, returnData, called, resultCode, success, errorString.
- 网络服务的通用方法**: Labels the requestData() method.

4.6.2 http 请求

本系统利用 okhttp3 处理 http 请求：

```

private String okHttp3Request(String path, String contentType) {
    if (path == null || path.length() == 0 ||
        contentType == null || contentType.length() == 0)
        return null;
    resultCode = -1;
    try {
        OkHttpClient okHttpClient = new OkHttpClient.Builder()
            .connectTimeout(SystemValues.SettingsConnectTimeout, TimeUnit.SECONDS)
            .readTimeout(SystemValues.SettingsReadTimeout, TimeUnit.SECONDS)
            .writeTimeout(SystemValues.SettingsWriteTimeout, TimeUnit.SECONDS)
            .retryOnConnectionFailure(false)
            .build();
        Request request = new Request.Builder()
            .url(path)
            .get()
            .addHeader(name: "Content-Type", contentType)
            .addHeader(name: "Accept", contentType)
            .addHeader(name: "Accept-Charset", value: "UTF-8")
            .addHeader(name: "Charset", value: "UTF-8")
            .build();
        Call call = okHttpClient.newCall(request);
        Response response = call.execute();
        success = response.isSuccessful();
        resultCode = response.code();
        String data = response.body().string();
        return data;
    } catch (SocketTimeoutException e) {
        Log.e(tag: "h", e.toString());
        errorString = e.getLocalizedMessage();
        success = false;
        resultCode = 404;
        return null;
    } catch (IOException e) {
        Log.e(tag: "h", e.toString());
    }
}

```

4.6.3 发送网络服务请求

以下是一个具体的网络服务请求的示例：

```

281
282     protected class RequestNewMessagesNumberTask extends AsyncTask<Void, Integer, Boolean> {
283
284         @Override
285         protected void onPreExecute() {...}
286
287         @Override
288         protected Boolean doInBackground(Void... params) {
289             try {
290                 wsCall = null;
291                 if (DeviceTools.isNetworkActive(getContext()))
292                     wsCall = WebServiceRequestForMessages.requestNewMessagesNumber(new WebServiceRequestObject(),
293                         ValueTools.ListToString(SystemValues.SelectedTerminalNames), TAG);
294                 serverOK = (wsCall != null && wsCall.isSuccess());
295                 if (serverOK) {
296                     if (wsCall.getReturnData() != null)
297                         newMessagesNumber = (int) wsCall.getReturnData();
298                 }
299             } catch (Exception e) {
300                 e.printStackTrace();
301             }
302         }
303     }
304 }

```

```

125     @
126     public static WebServiceRequestObject requestNewMessagesNumber(WebServiceRequestObject wsCall,
127                                                                     String terminals, String fromPage) {
128
129         if (SystemValues.LoginUserName == null || SystemValues.LoginUserPassword == null)
130             return null;
131         if (wsCall == null || terminals == null || terminals.trim().isEmpty())
132             return null;
133         wsCall.setTheURL(CommonValues.WebServiceContext + "u/m/n");
134         HashMap<String, String> theParameters = new HashMap();
135         theParameters.put("ct", CommonValues.ClientType);
136         theParameters.put("ci", SystemValues.DeviceID);
137         theParameters.put("u", SystemValues.LoginUserName);
138         theParameters.put("ap", SystemValues.LoginUserPassword);
139         theParameters.put("t", terminals);
140         theParameters.put("op", fromPage);
141         wsCall.setParameters(theParameters);
142         wsCall.setReturnType(WebServiceRequestObject.WebServiceReturnType.INT);
143         wsCall.requestData();           调用网络服务
144
145     }

```

4.7 调用 WebSocket

4.7.1 继承 WebSocketListener

以下是具体实现 WebSocketListener 的一个示例：

```
private class TerminalMessageListener extends WebSocketListener {

    @Override
    public void onOpen(WebSocket webSocket, Response response) {
        JSONObject clientAttributes = new JSONObject();
        try {
            clientAttributes.put("name": "SessionLanguage", "value": "zh_CN");
            clientAttributes.put("name": "MonitoringTerminals", ValueTools.ListToString(SystemValues.SelectedTerminalNames));
            webSocket.send(clientAttributes.toString()); WebSocket打开后立即向服务平台发送数据参数
            messageWebSocket = webSocket;
        } catch (JSONException e) {
            Log.d("TerminalMessageListener", e.toString());
            webSocket.close(code: 1003, reason: "初始化参数错误");
            messageWebSocket = null;
        }
    }

    @Override
    public void onMessage(WebSocket webSocket, String text) {
        try {
            if (!"\n".equals(text)) return;
            newMessagesNumber += 1;
            if (getActivity() == null) return;
            getActivity().runOnUiThread(() -> {
                if (newMessagesNumber == 0) { 只能在主线程中更新界面
                    messagesNumberShow.setVisibility(View.INVISIBLE);
                } else {
                    messagesNumberShow.setVisibility(View.VISIBLE);
                }
            });
        }
    }
}
```

4.7.2 发送 WebSocket 请求

以下是发送 WebSocket 请求的一个示例：

```
if (messageWebSocket == null) {
    try {
        Request request = new Request.Builder().url(WebSocketRequest.MessageNewNumberSocketPath + "&op=" + TAG).build();
        OkHttpClient webSocketClient = new OkHttpClient.Builder()
            .retryOnConnectionFailure(true)
            .connectTimeout(SystemValues.SettingsConnectTimeout, TimeUnit.SECONDS)
            .readTimeout(SystemValues.SettingsReadTimeout, TimeUnit.SECONDS)
            .writeTimeout(SystemValues.SettingsWriteTimeout, TimeUnit.SECONDS)
            .build();
        webSocketClient.newWebSocket(request, new TerminalMessageListener());
        webSocketClient.dispatcher().executorService().shutdown();
    } catch (Exception e) {
        Log.d(TAG, e.toString());
    }
}
```

4.7.3 发送心跳

当一个 WebSocke 请求成功打开以后，客户端需要定期向服务平台发送心跳：

```

if (successDone) {
    if (messageHeartBeatHandler == null) {
        messageHeartBeatHandler = new Handler();
        messageHeartBeatThread = (Runnable) () -> {
            if (messageWebSocket != null)
                sendMessageSocketHeartBeat();
        };
        messageHeartBeatHandler.postDelayed(this, WebSocketRequest.WebSocketHeartBeatInterval);
    }
    messageHeartBeatHandler.postDelayed(messageHeartBeatThread, WebSocketRequest.WebSocketHeartBeatInterval);
}

requestNewMessagesNumberTaskRunning = false;
}

private void sendMessageSocketHeartBeat() {
    if (messageWebSocket == null) return;
    try {
        messageWebSocket.send(WebSocketRequest.WebSocketHeartBeat);
    } catch (Exception e) {
        e.printStackTrace();
    }
}

```

定期发送心跳

4.7.4 关闭 WebSocket 连接

Activity 必须在 onPause 中关闭已打开 WebSocket 连接：

```

@Override
public void onPause() {
    super.onPause();

    if (requestNewMessagesNumberTask != null || requestNewMessagesNumberTaskRunning)
        requestNewMessagesNumberTask.cancel(mayInterruptIfRunning: true);
    if (requestNewAlertsNumberTask != null || requestNewAlertsNumberTaskRunning)
        requestNewAlertsNumberTask.cancel(mayInterruptIfRunning: true);

    closeMessageWebSocket();
    closeStatusWebSocket();
}

```

取消正在运行的子线程

关闭已打开的WebSocket连接

4.8 定位服务

4.8.1 定位监听器

类 com.qxsk9.beidouview.service.LocationService 提供本地定位服务，支持三种定位监听器：

```

public class LocationService extends Service {
    private static final String TAG = "LocationService";
    public static final int minDistance = 1;

    private LocationManager locationManager = null;
    private LocationListener NetworkLocationListener = null, GpsLocationListener = null;
    private BaiduLocationListener mBaiduLocationListener = null;
    private JSONObject lastLocation = null;
    private Context serviceContext;
    private boolean useNetwork = false, useGPS = false, useBaidu = false;
    private int interval = 5;

    @Override
    public void onCreate() {...}

    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        Log.d(TAG, msg: "onStartCommand");
        if (intent != null) {
            interval = intent.getIntExtra(name: "interval", defaultValue: 5);
            useNetwork = intent.getBooleanExtra(name: "useNetwork", defaultValue: true);
            useGPS = intent.getBooleanExtra(name: "useGPS", defaultValue: true);
            useBaidu = intent.getBooleanExtra(name: "useBaidu", defaultValue: true);
        }
        serviceContext = getApplicationContext();
    }
}

```

4.8.2 新位置数据的处理

当位置监听器收到新位置数据时，服务判断新位置是否更好、保存数据、发出广播：

```

protected class LocalLocationListener implements LocationListener {
    protected String TAG = "LocalLocationListener";

    @Override
    public void onLocationChanged(Location location) {
        Log.d(TAG, msg: "onLocationChanged");
        JSONObject locationData = TableLocallocation.getLocation(location);
        boolean isBetter = LocationTools.isBetterLocation(locationData, lastLocation);
        LocationTools.getLocationInfo(location); // 判断新位置是否更好
        String lastTime = null;
        if (lastLocation != null) {
            SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss.SSSZ");
            lastTime = sdf.format(new Date(lastLocation.getTime()));
        }
        Log.d(TAG, "上次上报时间: " + lastTime);
        if (!isBetter) {
            Log.d(TAG, msg: "not better");
            return;
        }
        lastLocation = locationData; // 在数据库中保存新位置数据
        TableLocalLocation.saveNew(serviceContext, locationData);

        Intent intent = new Intent();
        intent.setAction(CommonValues.Actions.LocationServiceNew);
        sendBroadcast(intent); // 发送广播、通知有新位置数据
        Log.d(TAG, msg: "sendBroadcast:" + CommonValues.Actions.LocationServiceNew);
    }
}

```

4.8.3 算法 isBetterLocation

本应用改写了 Android 官方的算法 isBetterLocation，使其适应于通用地址：

```

private static final int TWO_MINUTES = 1000 * 60 * 2;

/**
 * Determines whether one Location reading is better than the current Location fix
 *
 * @param location The new Location that you want to evaluate
 * @param currentBestLocation The current Location fix, to which you want to compare the new one
 */
public static boolean isBetterLocation(JSONObject location, JSONObject currentBestLocation) {
    try {
        if (currentBestLocation == null) {
            // A new location is always better than no location
            Log.d(tag: 'isBetterLocation', msg: 'currentBestLocation == null');
            return true;
        }

        // Check whether the new location fix is newer or older
        long timeDelta = location.getLong(name: 'record_time') - currentBestLocation.getLong(name: 'record_time');
        boolean isSignificantlyNewer = timeDelta > TWO_MINUTES;
        boolean isSignificantlyOlder = timeDelta < -TWO_MINUTES;
        boolean isNewer = timeDelta > 0;

        // If it's been more than two minutes since the current location, use
        // the new location
        // because the user has likely moved
        if (isSignificantlyNewer) {
            Log.d(tag: 'isBetterLocation', msg: 'isSignificantlyNewer');
            return true;
        } else if (isSignificantlyOlder) {
            Log.d(tag: 'isBetterLocation', msg: 'isSignificantlyOlder');
            return false;
        }
    }
}

```

4.8.4 判断服务是否正在运行

由于 service 不会因为应用停止而停止，所以在显示界面或者处理请求时需要判断 service 是否正在运行：

```

public static boolean isLocationServiceRunning(Context context) {
    return isServiceRunning(context, CommonValues.LocationService);
}

public static boolean isServiceRunning(Context context, String serviceName) {
    boolean isRunning = false;
    ActivityManager activityManager = (ActivityManager) context.getSystemService(Context.ACTIVITY_SERVICE);
    List<ActivityManager.RunningServiceInfo> myList = activityManager.getRunningServices(Integer.MAX_VALUE);
    if (myList.size() <= 0) {
        return false;
    }
    for (int i = 0; i < myList.size(); i++) {
        String mName = myList.get(i).service.getClassName().toString();
        if (mName.equals(serviceName)) {
            isRunning = true;
            break;
        }
    }
    return isRunning;
}

```

4.9 蓝牙服务

4.9.1 service 类

类 com.qxsk9.beidouview.mate.service.MateBluetoothService 定义了蓝牙服务：

The screenshot shows the Android Studio interface with the project tree on the left and the code editor on the right. The project tree shows a package named 'beidou' containing 'manifests', 'java', and 'com (test)'. Under 'java', there are several sub-packages: 'com.qxsk9.beidouview', 'mate', and 'service'. Inside 'service', the file 'MateBluetoothService.java' is selected and highlighted with a purple border. The code editor displays the Java code for the service:

```

public class MateBluetoothService extends Service {
    private static final String TAG = "MateBluetoothService";
    private BluetoothAdapter bluetoothAdapter = null;
    private String deviceName = null;
    private BluetoothSocket beidouMateSocket;
    private InputStream beidouMateInStream;
    private OutputStream beidouMateOutStream;
    private boolean isCanceled = false;
    private BeidouMateConnectThread connectThread = null;
    private DataReceiver dataReceiver = null;
    private Context context = null;

    private static final UUID UUID_SECURE =
        UUID.fromString("00001101-0000-1000-8000-00805F9B34FB"); // 蓝牙串口服务
    private static final UUID UUID_INSECURE =
        UUID.fromString("00001101-0000-1000-8000-00805F9B34FB");
}

```

4.9.2 在子线程中读写蓝牙串口

蓝牙服务被启动后将在一个子线程中循环读取蓝牙串口：

The screenshot shows the code for the 'BeidouMateConnectThread' class. The code is annotated with pink text and arrows pointing to specific methods:

- 在子线程中处理读写串口**: Points to the constructor 'BeidouMateConnectThread()'.
- 读串口**: Points to the method 'reading()'.
- 解析接收到的北斗数据**: Points to the method 'parseReceiveData(byte[] buffer, int size)'.
- 写串口**: Points to the method 'write(byte[] cmd)'.

```

private class BeidouMateConnectThread extends Thread {
    private String TAG = "MateConnectThread";
    private String lastCmd = null;
    private int sleepTime = 300, repeat = 0;

    public BeidouMateConnectThread() { Log.d(TAG, msg: "BeidouMateConnectThread"); }

    public void run() {}

    public void clearBuff() {}

    public void reading() {}

    public void parseReceiveData(byte[] buffer, int size) {}

    public boolean write(byte[] cmd) {}      写串口

    public void cancel() {}
}

```

4.9.3 发送广播

蓝牙服务解析串口接收到的数据，然后以广播的方式发送出去，并以命令类型为数据的标识：

```

public void parseReceiveData(byte[] buffer, int size) {
    case "$BDBSI": // 36,$BDBSI,01,04,0,0,0,0,0,0,0,0,0,0,0,0*5B
        bundle = BDCommandBDBSI.parse(cmd);
        break;
    case "$BDGXM":
        break;
    case "$CFGSY":
        if (!cmd.startsWith("$cfgsys")) break;
        break;
    }
    if (bundle != null && bundle.getString(key: "data_type") != null) {
        Intent intent = new Intent();
        intent.setAction(bundle.getString(key: "data_type"));
        intent.putExtras(bundle);
        sendBroadcast(intent);
        Log.d(TAG, msg: "broadcastData:" + bundle.getString(key: "data_type"));
    }
}

```

```

private static final String TAG = "BDCommandBDBSI";
public static Bundle parse(String cmd) {
    String[] values = cmd.split(regex: ",");
    Bundle bundle = new Bundle();
    bundle.putString("data_type", MateCommonValues.Actions.BeidouMateData_RDSS_STATUS);
    bundle.putInt("responseBeamId", values[1] != null && !"".equals(values[1]) ? Integer.valueOf(values[1]) : 0);
    bundle.putInt("timeLagId", values[2] != null && !"".equals(values[2]) ? Integer.valueOf(values[2]) : 0);
    bundle.putInt("beam1", !"".equals(values[3]) ? Integer.valueOf(values[3]).intValue() : 0);
    bundle.putInt("beam2", !"".equals(values[4]) ? Integer.valueOf(values[4]).intValue() : 0);
    bundle.putInt("beam3", !"".equals(values[5]) ? Integer.valueOf(values[5]).intValue() : 0);
    bundle.putInt("beam4", !"".equals(values[6]) ? Integer.valueOf(values[6]).intValue() : 0);
    bundle.putInt("beam5", !"".equals(values[7]) ? Integer.valueOf(values[7]).intValue() : 0);
    bundle.putInt("beam6", !"".equals(values[8]) ? Integer.valueOf(values[8]).intValue() : 0);
    bundle.putInt("beam7", !"".equals(values[9]) ? Integer.valueOf(values[9]).intValue() : 0);
    bundle.putInt("beam8", !"".equals(values[10]) ? Integer.valueOf(values[10]).intValue() : 0);
    bundle.putInt("beam9", !"".equals(values[11]) ? Integer.valueOf(values[11]).intValue() : 0);
    String beam10 = values[12].split(regex: "\\\\" *)[0];
    bundle.putInt("beam10", !"".equals(beam10) ? Integer.valueOf(beam10).intValue() : 0);
}

```

4.9.4 接收广播

蓝牙服务定义了一个广播接收器，当接收到北斗命令时写入蓝牙串口：



```

434     private class DataReceiver extends BroadcastReceiver {
435
436         @Override
437         public void onReceive(Context context, Intent intent) {
438             if (connectThread == null) {
439                 Log.e(TAG, msg: "DataReceiver: connectThread is null");
440                 stopSelf();
441                 return;
442             }
443             String action = intent.getAction();
444             Log.d(TAG, msg: "DataReceiver:" + action);
445             if (MateCommonValues.Actions.BeidouMateSendCommandRequest.equals(action)) {
446
447                 byte[] cmd = intent.getByteArrayExtra(name: "cmd");
448                 if (cmd == null) {
449                     Log.e(TAG, msg: "Null cmd");
450                     return;
451                 }
452                 try {
453                     String cmdS = new String(cmd, charsetName: "GBK");
454                     Log.d(TAG, msg: "command to be sent: " + cmdS);
455                     if (cmdS.length() < 5) {
456                         Log.e(TAG, msg: "Wrong cmd");
457                         return;
458                     }
459                     connectThread.write(cmd);
460                 } catch (Exception e) {

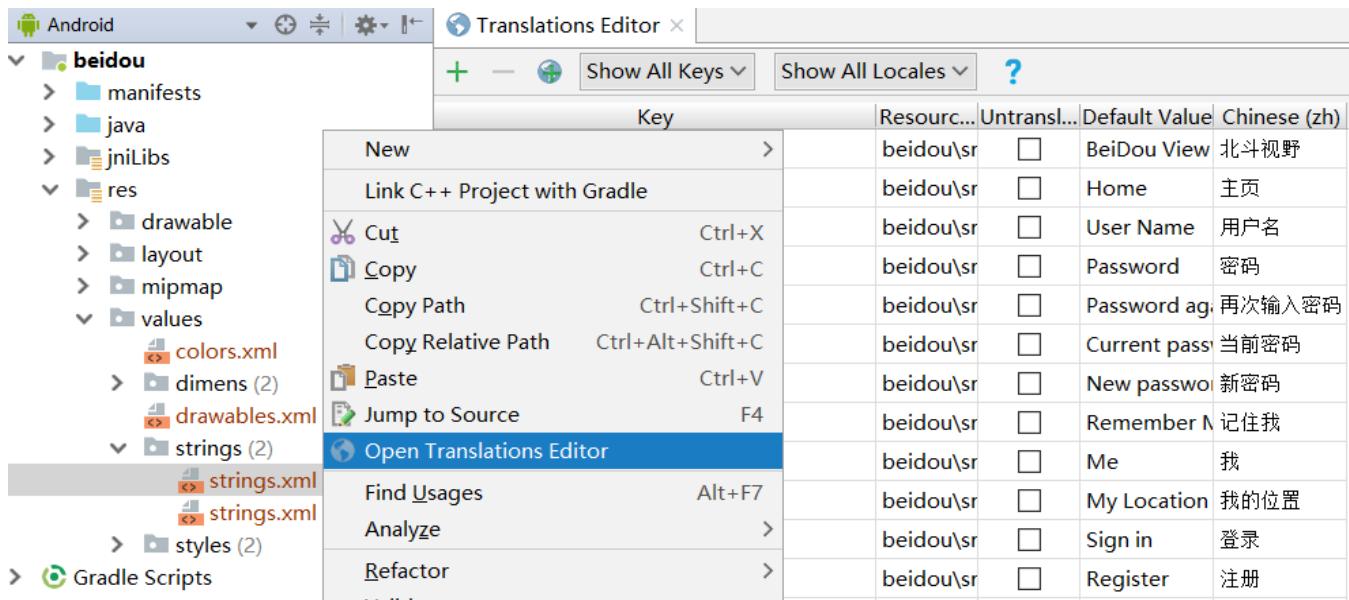
```

接收北斗命令的广播接收器

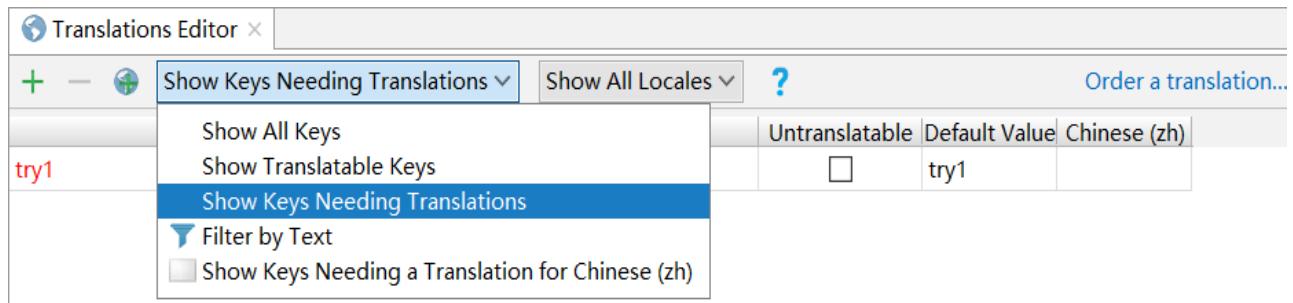
把接收到的北斗命令写入蓝牙串口

4.10 编辑本地化文件

右键 Strings.xml 文件、选择“Open Translations Editor”，打开翻译编辑器，可以方便地翻译多种语言：



可以在过滤器里选择只显示需要翻译的内容：



4.11 编译、构建、打包

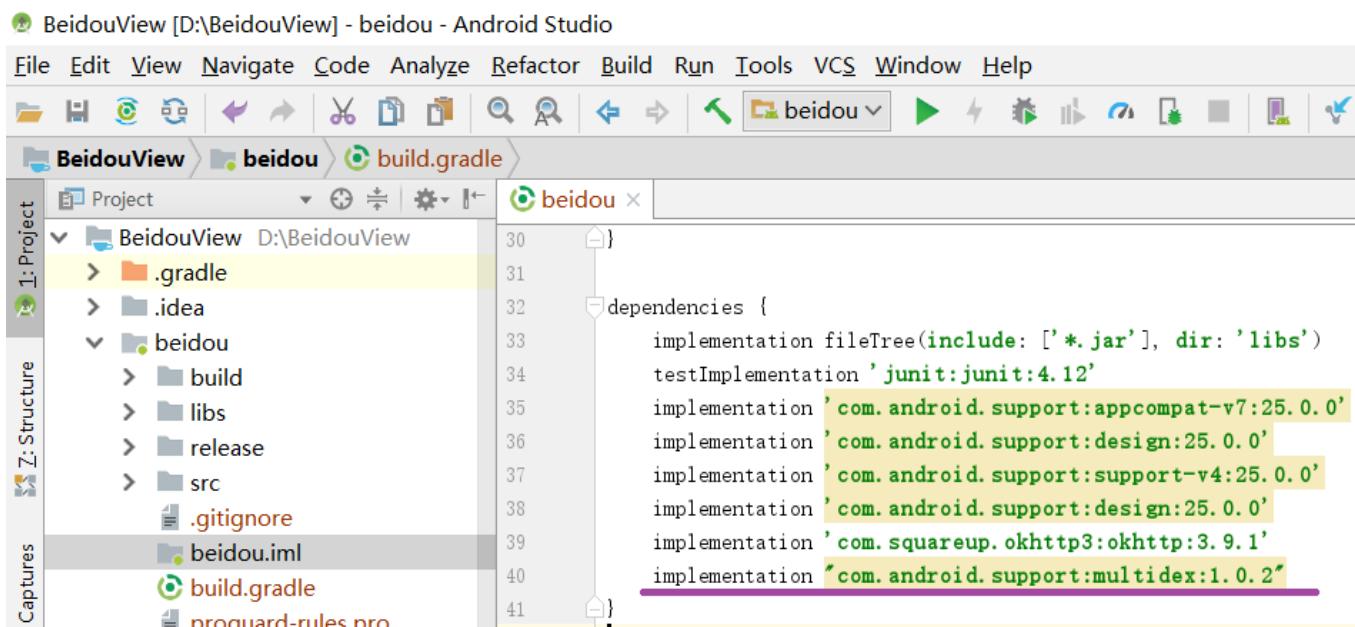
4.11.1 Android 的 65k 问题

“Android 有个限制，每个 App 中函数最多只能有 65536 个”

<http://www.jianshu.com/p/245022d136e1>

由于引入了百度地图的库包，本应用直接碰到 65k 问题。

以下是实践的结论：在模块的 gradle 配置文件中添加依赖。（此法适用于最新版 AS）



4.11.2 “Failed to resolve: com.android.support:multidex:1.0.2”

错误信息：

```

Error:Failed to resolve: com.android.support:multidex:1.0.2
<a href="add.google.maven.repository">Add Google Maven repository and sync project</a><br>
<a href="openFile:D:/BeidouView/beidou/build.gradle">Open File</a><br>
<a href="open.dependency.in.project.structure">Show in Project Structure dialog</a>

```

解决办法：修改项目 build.grade

```

allprojects {
    repositories {
        google()
        jcenter()
        maven {
            url 'https://maven.google.com';
        }
        mavenLocal()
    }
}

```

```

    }
}

```

4.11.3 “is not translated in "en" (English) [MissingTranslation]”

打包 APK 时，错误信息：

Error:(63) Error: "baidutieba" is not translated in "en" (English) [MissingTranslation]

Error:(67) Error: "share_to_baidutieba" is not translated in "en" (English) [MissingTranslation]

解决办法：

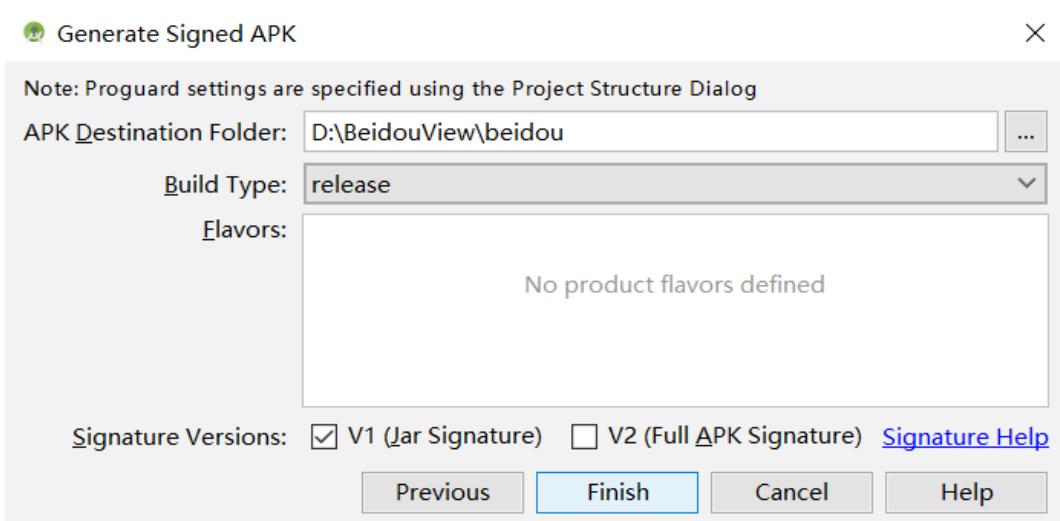
<https://blog.csdn.net/KjunChen/article/details/50043487>

4.11.4 V1(Jar Signature)和 V2(Full APK Signature)

在新版 AS 中签名打包时出现新的选项：

<https://stackoverflow.com/questions/42648499/difference-between-signature-versions-v1jar-signature-and-v2full-apk-signat>

本系统签名打包时直接只选择了 V1：



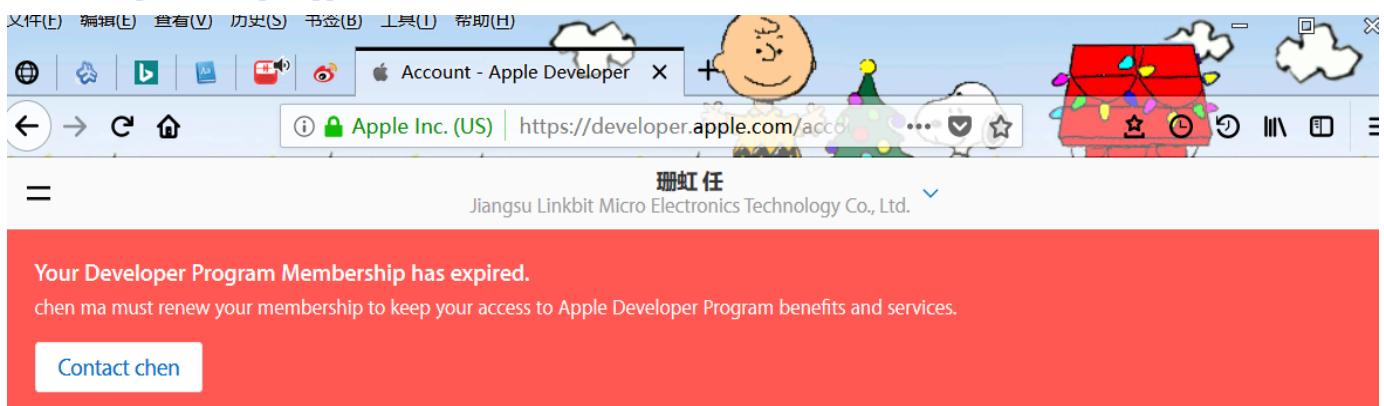
5 iOS 应用的开发

5.1 开发者账号

5.1.1 企业账户还是个人账户？

iOS 应用的发布需要开发者账户。对于企业，最好购买企业账户而不是使用个人账户：

<https://developer.apple.com/account/>



Jiangsu Linkbit Micro Electronics Technology
Co., Ltd.
Apple Developer Program

企业账户如果不及时续费，则会“过期”，相应的应用将会被下架：

iTunes Connect 任珊虹 ✓ Jiangsu Linkbit Micro Electronics Technology Co., Ltd.

开发人员计划会员资格已过期
您的会员资格已过期，在续期会员资格之前，您的 App 将从 App Store 中移除。若要续期会员资格，具有团队代理人职能的用户必须登录 Apple 开发人员网站并在该网站上续期会员资格。如需更多信息，请参阅[常见问题](#)。

新闻

以更轻松的方式联系我们

现在，您可以在“联系我们”页面获取有关开发、分发、会员资格和其他信息的帮助。您只需在“Apple Developer Support”中选取您的主题，然后向我们发送信息或申请电话支持即可。您可以在世界各地获取英文支持，也可以在所选区域获取法文、德文、意大利文、日文、韩文、葡萄牙文、西班牙文和普通话（简体中文电子邮件）的支持。[了解更多](#)

Transporter 的新 IP 地址

自2018年8月21日起，通过 Apple 的命令行交付工具 Transporter 交付的 App 将添加新 IP 地址。如果您在启用防火墙的情况下交付 App 或 App 元数据，请确保防火墙规则允许通过新 IP 地址和端口的输出和返回连接。[了解更多](#)

5.1.2 注册账户

企业账户的注册者成为“Team Agent”，负责账户及其成员的管理：

The screenshot shows the 'Membership' tab selected in the sidebar. The main content area displays the following membership information:

Program Type	Apple Developer Program
Team Name	Jiangsu Linkbit Micro Electronics Technology Co., Ltd.
Team ID	5HQ6LMM4P6
Entity Type	Company / Organization
Phone	025-XXXX-XXXX
Address	Room 602,Beidou Building,Hui Da Road No. 6,High-tech Zone,Pukou District,Nanjing, Jiangsu 210061 China
Expiration Date	May 12, 2017
Device Reset Date	May 12, 2017
Team Agent	chen ma
Your Role	Admin

5.1.3 管理成员

Agent 可以邀请其它苹果开发者账户成为本账户的成员，并赋予他们管理员角色：

The 'People' section of the account page is shown. On the left, there's a sidebar with links like Documentation, Downloads, and Forums. The main area shows two sections: 'Agent' and 'Admins (3)'. Under 'Agent', there's a list with 'chen ma'. Under 'Admins', there are three entries: '丽民路' (with email '123456789@qq.com'), '时空全息' (with email 'qxsk9@sina.com'), and '珊瑚任' (with email '123456789@163.com'). Each entry has a checkbox next to it.

管理员也可以添加账户成员，并赋予他们管理员角色：

The screenshot shows the 'People' section of the Apple Developer Account. On the left sidebar, under 'Program Resources', 'People' is selected. The main content area is titled 'Invite People' with the sub-section 'Invite as Admins'. It contains a text input field with the placeholder 'Enter one or more email addresses.' Below it is another section titled 'Invite as Members' with a similar input field. At the bottom right of the main area is a large 'Invite' button.

5.1.4 管理账户

在 iTunes Connect 界面，管理员可以使用多个功能：

The screenshot shows the iTunes Connect dashboard. It features several large icons with labels: '我的 App' (My App), 'App 分析' (App Analytics), '销售和趋势' (Sales & Trends), '付款和财务报告' (Billing & Financial Reports), '用户和职能' (Users & Roles), '协议、税务和银行业务' (Contracts, Tax, and Banking), and '资源和帮助' (Resources & Help). At the bottom of the page, there is a footer with copyright information and links to 'Terms of Service', 'Privacy Policy', and 'Contact Us'.

例如，检查账户成员的权限：

Apple ID	姓名 ^	职能	App
...n	machen	管理, 法务	所有 App
...n	任珊虹	管理	所有 App
qxsk9@sina.com	全息时空	管理	所有 App

5.1.5 管理应用

管理员可以设置应用的名字和其它属性：

APP STORE 信息

App 信息 存储

此信息用于这个 App 的所有平台。任何更改将在 App 的下一个版本中发布。

App 信息

若要更改 App 名称或类别，请创建一个新的 App 版本。其他更改将会立即显示在 App Store 中。

可本地化的信息

简体中文 ?

名称 ?
北斗视野
26

隐私政策网址(URL) ?
http://example.com (可不填)

副标题 ?
可选
30

综合信息

套装 ID ?
com.qxsk9.BeidouView

主要语言 ?
简体中文

SKU ?
001

类别 ?
导航

Apple ID ?
...n

许可协议 [编辑](#)
[Apple 标准许可协议](#)

分级 ?
4+岁 [其他分级](#)

可以查看版本历史和发布状态：

iOS App Store 版本

下面显示的版本为您的 App Store 版本。

- > 版本 1.2
- > 版本 1.1
- ▼ 版本 1.0

活动	用户	日期
● 可供销售	Apple	2016年5月22日 上午2:59
● 正在审核	Apple	2016年5月22日 上午2:57
● 元数据被拒绝	Apple	2016年5月20日 上午7:12
● 正在审核	Apple	2016年5月20日 上午1:34
● 正在等待审核	rshmara@sina.com	2016年5月17日 上午10:19
● 准备提交	rshmara@sina.com	2016年5月13日 下午4:52

可以查看评分和评论：

iOS App 评分与评论

中国 ▼ 最新发表 ▼

5.0 满分 5 分

1 个评分

所有版本 ▼ 所有评分 ▼ 所有评论 ▼ 1条评论

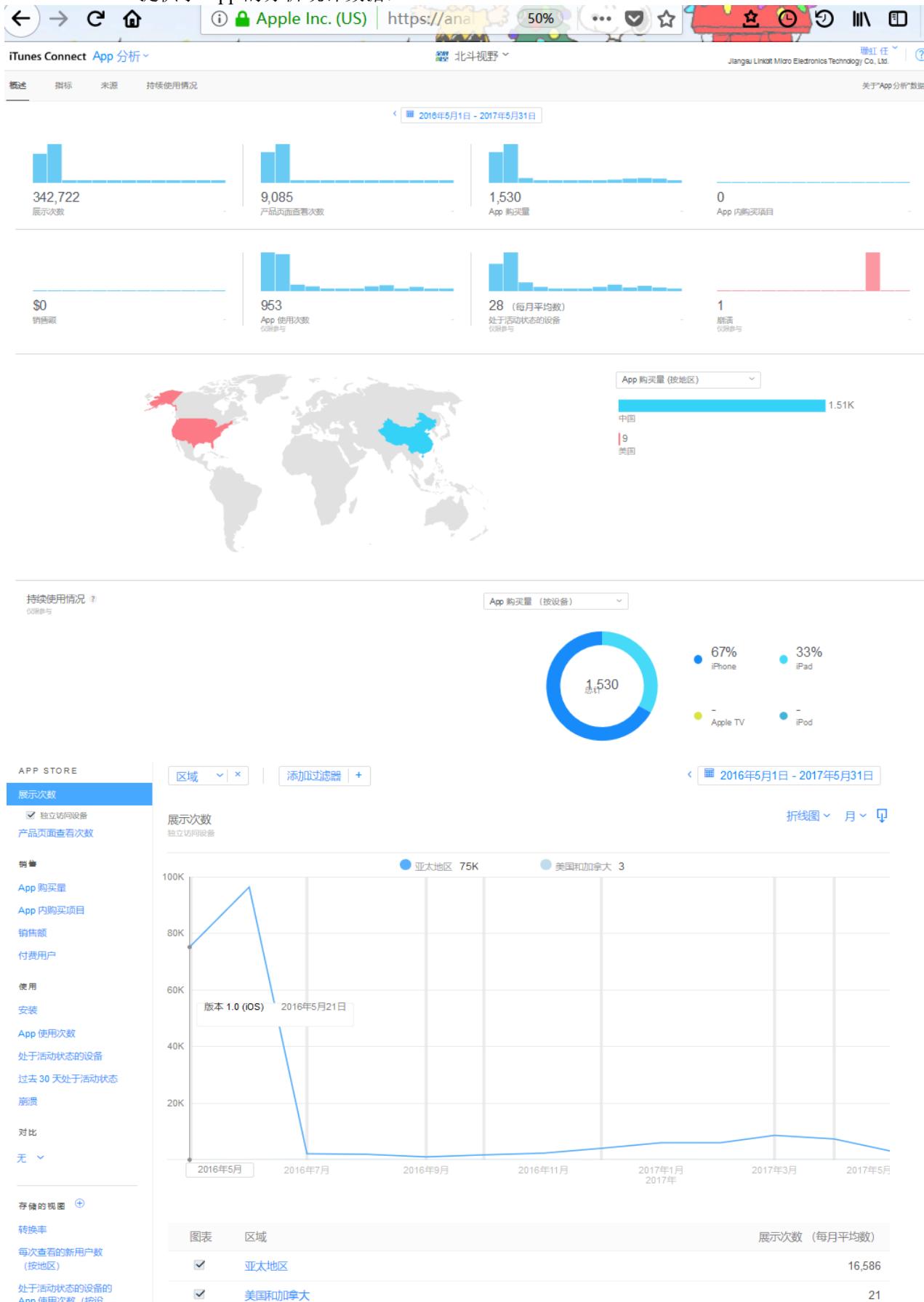
垃圾! ★☆☆☆☆
评论人：魍魉魑魅阿修罗 - 2016年6月5日

想说爱你很难，苹果6p下载后注册很顺利，地图下载也没有问题。但是下载完后卡顿，打不开图标，地图也打不开悲剧啊！果断卸载！！！

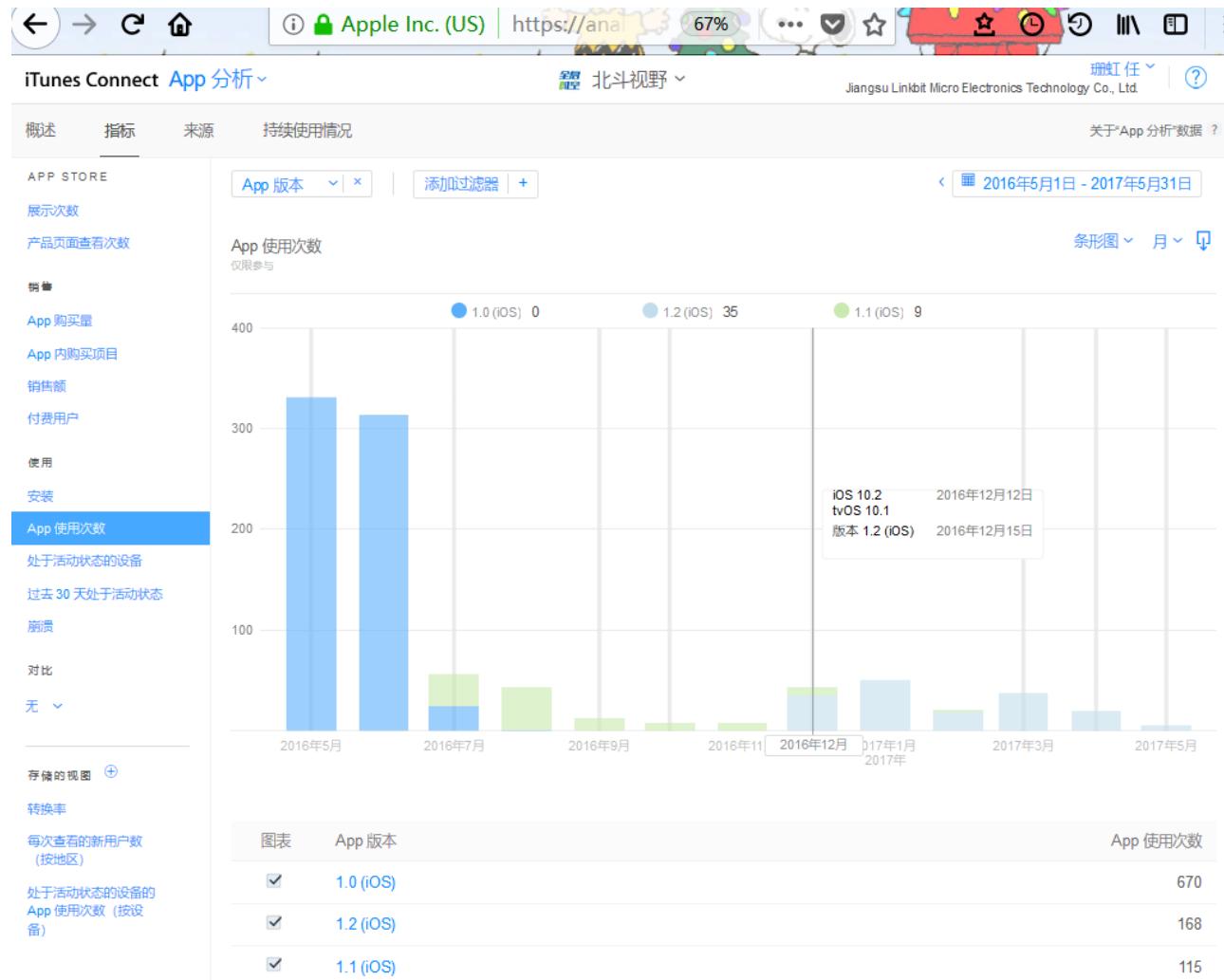
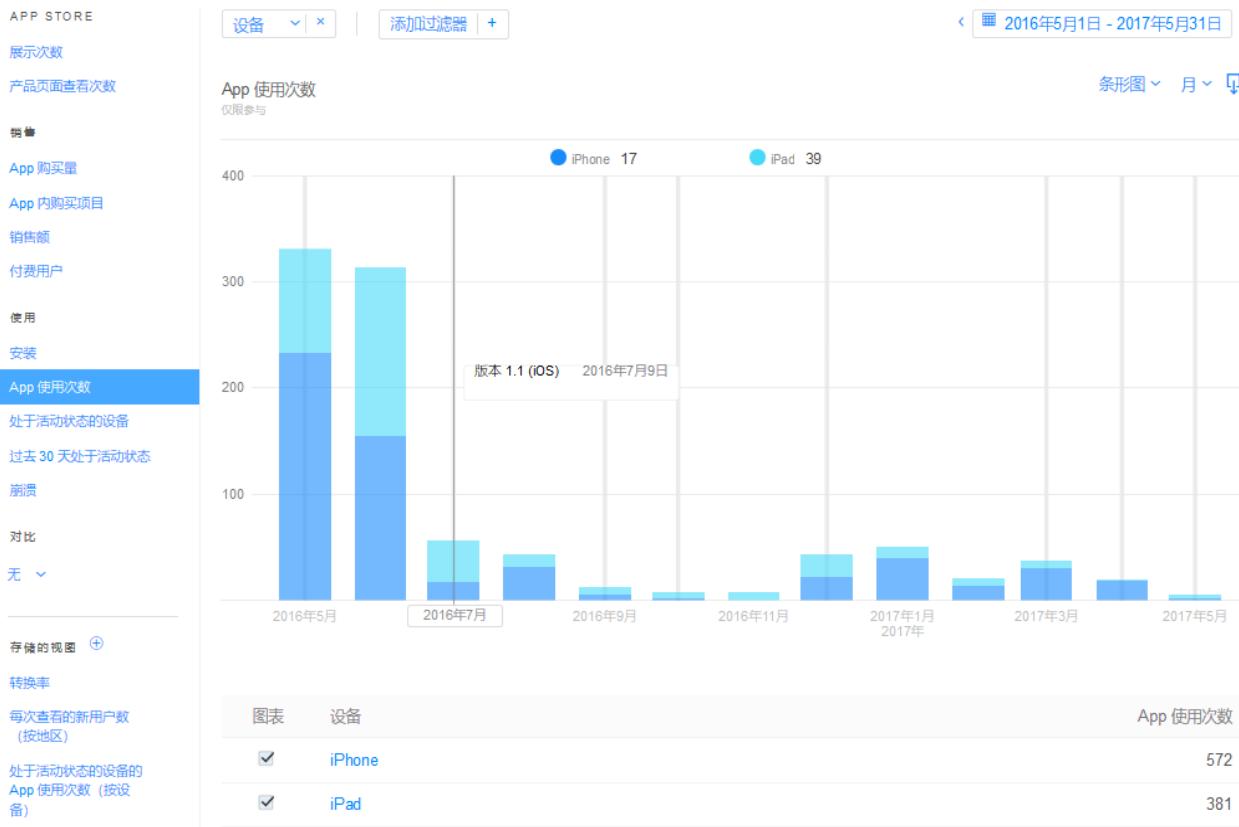
版本 1.0 | 中国 | 报告问题

5.1.6 App 分析数据

iTunes Connect 提供了 App 的分析统计数据：



全息时空运营服务平台 开发者指南



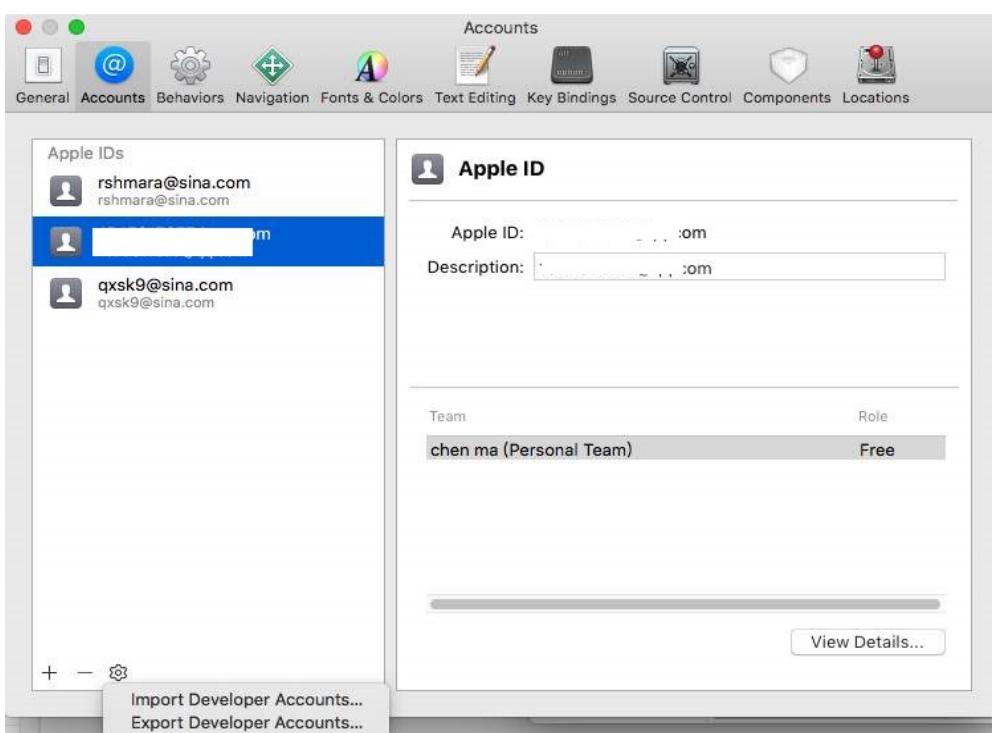
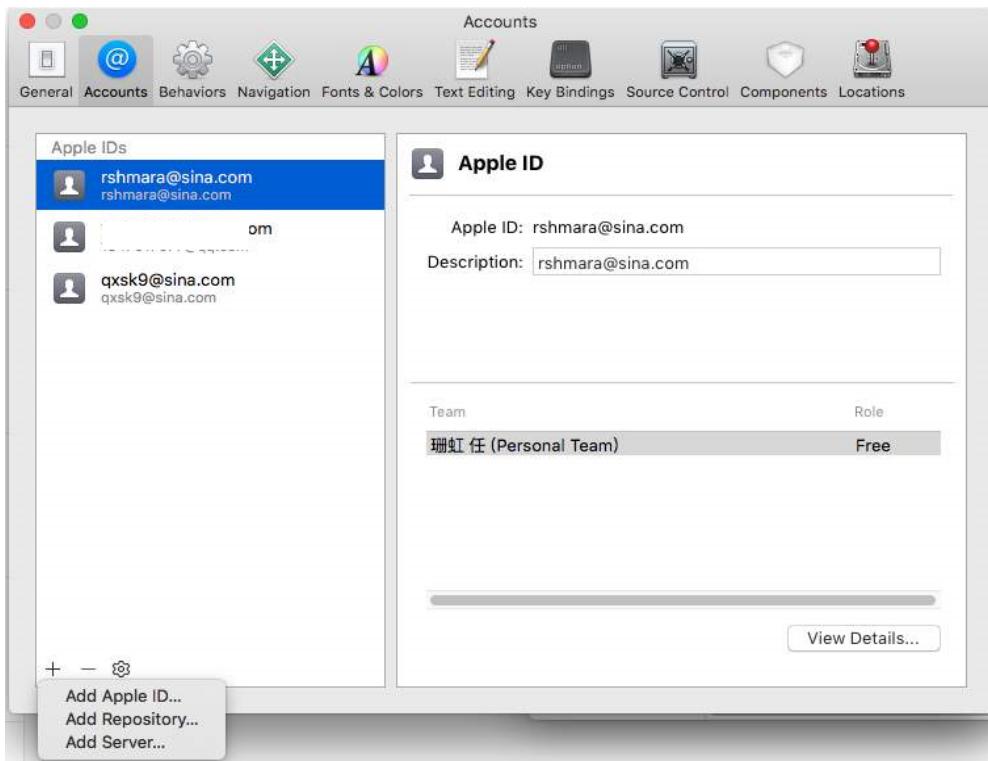


5.2 配置 Xcode

iOS 开发依赖于开发集成环境 Xcode，可以在苹果商店中免费下载。

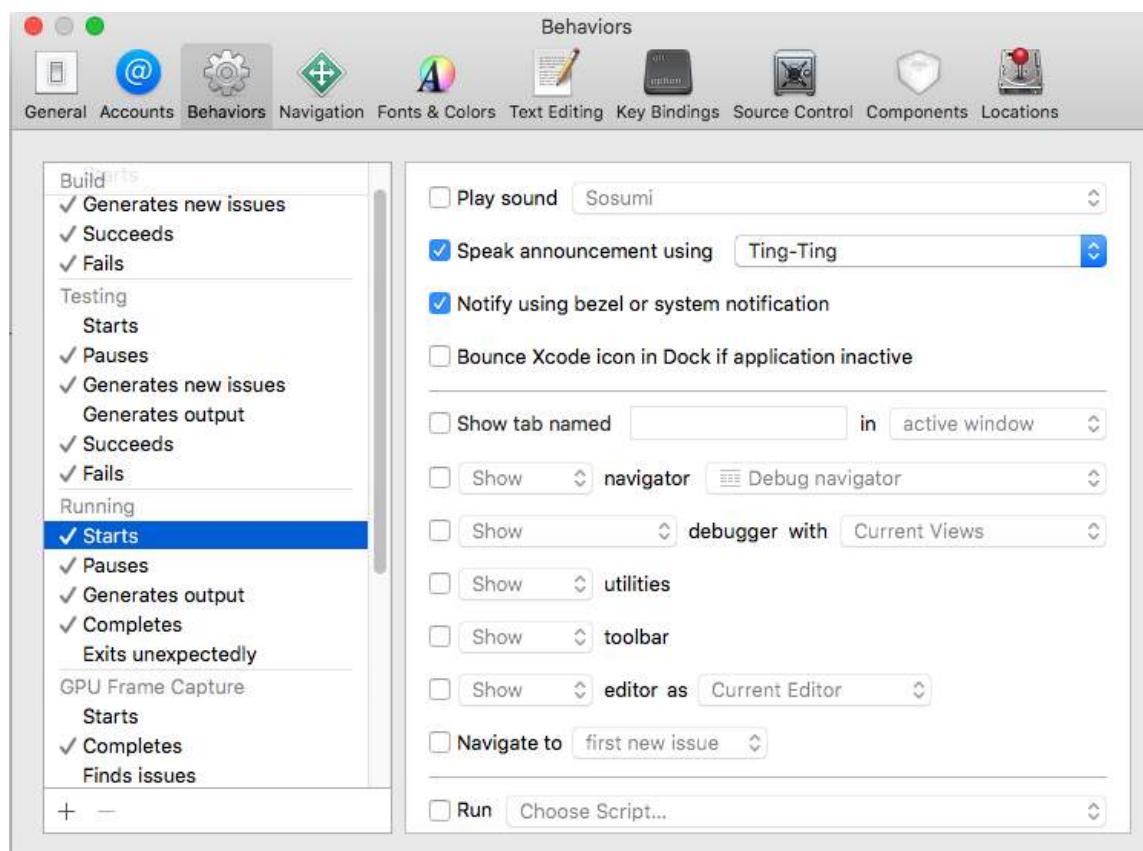
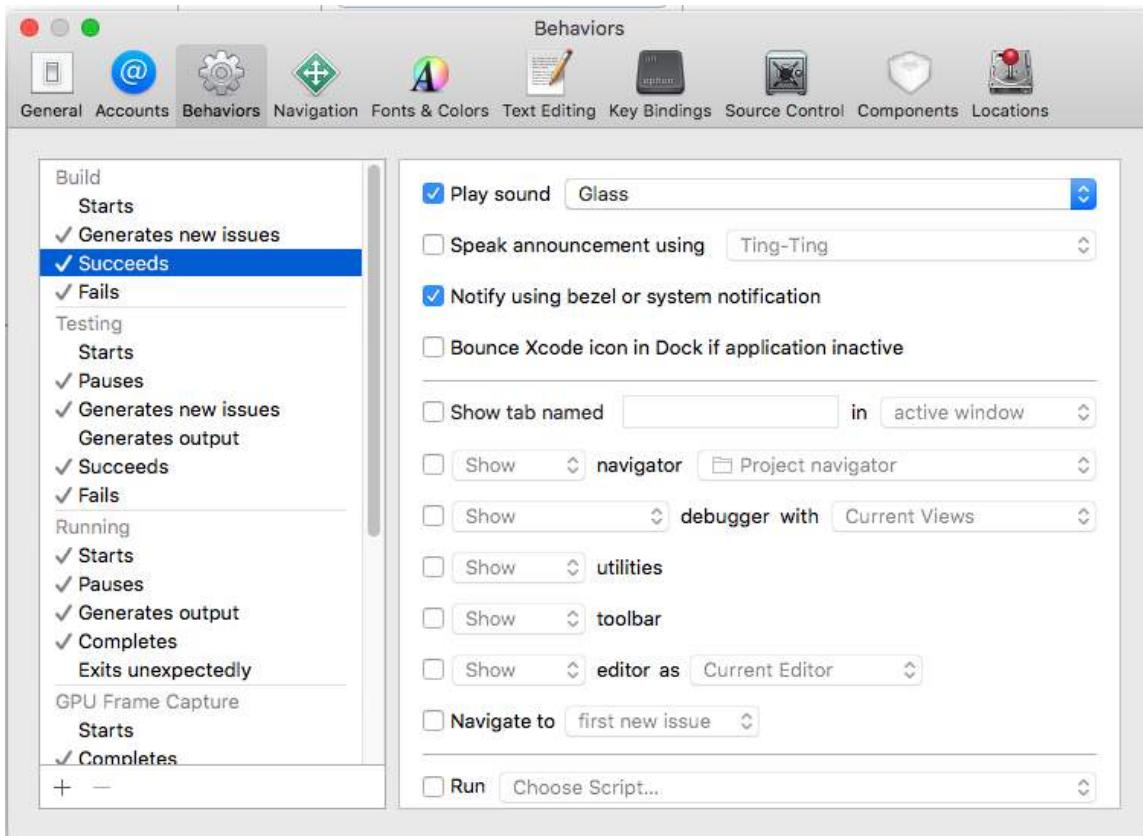
5.2.1 设置开发账户

打开 xcode，点击左上角的 xcode、选择 preferences、打开属性窗口，点击“Accounts”页签，添加和管理开发者账户：



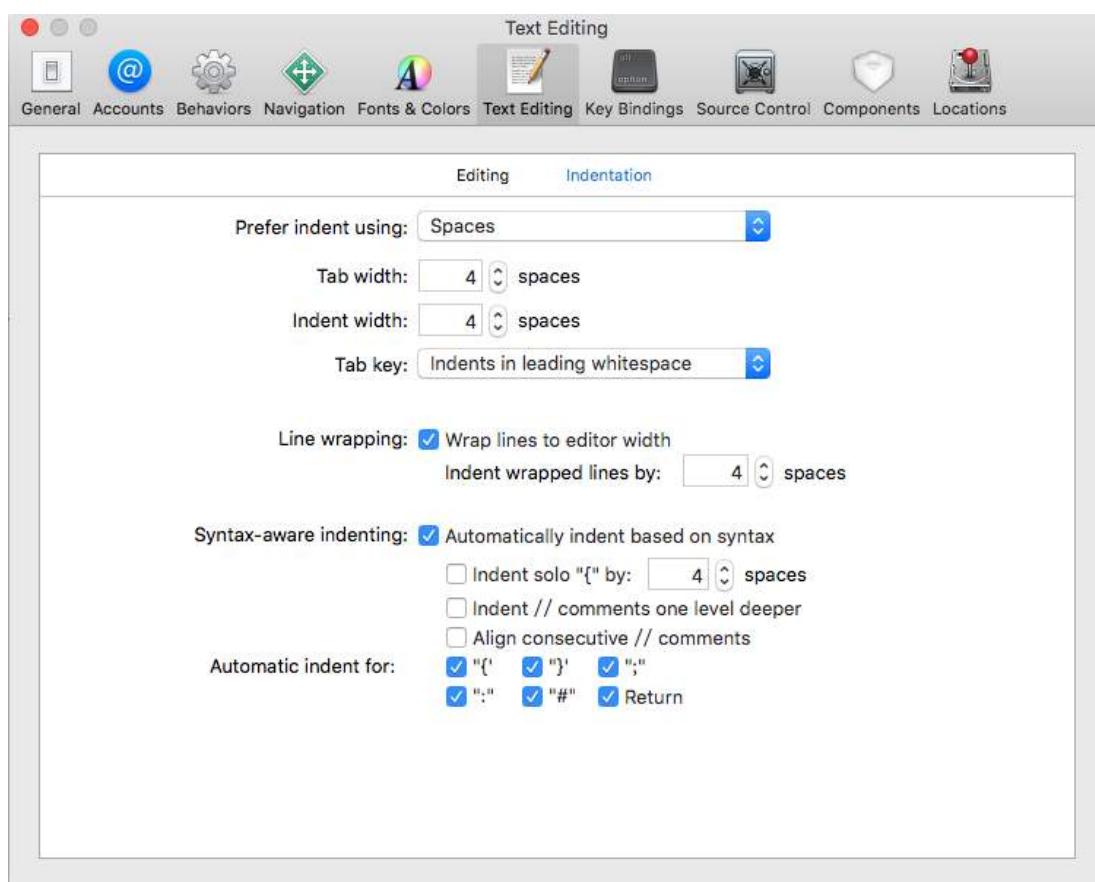
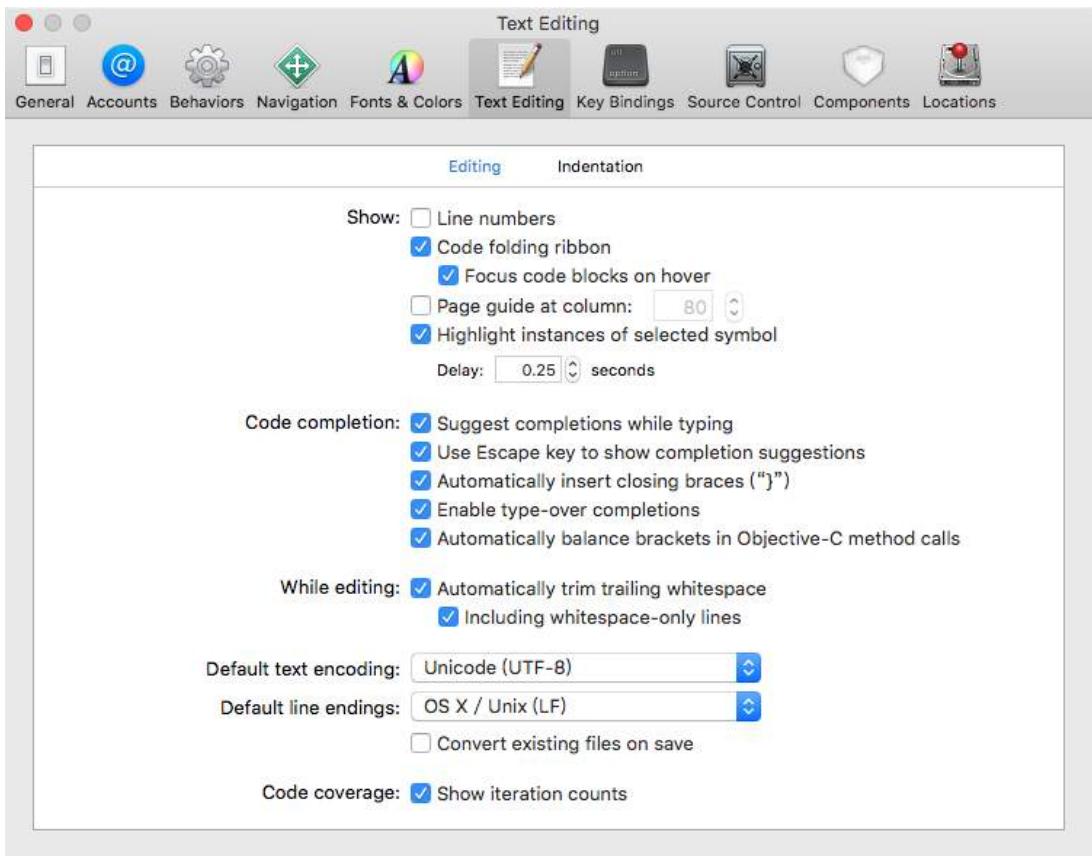
5.2.2 设置声音

可以设置在编译/运行时失败或者成功的提示语音：



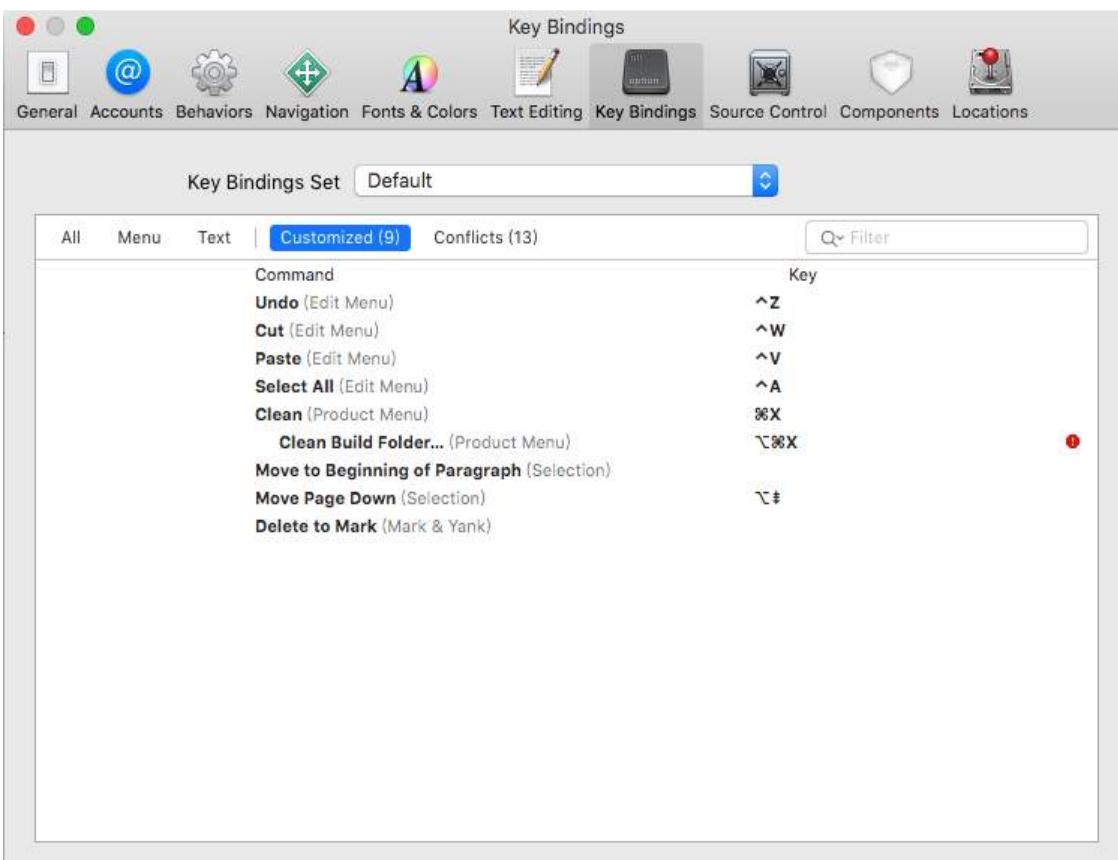
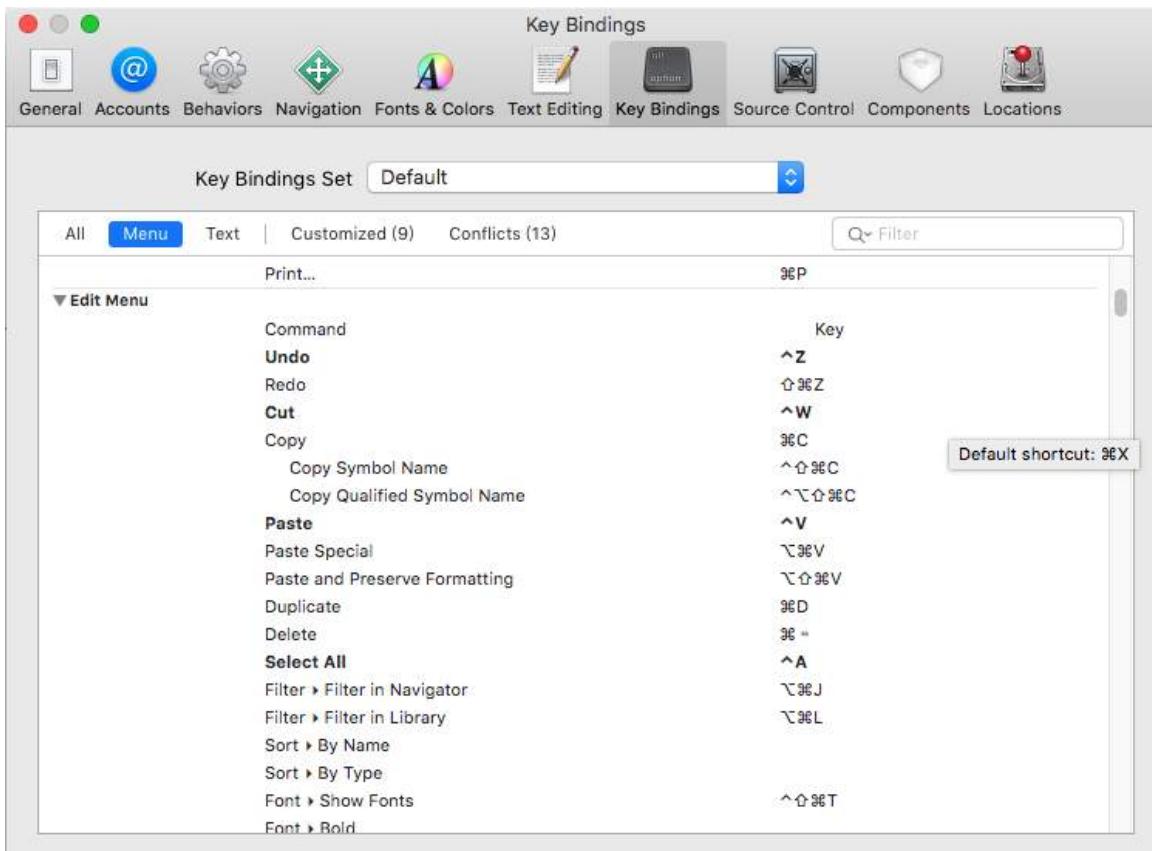
5.2.3 设置编辑选项

可以选择自己习惯的编辑参数：



5.2.4 定制快捷键

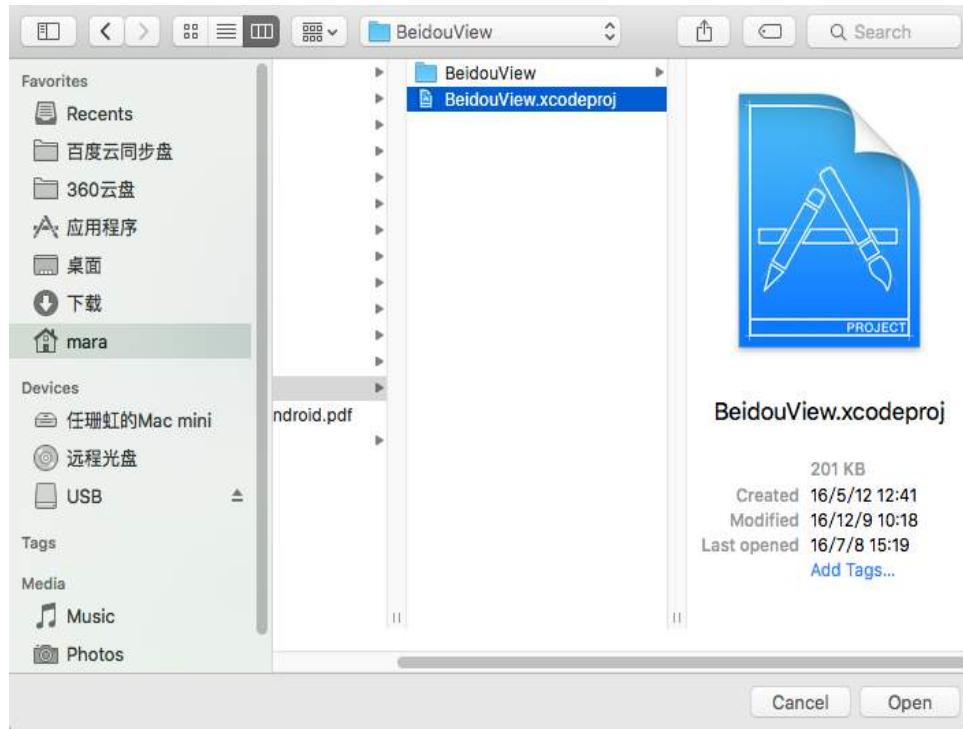
可以设置自己熟悉的快捷键（如 ctrl-c/ctrl-v）：



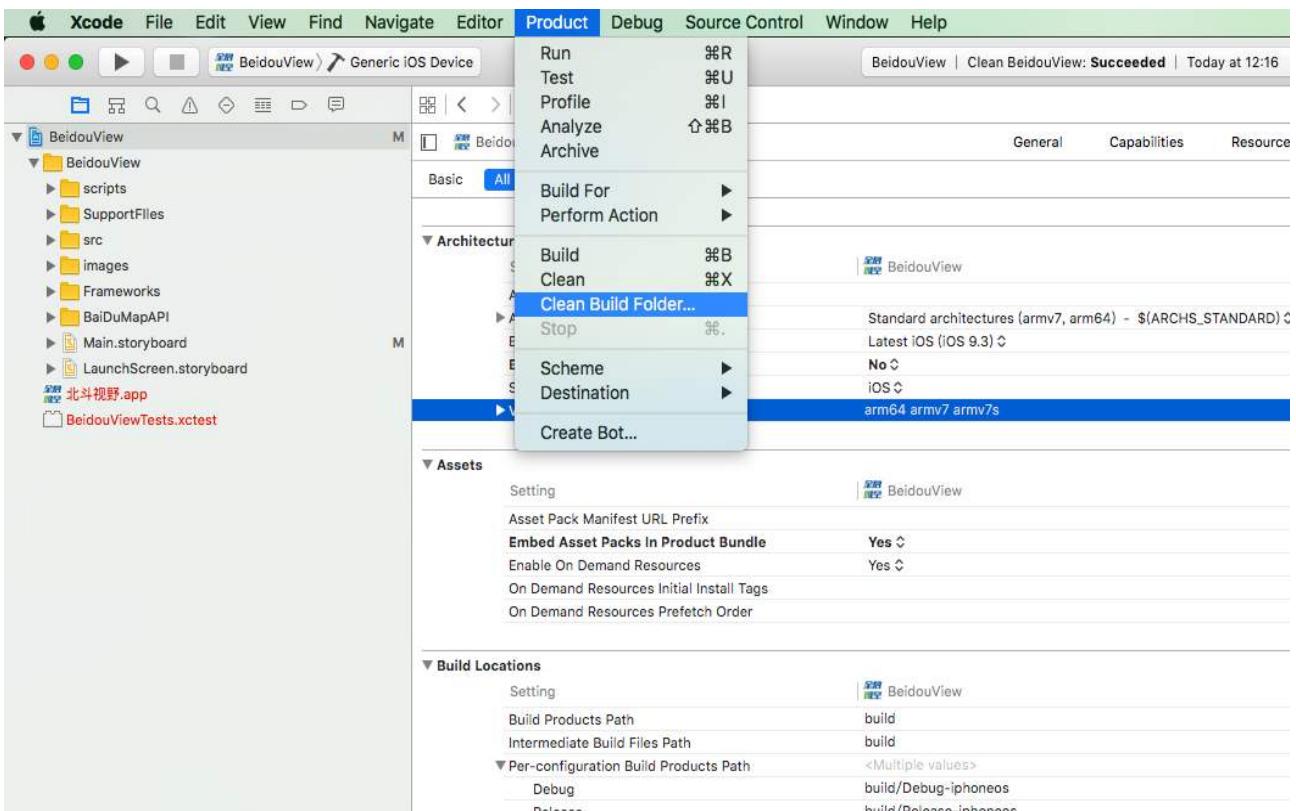
5.3 管理项目

5.3.1 项目管理文件

本应用的项目管理文件是“BeidouView.xcodeproj”，双击即可在 Xcode 中打开：



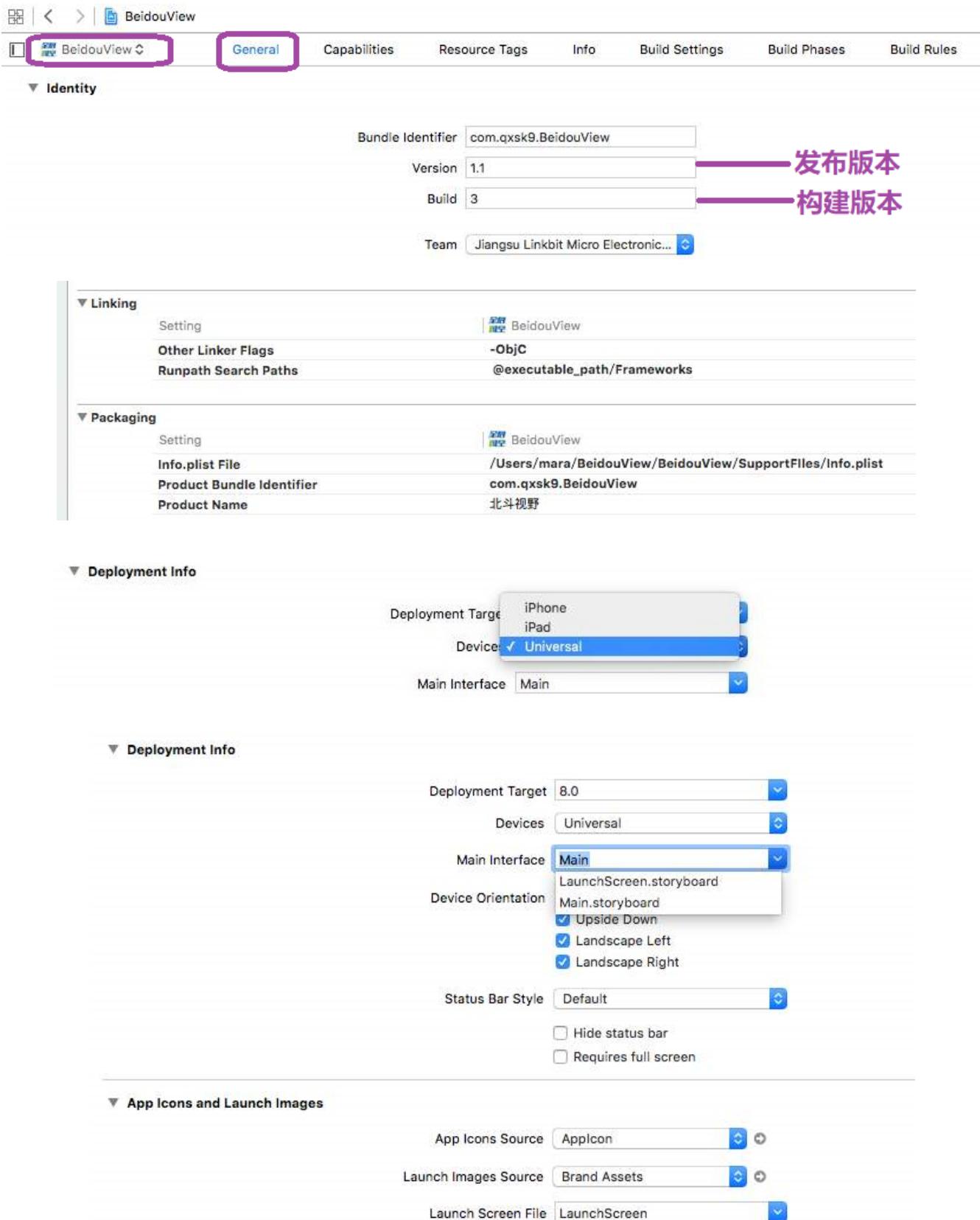
若项目打开后显示红色错误信息，可以点击菜单项“Product->Clean Build Folder...”以清理项目环境：



5.3.2 设置目标属性

5.3.2.1 “General”

选择页签“General”，设置项目的标识、版本、设备、主界面等属性，并且添加库文件：



▼ Linked Frameworks and Libraries

Name	Status
BaiduMapAPI_Base.framework	Required ▾
BaiduMapAPI_Location.framework	Required ▾
BaiduMapAPI_Map.framework	Required ▾
BaiduMapAPI_Search.framework	Required ▾
BaiduMapAPI_Utils.framework	Required ▾
libstdc++-6.0.9.tbd	Required ▾
CoreTelephony.framework	Required ▾
libsqllite3.0.tbd	Required ▾
Security.framework	Required ▾
CoreGraphics.framework	Required ▾
SystemConfiguration.framework	Required ▾
OpenGL ES.framework	Required ▾
CoreLocation.framework	Required ▾



添加地图库和其它必要的库文件

5.3.2.2 “Info”

选择页签“Info”，设置项目的本地语言、界面方向等：

BeidouView

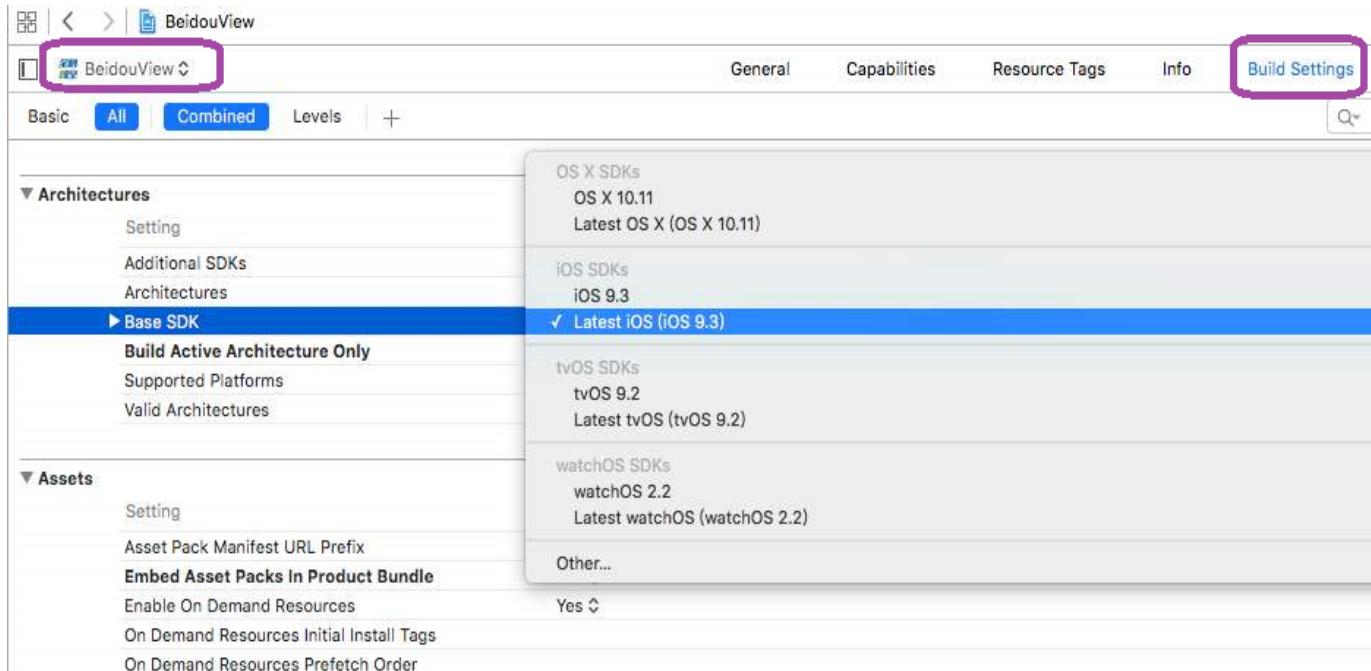
BeidouView General Capabilities Resource Tags **Info** Build Settings Build Phases Build Rules

Custom iOS Target Properties

Key	Type	Value
Bundle name	String	\$(PRODUCT_NAME)
Launch screen interface file base name	String	LaunchScreen
LSApplicationQueriesSchemes	Array	(1 item)
Item 0	String	baidumap
Localization native development region	String	China
Bundle version	String	3
Bundle OS Type code	String	APPL
Main storyboard file base name	String	Main
Bundle versions string, short	String	1.1
App Transport Security Settings	Dictionary	(1 item)
Allow Arbitrary Loads	Boolean	YES
InfoDictionary version	String	6.0
Executable file	String	\$(EXECUTABLE_NAME)
Required device capabilities	Array	(1 item)
Item 0	String	armv7
Supported interface orientations (iPad)	Array	(4 items)
Bundle identifier	String	\$(PRODUCT_BUNDLE_IDENTIFIER)
Application Category	String	????
Bundle creator OS Type code	String	YES
Application requires iPhone environment	Boolean	YES
Bundle display name	String	\$(PRODUCT_NAME)
NSLocationAlwaysUsageDescription	Boolean	YES
Supported interface orientations	Array	(4 items)
Item 0	String	Portrait (bottom home button)
Item 1	String	Landscape (left home button)
Item 2	String	Landscape (right home button)
Item 3	String	Portrait (top home button)
Document Types (0)		
Exported UTIs (0)		
Imported UTIs (0)		
URL Types (0)		

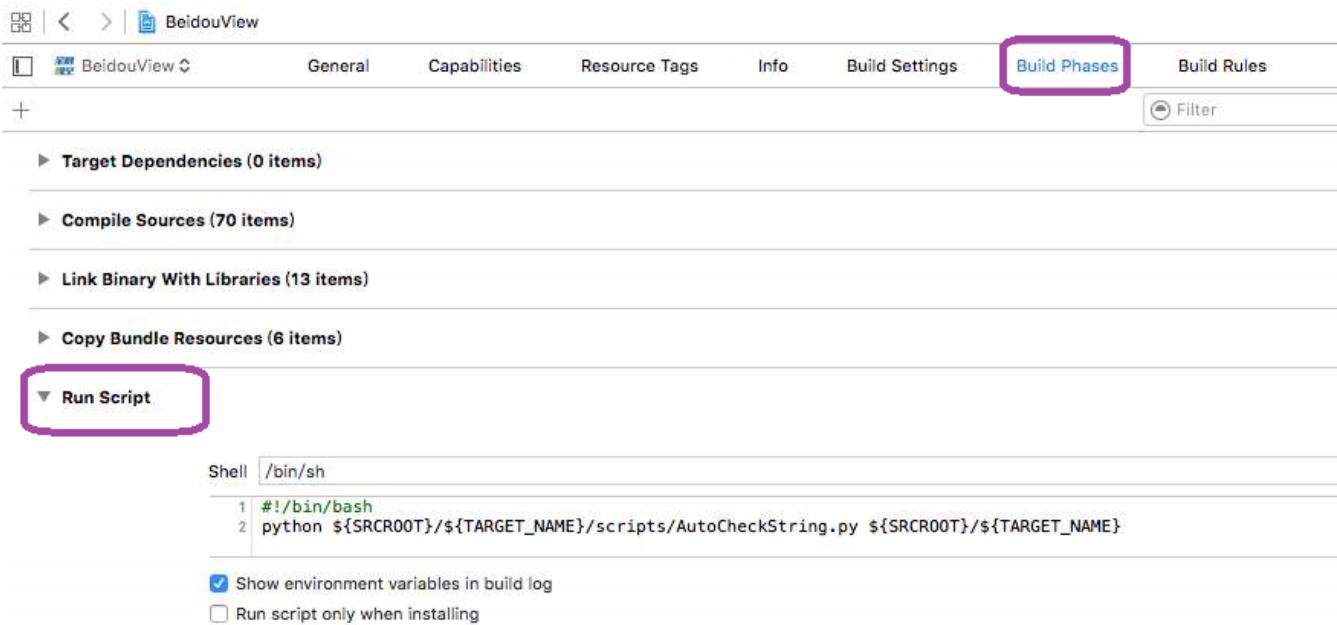
5.3.2.3 “Build Settings”

在项目属性界面中，选择目标为项目名、选择页签“Build Settings”，设置架构等属性：



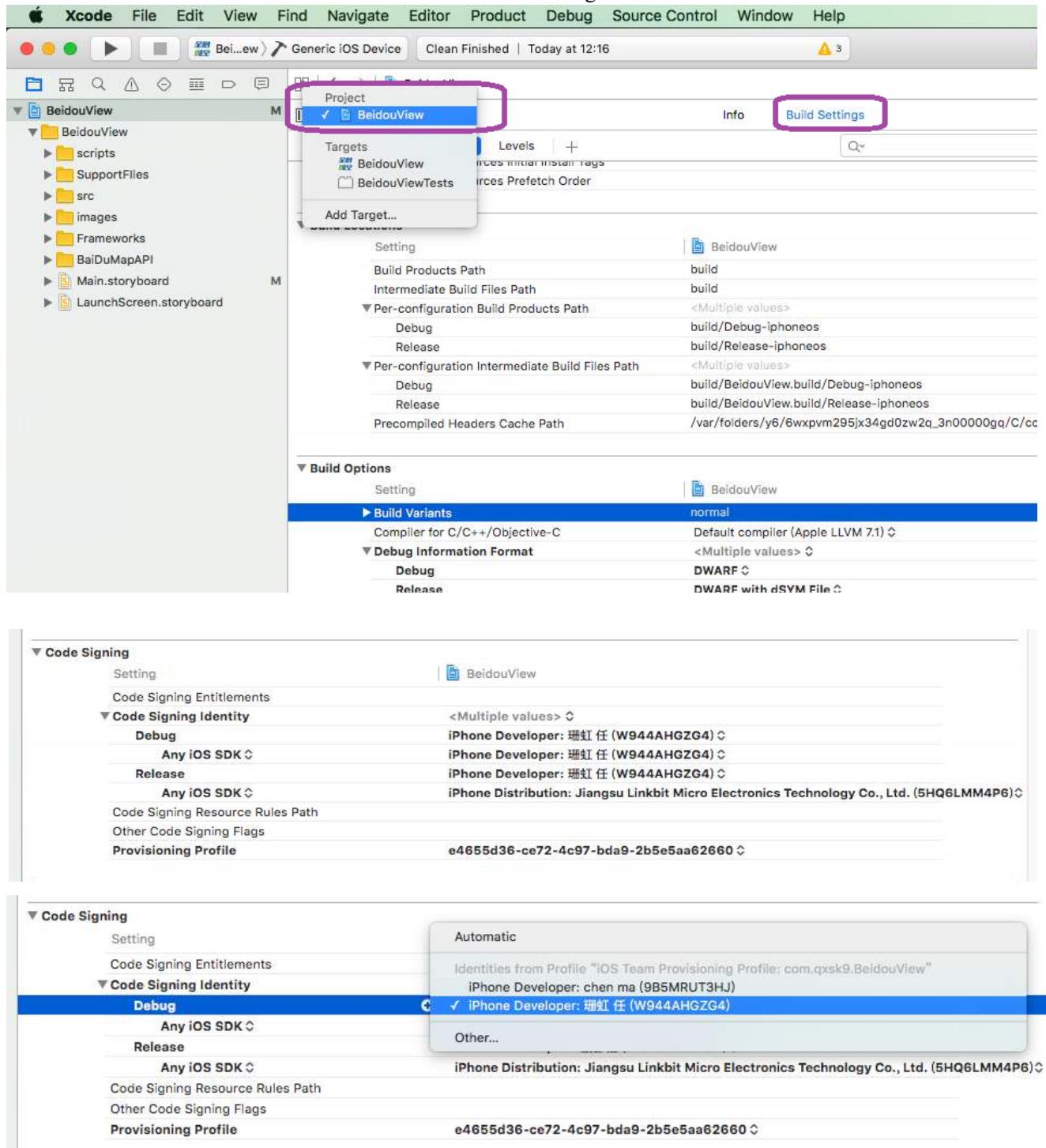
5.3.2.4 “Build Phases”

在页签“Build Phases”中填写构建时运行的脚本：



5.3.3 设置项目属性

在项目属性界面中，选择项目本身、选择页签“Build Settings”，设置构建、代码签名、发布等属性：



Code Signing

- BeidouView**
- Build Settings**

Deployment

- Setting
- Additional Strip Flags
- Alternate Install Group
- Alternate Install Owner
- Alternate Install Permissions
- Alternate Permissions Files
- Deployment Location
- Deployment Postprocessing
- Install Group
- Install Owner
- Install Permissions
- Installation Build Products Location
- Installation Directory
- OS X Deployment Target
- Resources Targeted Device Family
- Skip Install
- Strip Debug Symbols During Copy**
- Strip Linked Product
- Strip Style
- Targeted Device Family**
- Use Separate Strip
- iOS Deployment Target**
- tvOS Deployment Target
- watchOS Deployment Target

Headers

- Setting
- Run unifdef on Product Headers
- Unifdef Flags for Product Headers

Kernel Module

- Setting
- Module Identifier

iOS Deployment Target dropdown menu (highlighted):

- iOS 9.3
- iOS 9.2**
- iOS 9.1
- iOS 9.0
- iOS 8.4
- iOS 8.3
- iOS 8.2
- iOS 8.1
- iOS 8.0
- iOS 7.1
- iOS 7.0
- iOS 6.1
- iOS 6.0

Apple LLVM 7.1 - Code Generation

- Setting
- Generate Position-Dependent Code
- No Common Blocks
- Optimization Level**
- Debug
- Release

Apple LLVM 7.1 - Language

- Setting
- C Language Dialect

Apple LLVM 7.1 - Language - C++

- Setting
- C++ Language Dialect
- C++ Standard Library

Apple LLVM 7.1 - Language - Modules

- Setting
- Enable Modules (C and Objective-C)

Apple LLVM 7.1 - Language - Objective C

- Setting
- Objective-C Automatic Reference Counting

5.4 国际化

为了国际化应用，需要把界面和代码中的字串翻译成不同语言。如何找到所有需要国际化的字串并且一一添加到本地化文件中？这是一件枯燥易错的事。

我在网上搜到了一个 Python 脚本“AutoCheckString.py”，可以自动提取 StoryBoard 的字串并且写入缺省的本地化文件中，它还可以判断字串是否已在/已不在文件中（增量添加），非常有帮助：



自动提取Storyboard中的字串
并写入本地化文件中

```

Xcode  File  Edit  View  Find  Navigate  Editor  Product  Debug  Source Control  Window  Help
BeidouView  Generic iOS Device  Clean Finished | Today at 12:16
BeidouView  BeidouView  scripts  AutoCheckString.py  No Selection
BeidouView  M
BeidouView  scripts  AutoCheckString.py
SupportFiles  QXSApp-Bridging-Header.h
Assets.xcassets
Info.plist
EmptyFile.mm
Localizable.strings  Localizable.strings
src
images  mapapi.bundle
Frameworks  libsqlite3.0.tbd
libstdc++.6.0.9.tbd
CoreTelephony.framework
Security.framework
CoreGraphics.framework
SystemConfiguration.framework
OpenGLES.framework
CoreLocation.framework
BaiDuMapAPI  BaiduMapAPI_Base.framework
BaiduMapAPI_Lotion.framework
BaiduMapAPI_Map.framework
BaiduMapAPI_Search.framework
BaiduMapAPI_Utils.framework
Main.storyboard  Main.storyboard
LaunchScreen.storyboard  LaunchScreen.storyboard
untitled.py
Created by linyu on 2015-02-13.
Copyright (c) 2015 __MyCompanyName__. All rights reserved.

import imp
import sys
import os
import glob
import re
import time

imp.reload(sys)
sys.setdefaultencoding('utf-8') #设置默认编码,只能是utf-8,下面\u4e00-\u9fa5要求的

KSourceFile = 'Base.lproj/*.storyboard'
KTargetFile = '*.lproj/*.strings'
KGenerateStringsFile = 'TempfileOfStoryboardNew.strings'

ColonRegex = ur'["](.*?)["]'
KeyParamRegex = ur'["](.*?)["](\s*)=(\s*)["](.*?)["]'
AnnotationRegexPrefix = ur'/(.*?)/'

def getCharset(string_txt):
    filedata = bytearray(string_txt[:4])
    if len(filedata) < 4 :
        return 0
    if (filedata[0] == 0xEF) and (filedata[1] == 0xBB) and (filedata[2] == 0xBF):
        print 'utf-8'
        return 1
    elif (filedata[0] == 0xFF) and (filedata[1] == 0xFE) and (filedata[2] == 0x00):
        print 'utf-32/UCS-4,little endian'
        return 3
    elif (filedata[0] == 0x00) and (filedata[1] == 0x00) and (filedata[2] == 0xFE):
        print 'utf-32/UCS-4,big endian'
        return 3
    return 0

#compare and remove the useless param in original string
for key in originalString_dic:
    if(key not in newString_dic):
        keystr = '"%s"%key'
        replacestr = '/'+keystr
        match = re.search(replacestr , originalString_txt)
        if match is None:
            originalString_txt = originalString_txt.replace(keystr,replacestr)
#compare and add new param to original string
executeOnce = 1
for key in newString_dic:
    values = (key, newString_dic[key])
    if(key not in originalString_dic):
        newline = ''
        if executeOnce == 1:
            timestamp = time.strftime('%Y-%m-%d %H:%M:%S',time.localtime(time.time()))
            newline = '\n/#####' + AutoGenStrings + timestamp + '\n'
            newline += '#      AutoGenStrings      '+timestamp+'\n'
            newline += '/#####' + AutoGenStrings + timestamp + '\n'
            executeOnce = 0
        newline += '\n' + annotation_dic[key]
        newline += '\n"%s" = "%s";\n'%values
        originalString_txt += newline
#write into original file
sbfw=open(originalStringPath,"w")
sbfw.write(originalString_txt)
sbfw.close()

```

全息时空运营服务平台 开发者指南

以下是利用这个脚本生成的关键字、进行翻译后的本地化文件：

BeidouView M

- BeidouView
- scripts
- SupportFiles
- src
 - AppDelegate.swift M
 - Extentions
 - Objects
 - Utilities
 - ViewControllers
- images
- Frameworks
- BaiDuMapAPI
- Main.storyboard
 - Main.storyboard (Base) M
 - Main.strings (English) M
 - Main.strings (...e (Simplified)) M
- LaunchScreen.storyboard M

```
/* Class = "UITextField"; placeholder = "Enter user name"; ObjectID = "15L-6a-7QX"; */
"15L-6a-7QX.placeholder" = "输入用户名";

/* Class = "UIButton"; normalTitle = "Login"; ObjectID = "MRT-29-8e8"; */
"MRT-29-8e8.normalTitle" = "登录";

/* Class = "UITextField"; placeholder = "Enter password"; ObjectID = "qNk-n3-ps4"; */
"qNk-n3-ps4.placeholder" = "输入密码";

//#####
//#           AutoGenStrings          2016-03-20 12:21:28
//#####

/* Class = "UIButton"; normalTitle = "Terminal"; ObjectID = "BMA-mz-g87"; */
"BMA-mz-g87.normalTitle" = "终端";

/* Class = "UIButton"; normalTitle = "Map"; ObjectID = "jme-v1-hEE"; */
"jme-v1-hEE.normalTitle" = "地图";

/* Class = "UIButton"; normalTitle = "About"; ObjectID = "1n8-Oy-gVY"; */
"1n8-Oy-gVY.normalTitle" = "关于";

/* Class = "UIButton"; normalTitle = "Feedback"; ObjectID = "crp-Zz-wb7"; */
"crp-Zz-wb7.normalTitle" = "反馈";

/* Class = "UIButton"; normalTitle = "Fence"; ObjectID = "bc5-bP-3k0"; */
"bc5-bP-3k0.normalTitle" = "围栏";

/* Class = "UIButton"; normalTitle = "Password"; ObjectID = "vv9-C9-FuC"; */
"vv9-C9-FuC.normalTitle" = "密码";

/* Class = "UIButton"; normalTitle = "Track"; ObjectID = "WFu-qP-BvZ"; */
//"WFu-qP-BvZ.normalTitle" = "轨迹";

/* Class = "UIButton"; normalTitle = "Authorize"; ObjectID = "JbF-3u-wrp"; */
"JbF-3u-wrp.normalTitle" = "授权";

//#####
//#           AutoGenStrings          2016-03-20 18:05:56
//#####

/* Class = "UILabel"; text = "User name"; ObjectID = "Cae-1o-N4D"; */
"Cae-1o-N4D.text" = "用户名";
```

The screenshot shows the Xcode interface with a project tree on the left and a code editor on the right.

Project Tree:

- BeidouView
- BeidouView
- scripts
- SupportFiles
- src
 - AppDelegate.swift
 - Extentions
 - Objects
 - Utilities
 - ViewControllers
- images
- Frameworks
- BaiDuMapAPI
- Main.storyboard
 - Main.storyboard (Base)
 - Main.strings (English) **Selected**
 - Main.strings (...e (Simplified))
- LaunchScreen.storyboard

Code Editor (Main.strings (English)):

```
/* Class = "UITextView"; text = "全系时空北斗运营服务平台"; ObjectID = "g0Z-gB-FJv"; */
//g0Z-gB-FJv.text" = "全系时空北斗运营服务平台";

//Wmv-9e-rG4.segmentTitles[1] = "60";

/* Class = "UILabel"; text = "Terminal"; ObjectID = "g4q-JV-gAU"; */
"g4q-JV-gAU.text" = "Terminal";

//lVQ-to-neS.segmentTitles[3] = "600";

/* Class = "UILabel"; text = "Password again"; ObjectID = "B1h-qU-26R"; */
"B1h-qU-26R.text" = "Password again";

/* Class = "UITabBarItem"; title = "Setting"; ObjectID = "XE0-cZ-Cud"; */
"XE0-cZ-Cud.title" = "Setting";

/* Class = "UIButton"; normalTitle = "Select"; ObjectID = "uWH-9f-YTn"; */
"uWH-9f-YTn.normalTitle" = "Select".
```

这个脚本可以设置为在每次构建应用时自动执行，参见 5.3.2.4 节。

5.5 应用地图

5.5.1 申请 AK

iOS 开发需要申请专门的 AK:

<http://lbsyun.baidu.com/apiconsole/key>

The screenshot shows the Baidu Map API console interface. On the left, there's a sidebar with navigation links: '我的应用' (My Application), '查看应用' (View Application), '创建应用' (Create Application), '回收站' (Recycle Bin), '我的配额' (My Quota), '配额管理' (Quota Management), '申请记录中心' (Application Record Center), '我的数据' (My Data), '数据管理平台' (Data Management Platform), '商用授权' (Commercial Authorization), '商用授权' (Commercial Authorization again), and '订单中心' (Order Center). The main content area is titled '修改应用' (Modify Application) and shows fields for '应用AK' (Application AK), '应用名称' (Application Name: 北斗视野), '应用类型' (Application Type: iOS端), and a list of service checkboxes. A note at the bottom states: '新申请的Mobile与Browser类型的ak不再支持云存储接口的访问,如要使用云存储,请申请Server类型ak。' (Newly applied Mobile and Browser types of ak no longer support cloud storage interface access. If you need to use cloud storage, please apply for a Server type ak.) There is also a '安全码' (Security Code) field containing 'com.qxsk9.BeidouView' and a '提交' (Submit) button.

5.5.2 导入地图库

参见官网指南：

<http://lbsyun.baidu.com/index.php?title=iossdk/guide/create-project/androidwear>

如下编写头文件 “QXSKapp-Bridging-Header.h”：

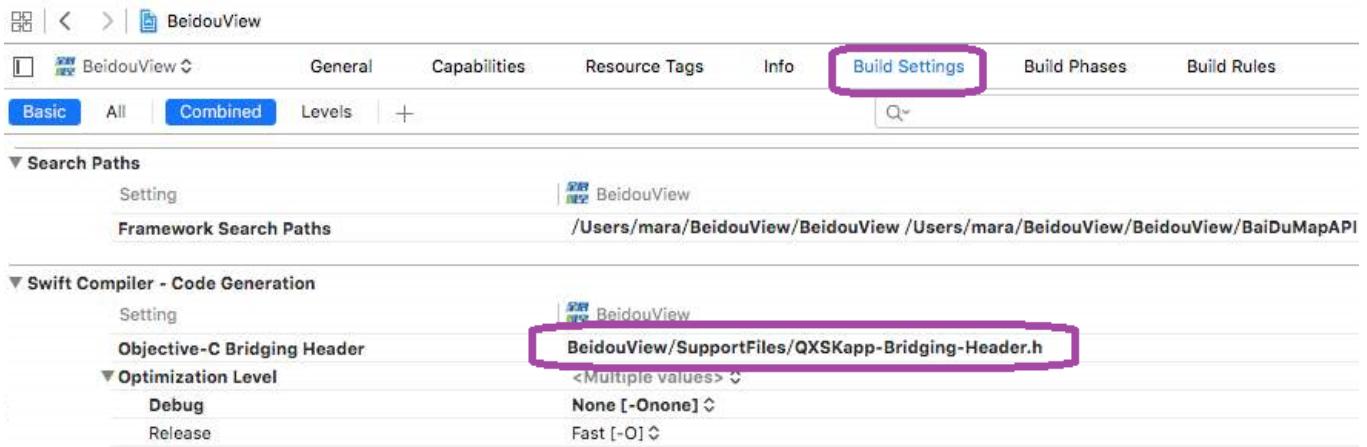
The screenshot shows the Xcode interface with the file 'QXSKapp-Bridging-Header.h' selected in the project navigator. The code in the editor is as follows:

```

// Use this file to import your target's public headers that you would like to expose to Swift.
//

#import <BaiduMapAPI_Base/BMKBaseComponent.h> //引入base相关所有的头文件
#import <BaiduMapAPI_Map/BMKMapView.h> //引入地图功能所有的头文件
#import <BaiduMapAPI_Location/BMKLocationComponent.h> //引入定位功能所有的头文件
#import <BaiduMapAPI_Utils/BMKUtilsComponent.h> //引入计算工具所有的头文件
#import <BaiduMapAPI_Map/BMKMapView.h> //只引入所需的单个头文件
#import <BaiduMapAPI_Search/BMKSearchComponent.h> //引入检索功能所有的头文件
//#import <BaiduMapAPI_Cloud/BMKCloudSearchComponent.h> //引入云检索功能所有的头文件
//#import <BaiduMapAPI_Radar/BMKRadarComponent.h> //引入周边雷达功能所有的头文件

```



5.5.3 初始化地图 SDK

在“AppDelegate.swift”中初始化百度地图接口：

```

1 // AppDelegate.swift
2 // QXSKapp
3 // Created by 玛瑞 on 16/3/19.
4 //
5 import UIKit
6
7 @UIApplicationMain
8 class AppDelegate: UIResponder, UIApplicationDelegate, BMKGeneralDelegate {
9
10    var window: UIWindow?
11
12    var mapManager: BMKMapManager?
13
14
15    func application(application: UIApplication, didFinishLaunchingWithOptions launchOptions: [NSObject: AnyObject]?) -> Bool {
16        // Override point for customization after application launch.
17
18        /* ---- 初始化全局变量和系统参数 --- */
19        loadCurrentUser()
20
21        /* ---- 初始化百度地图接口 --- */
22        mapManager = BMKMapManager()
23        // 如果要关注网络及授权验证事件, 请设定generalDelegate参数
24        let ret = mapManager?.start(BaiDuXcodeKey, generalDelegate: nil)
25        if ret == true
26        {
27            //PrintLog("百度地图接口成功启动")
28        } else {
29            PrintLog("百度地图接口启动失败")
30        }
31
32
33
34
35
36
37
38
39        return true
40    }

```

5.5.4 View 中的地图

在 View 的各个阶段注意处理地图的初始化和销毁:

```

new 2 | AppDelegate.swift | LocationViewController.swift |
46  override func viewDidLoad() {
47      super.viewDidLoad()

48
49      mapView.showMapPoi = true // 设定地图是否显示底图poi标注, 默认YES
50      mapView.zoomEnabled = true // 设定地图view能否支持用户多点缩放(双指)
51      mapView.zoomEnabledWithTap = true // 设定地图View能否支持用户缩放(双击或双指单击)
52      mapView.zoomLevel = 15 // 地图级别 3~21
53      mapView.gesturesEnabled = true // 支持所有手势
54      mapView.scrollEnabled = true // 设定地图view能否支持用户移动地图
55      mapView.overlookEnabled = true // 设定地图view能否支持俯仰角
56      mapView.rotateEnabled = true // 设定地图view能否支持旋转
57      mapView.showMapScaleBar = true // 设定是否显式比例尺
58      mapView.showsUserLocation = true // 设定是否显示定位图层
59
60
61      activityIndicator = UIActivityIndicatorView(activityIndicatorStyle: UIActivityIndicatorViewStyle.gray)
62      activityIndicator.center = self.view.center
63      self.view.addSubview(activityIndicator)
64
65  }
66
67  override func viewDidAppear(animated: Bool) {
68
69  override func viewWillAppear(animated: Bool) {
70      super.viewWillAppear(animated)
71      mapView.viewWillAppear()
72      mapView.delegate = self // 此处记得不用的时候需要置nil, 否则影响内存的释放
73  }
74
75  override func viewWillDisappear(animated: Bool) {
76      super.viewWillDisappear(animated)
77
78      if ( timer != nil ) {
79          timer!.invalidate()
80      }
81
82      mapView.viewWillDisappear()
83      mapView.delegate = nil // 不用时, 置nil
84
85  }
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
}

```

初始化地图参数

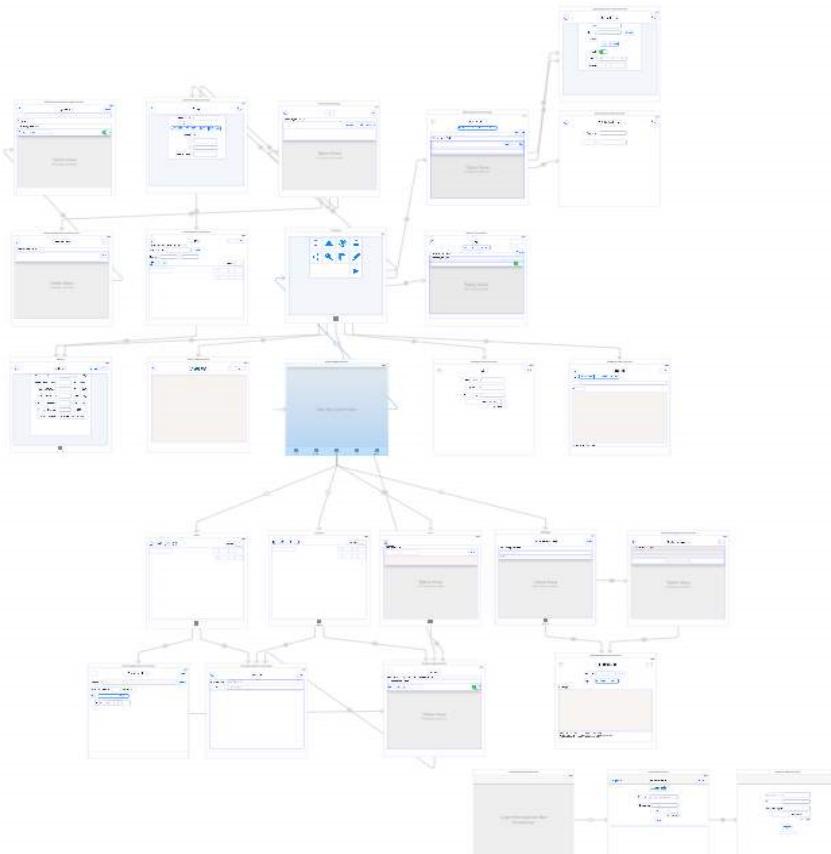
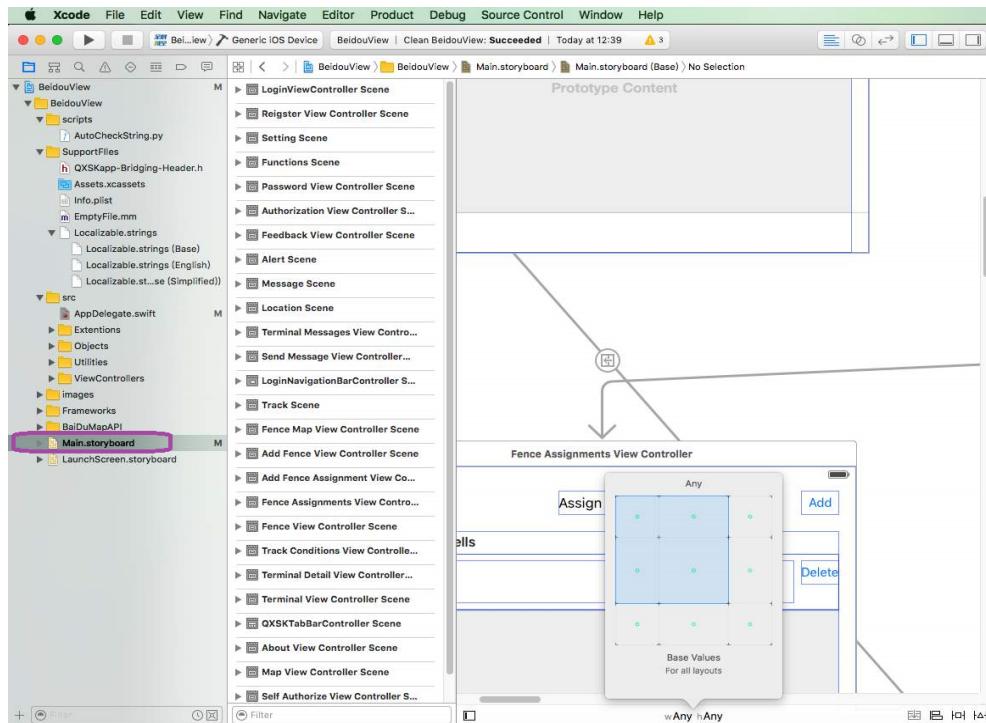
初始化地图

释放地图资源

5.6 故事板 (Storyboard)

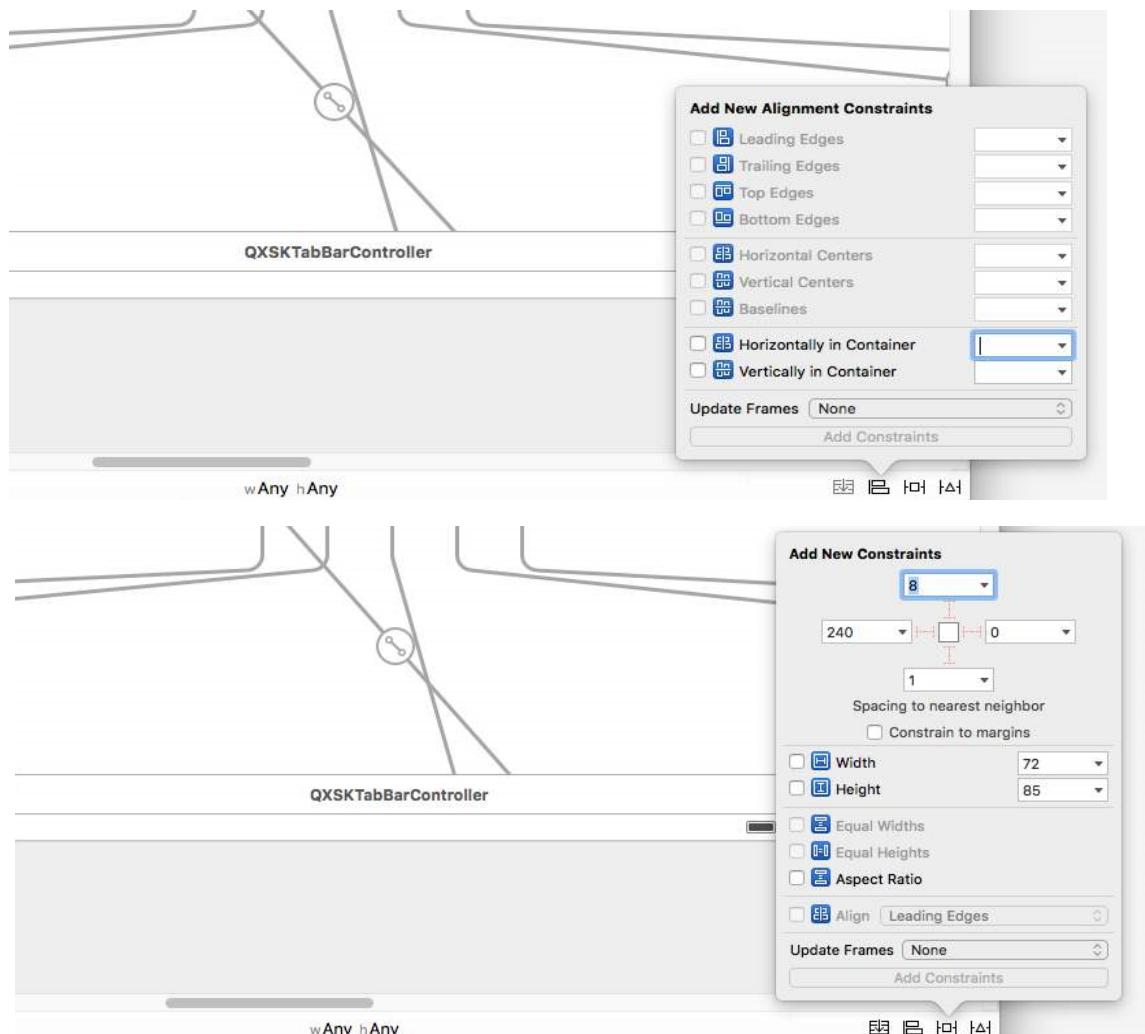
5.6.1 应用的故事板

应用的故事板是“Main.storyboard”：



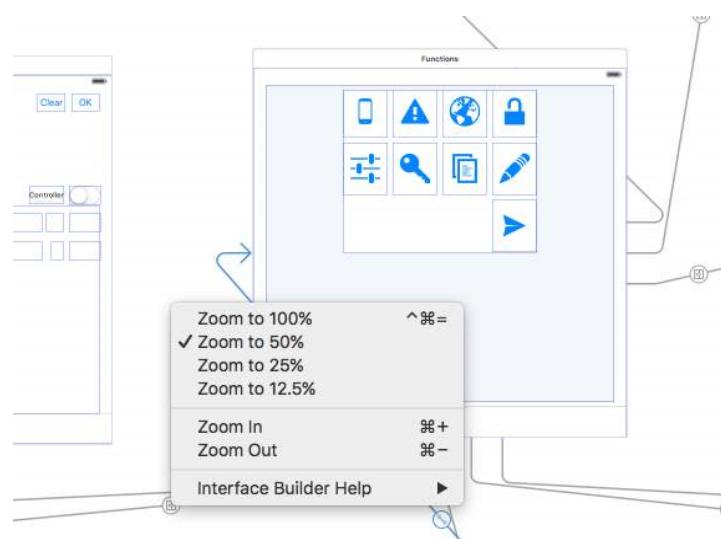
5.6.2 功能按钮

可以利用右下角的功能按钮设置界面元素的对齐约束，可以极大提高设计效率：



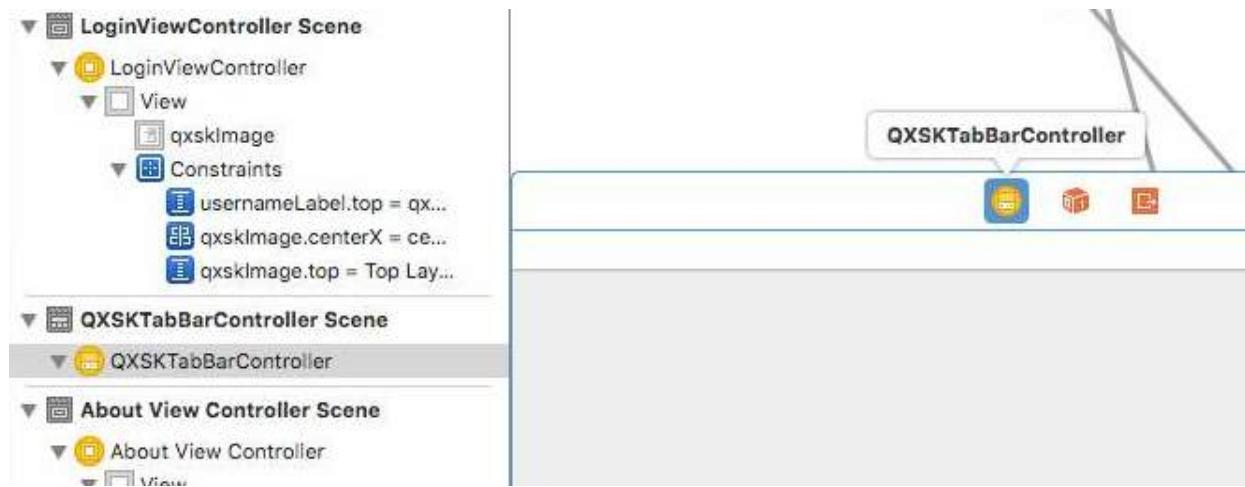
5.6.3 缩放故事板

点击右键，可以设置画板的缩放率：

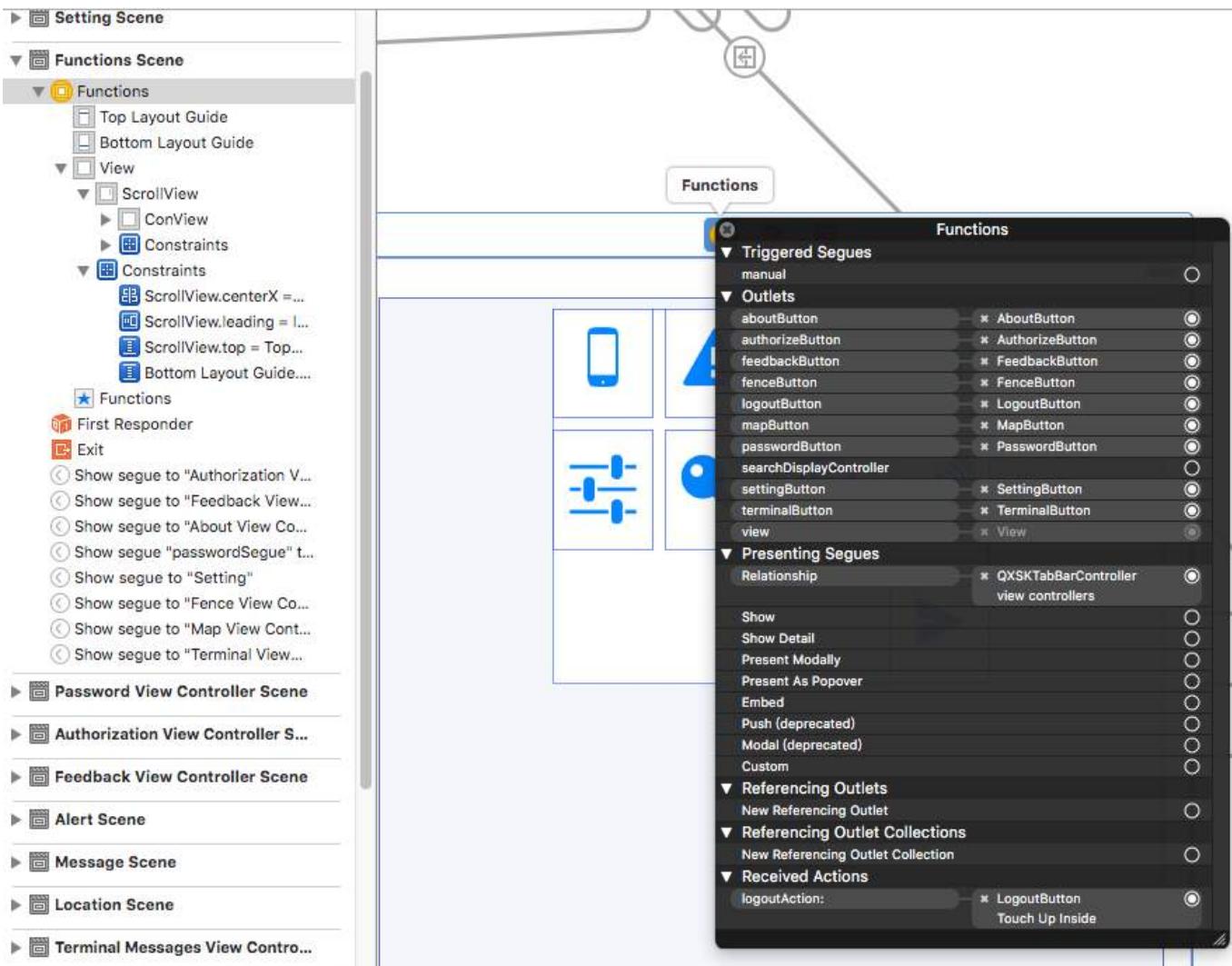


5.6.4 场景 (Scene) 和联系 (Segues)

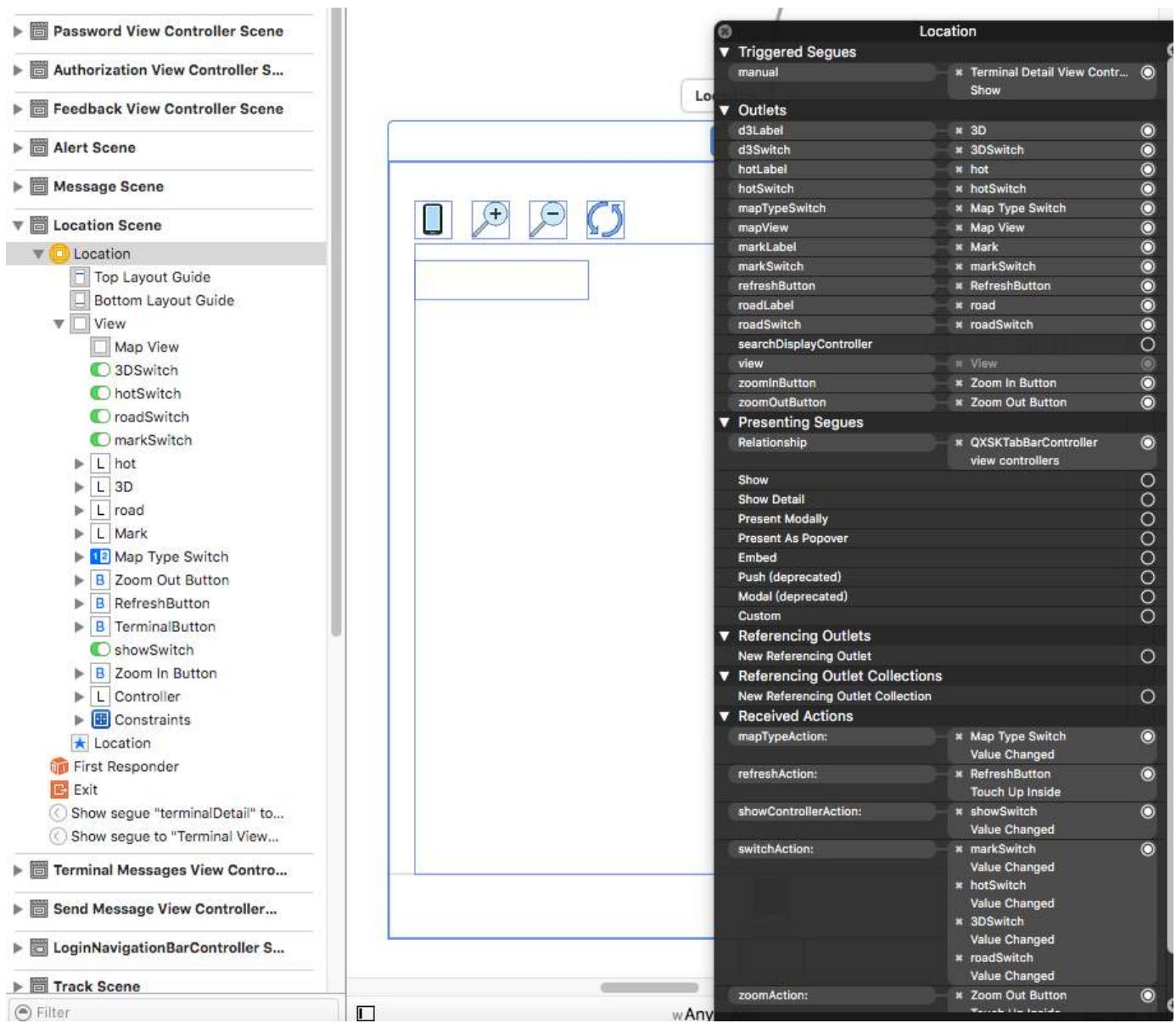
故事板左边是场景导览器，可以方便地选中和设置属性：



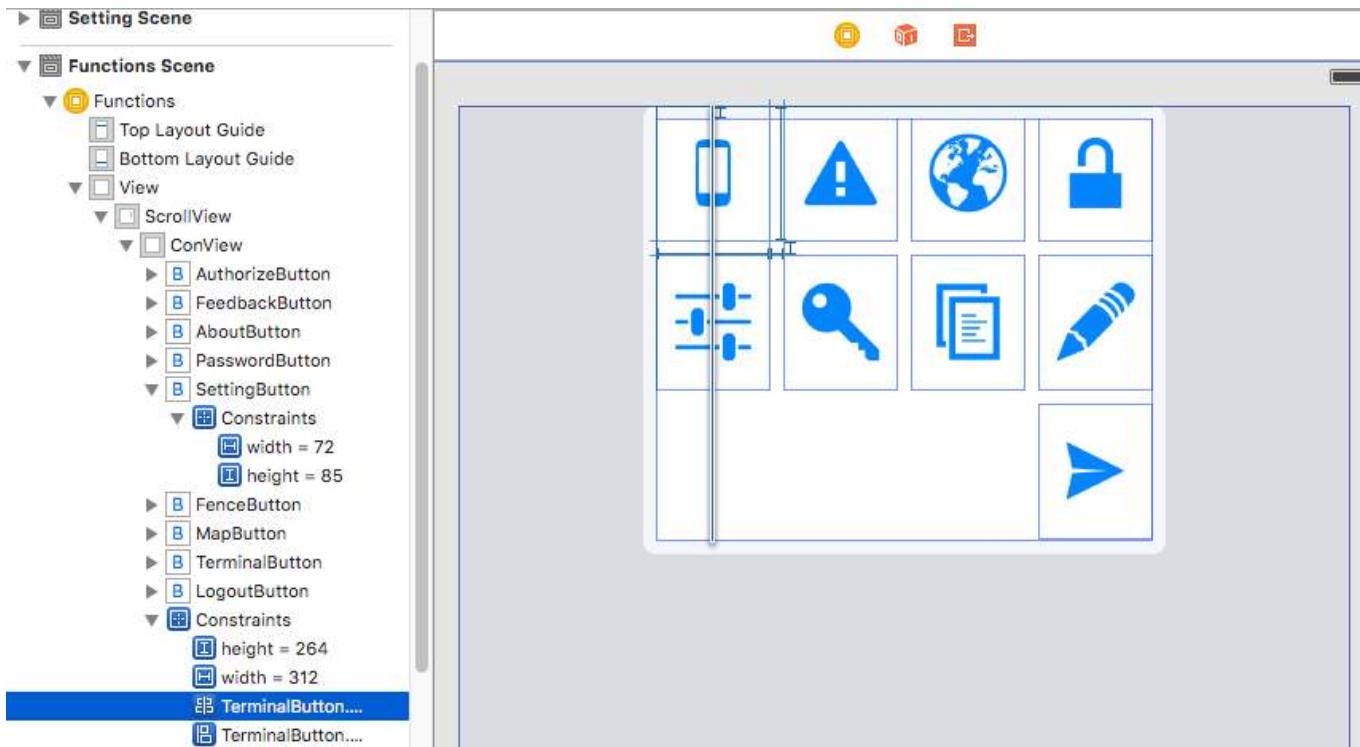
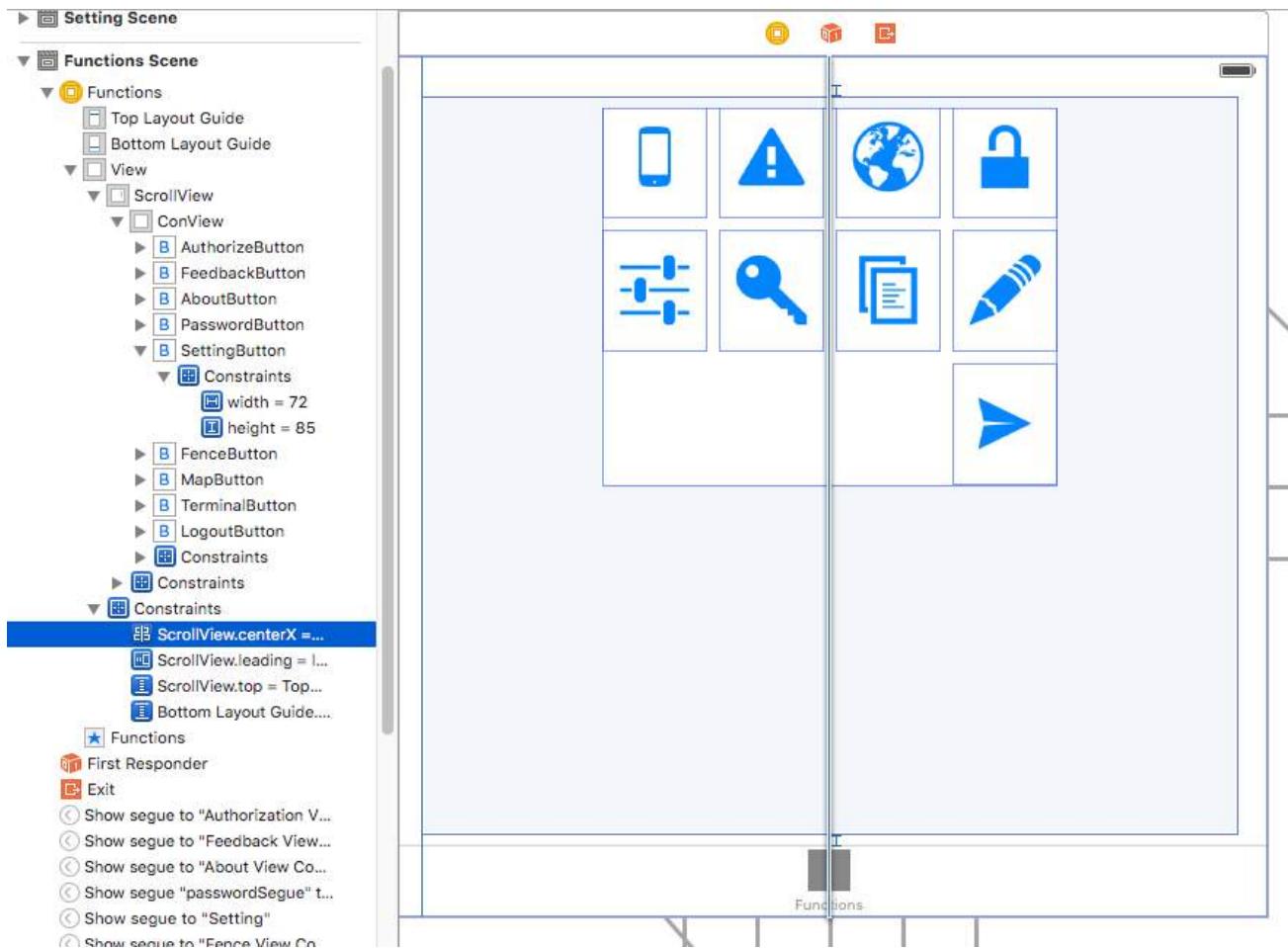
在组件上右键，可以弹出属性和操作菜单：



在右键菜单中可以方便地观察和管理 Segues:



导览器中的属性与故事板上的图示是同步变化的，这对于调整对齐约束尤为有帮助：

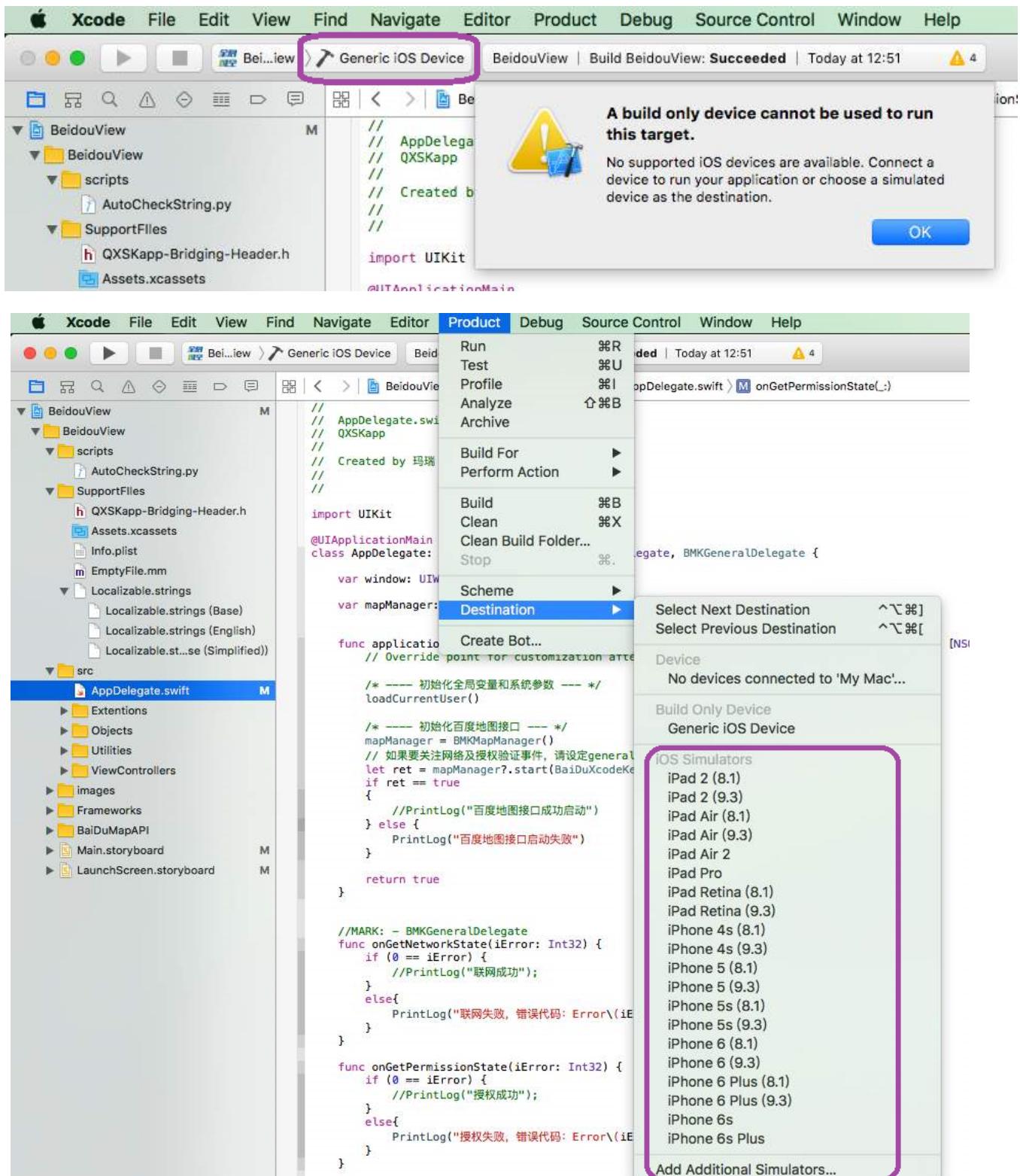


5.7 调试

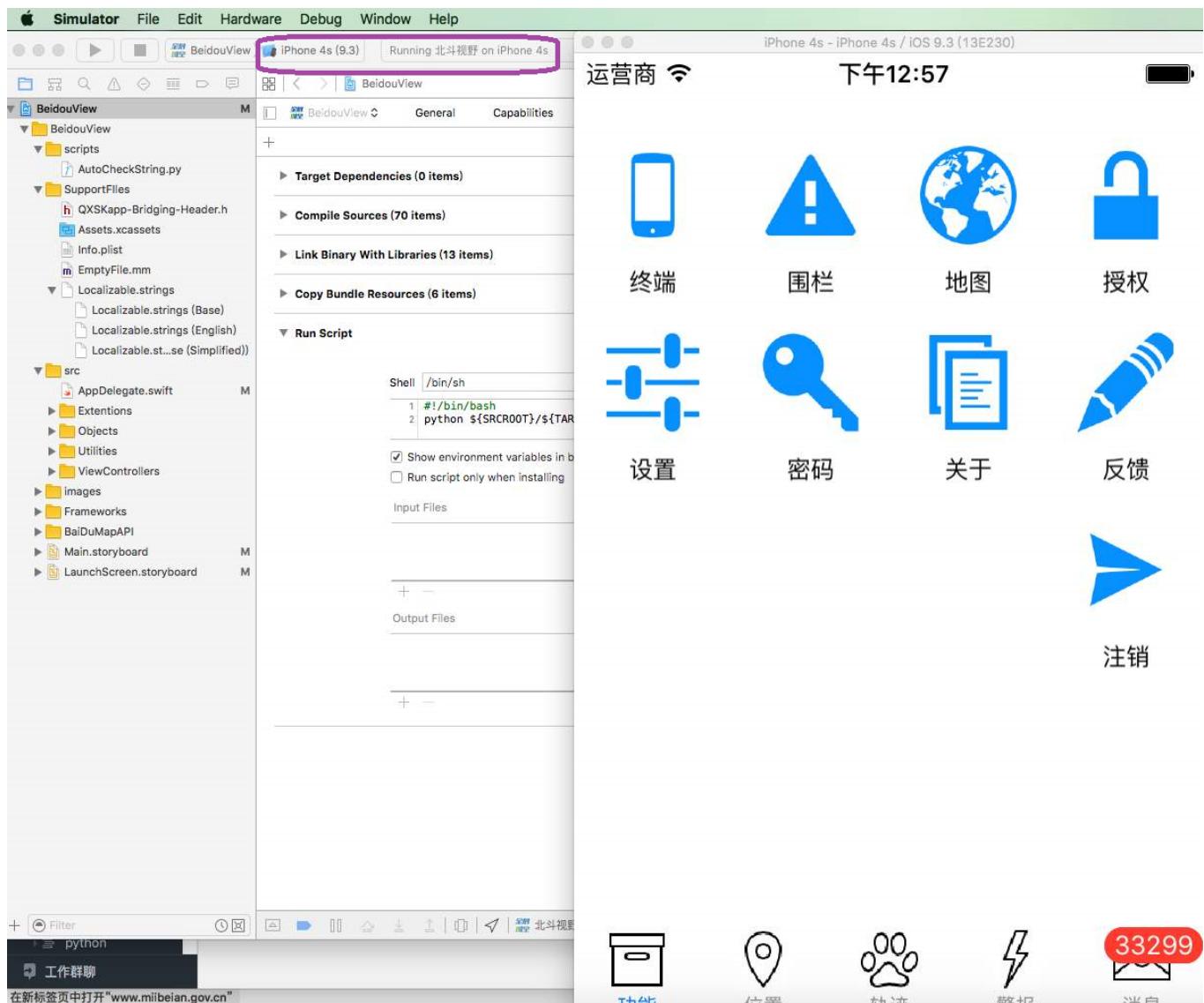
5.7.1 模拟器

5.7.1.1 运行中的模拟器

在 Xcode 中调试应用时会自动打开模拟器：



当选择模拟器型号后，应用运行在相应的界面尺寸里：



5.7.1.2 切换模拟器

点击当前模拟器型号名，则弹出可选模拟器列表，开发者可以随时切换模拟器：



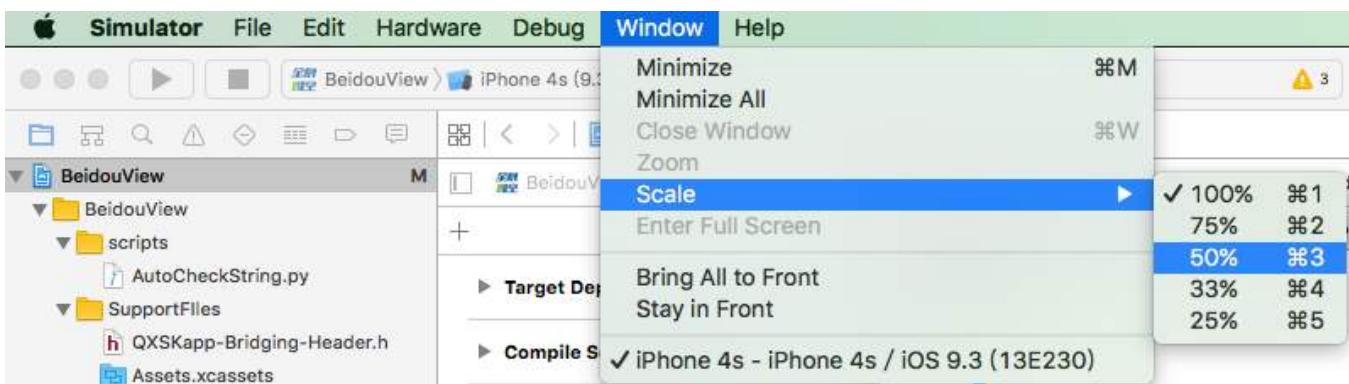
5.7.1.3 手动打开模拟器

若要手动打开模拟器，则利用菜单项“Xcode → Open Developer Tool → Simulator”：



5.7.1.4 缩放模拟器

模拟器很大只，会铺满或者超出屏幕，选择菜单项"Window->Scale"可以调整模拟器的缩放：



5.7.2 截屏

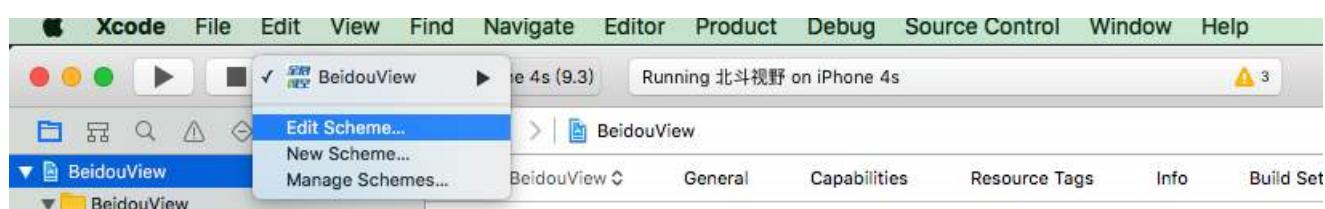
cmd+shift+3：对整个屏幕进行截图；

cmd+shift+4：对自行选择的区域进行截图；

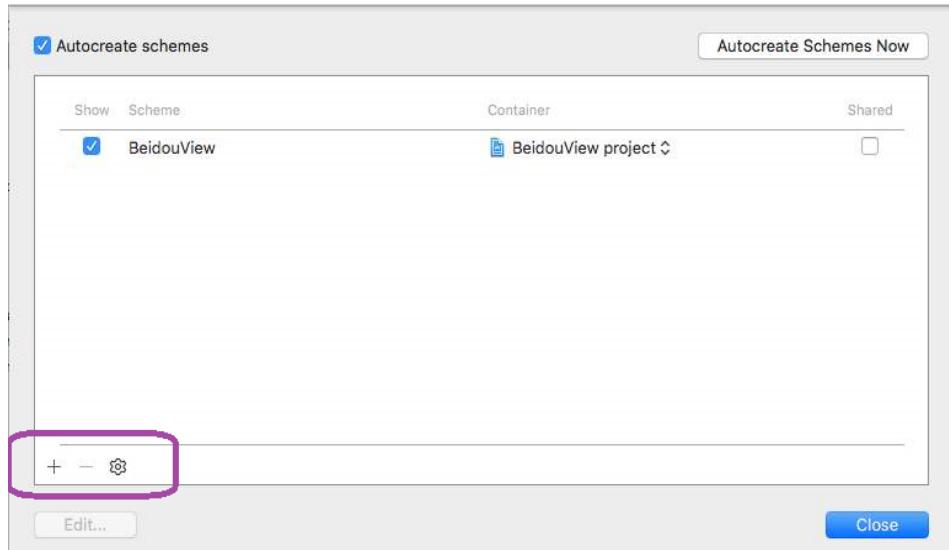
cmd+shift+4+space（空格键）：对选定的某个应用程序界面窗口进行截图。

5.7.3 管理模式 (Schema)

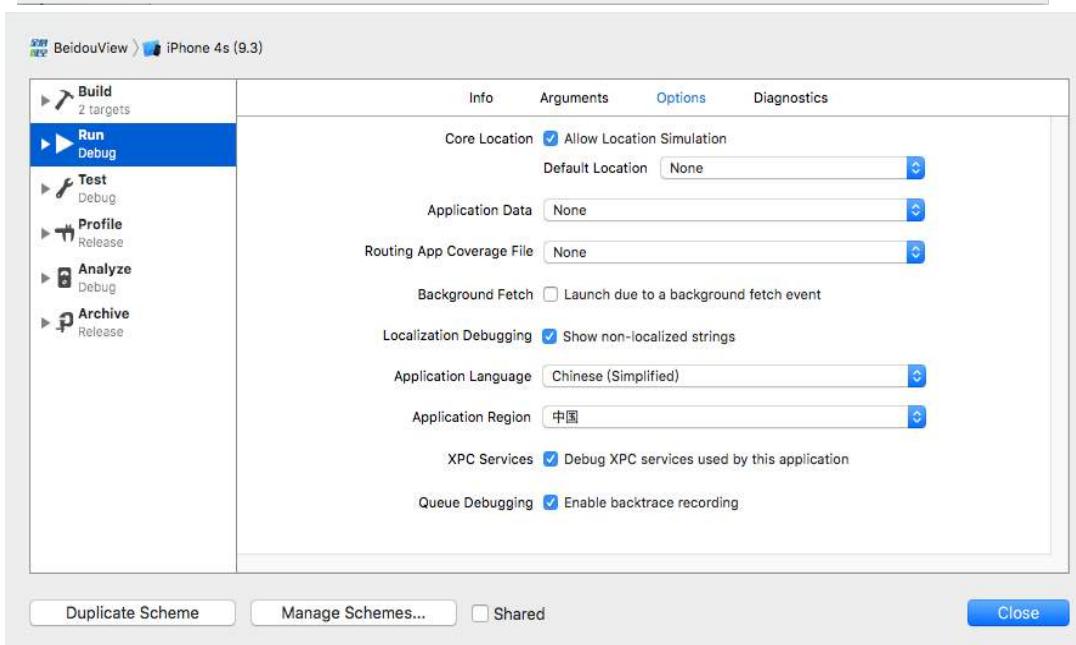
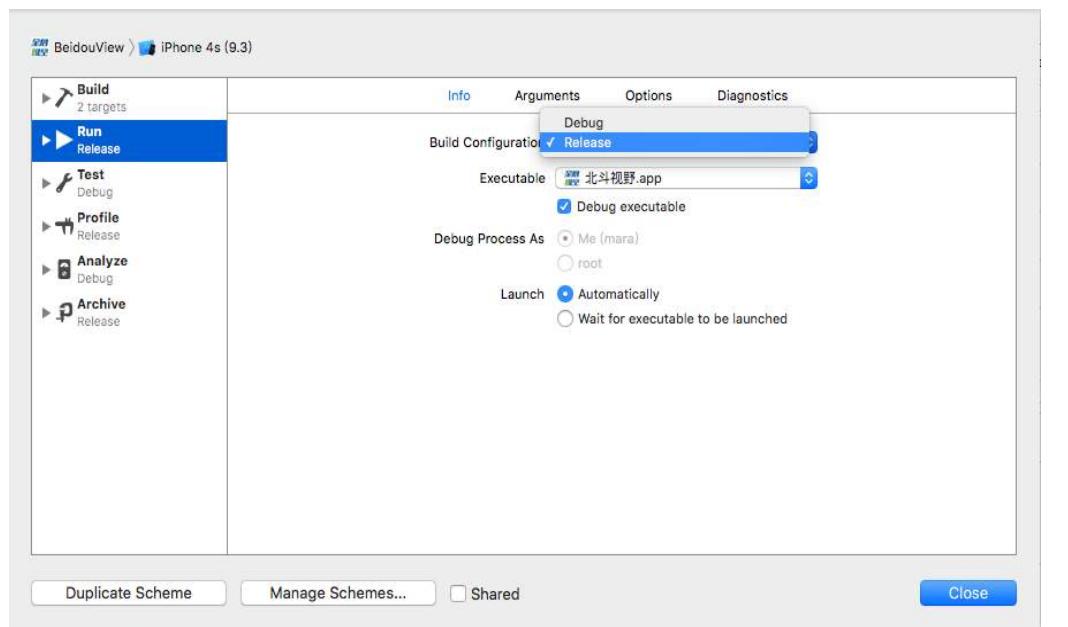
点击项目名，则弹出的模式的操作菜单：



选择“Manage Schemas...”，则显示模式管理界面，可以增删改模式：



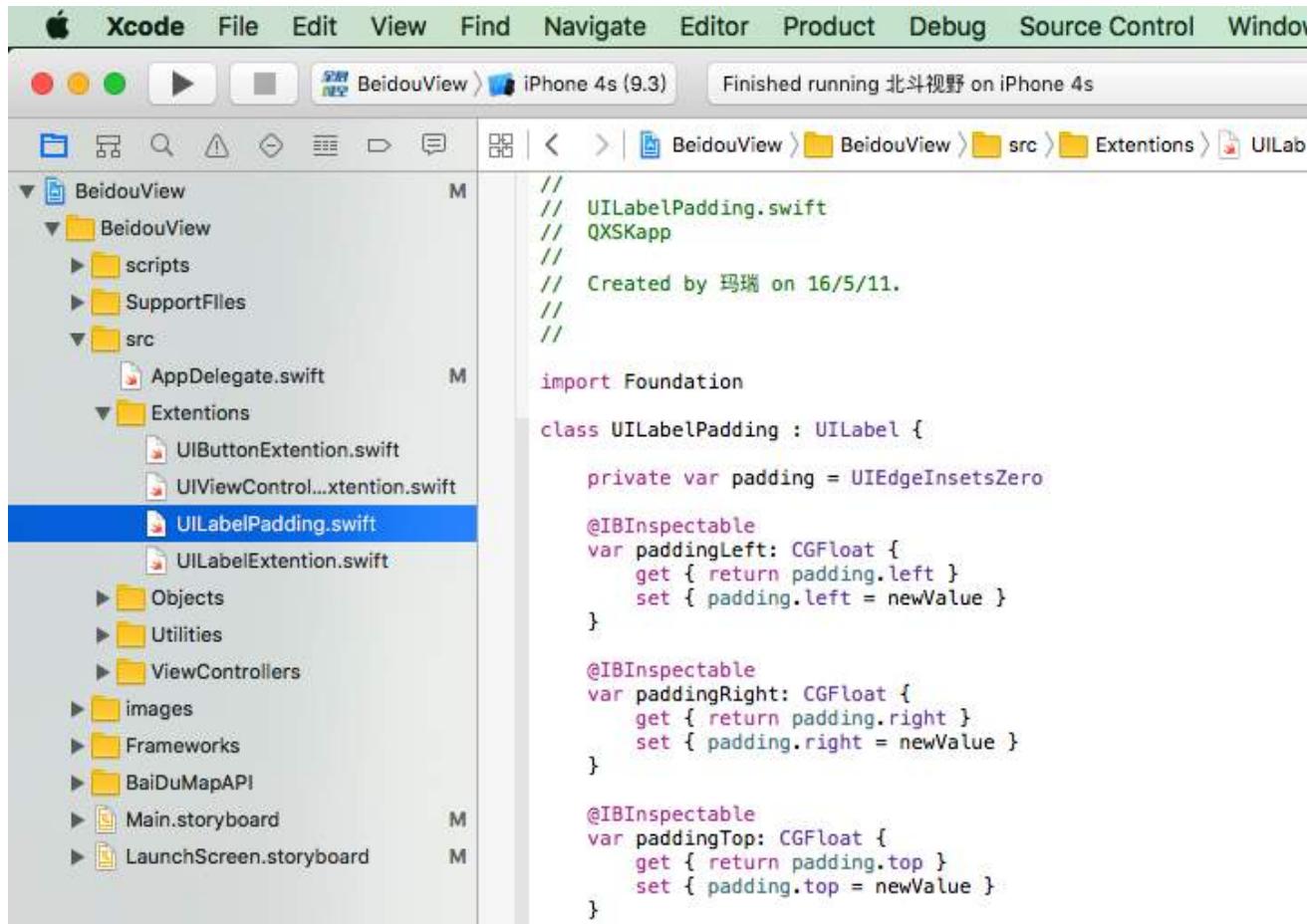
选择“Edit Schema”，则显示模式设置界面：



5.8 Swift 编码

5.8.1 定制组件

如果标准组件不满足需求，则可以扩展组件（如标签、按钮、视图）的定义，定制自己的属性和操作：



```

// 
// UILabelPadding.swift
// QXSKapp
//
// Created by 玛瑞 on 16/5/11.
//
//

import Foundation

class UILabelPadding : UILabel {

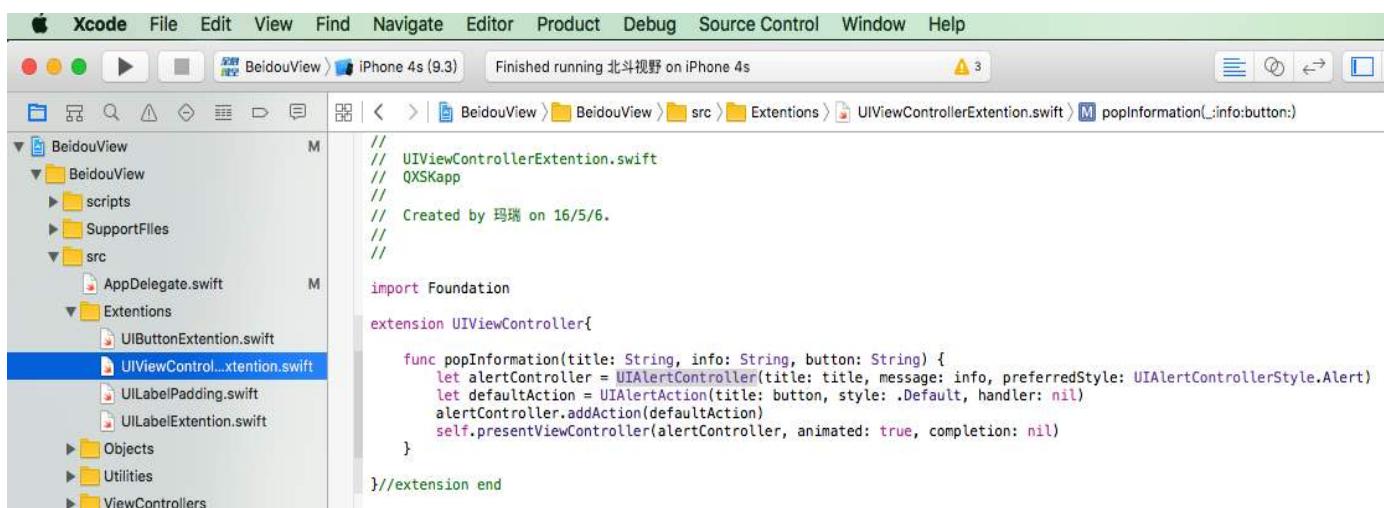
    private var padding = UIEdgeInsetsZero

    @IBInspectable
    var paddingLeft: CGFloat {
        get { return padding.left }
        set { padding.left = newValue }
    }

    @IBInspectable
    var paddingRight: CGFloat {
        get { return padding.right }
        set { padding.right = newValue }
    }

    @IBInspectable
    var paddingTop: CGFloat {
        get { return padding.top }
        set { padding.top = newValue }
    }
}

```



```

// 
// UIViewControllerExtention.swift
// QXSKapp
//
// Created by 玛瑞 on 16/5/6.
//
//

import Foundation

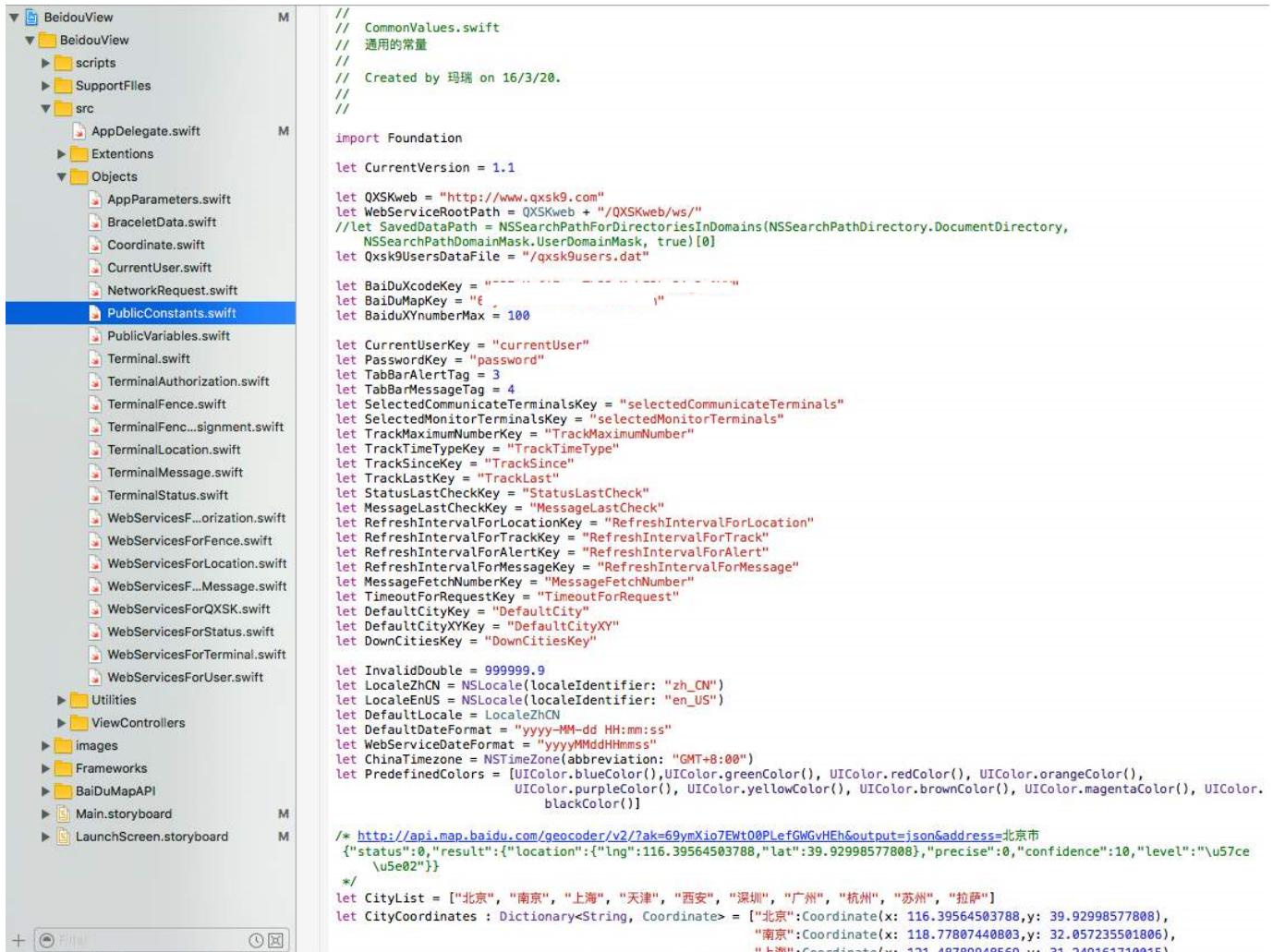
extension UIViewController{

    func popInformation(title: String, info: String, button: String) {
        let alertController = UIAlertController(title: title, message: info, preferredStyle: UIAlertControllerStyle.Alert)
        let defaultAction = UIAlertAction(title: button, style: .Default, handler: nil)
        alertController.addAction(defaultAction)
        self.presentViewController(alertController, animated: true, completion: nil)
    }
}

```

5.8.2 应用的常量

文件“PublicConstants.swift”中包含应用中的常量：



```

// CommonValues.swift
// 通用的常量
//
// Created by 玛瑞 on 16/3/20.
//
//

import Foundation

let CurrentVersion = 1.1

let QXSKweb = "http://www.qxsk9.com"
let WebServiceRootPath = QXSKweb + "/QXSKweb/ws/"
//let SavedDataPath = NSSearchPathForDirectoriesInDomains(NSSearchPathDomainMask.UserDomainMask, true)[0]
let Qxsk9UsersDataFile = "/qxsk9users.dat"

let BaiduXcodeKey = "-----"
let BaiduMapKey = "-----"
let BaiduXYnumberMax = 100

let CurrentUserKey = "currentUser"
let PasswordKey = "password"
let TabBarAlertTag = 3
let TabBarMessageTag = 4
let SelectedCommunicateTerminalsKey = "selectedCommunicateTerminals"
let SelectedMonitorTerminalsKey = "selectedMonitorTerminals"
let TrackMaximumNumberKey = "TrackMaximumNumber"
let TrackTimeTypeKey = "TrackTimeType"
let TrackSinceKey = "TrackSince"
let TrackLastKey = "TrackLast"
let StatusLastCheckKey = "StatusLastCheck"
let MessageLastCheckKey = "MessageLastCheck"
let RefreshIntervalForLocationKey = "RefreshIntervalForLocation"
let RefreshIntervalForTrackKey = "RefreshIntervalForTrack"
let RefreshIntervalForAlertKey = "RefreshIntervalForAlert"
let RefreshIntervalForMessageKey = "RefreshIntervalForMessage"
let MessageFetchNumberKey = "MessageFetchNumber"
let TimeoutForRequestKey = "TimeoutForRequest"
let DefaultCityKey = "DefaultCity"
let DefaultCityXYKey = "DefaultCityXY"
let DownCitiesKey = "DownCitiesKey"

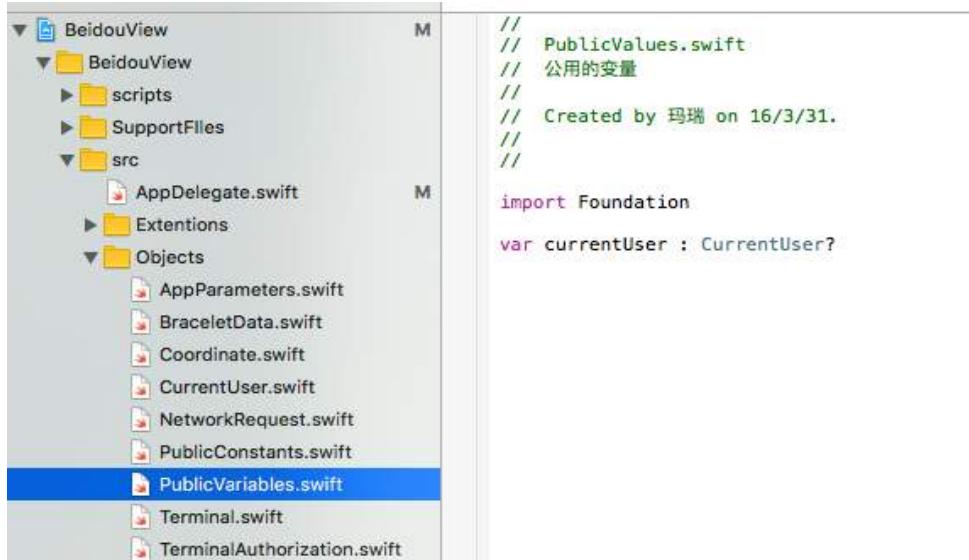
let InvalidDouble = 999999.9
let LocaleZhCN = NSLocale(localeIdentifier: "zh_CN")
let LocaleEnUS = NSLocale(localeIdentifier: "en_US")
let DefaultLocale = LocaleZhCN
let DefaultDateFormat = "yyyy-MM-dd HH:mm:ss"
let WebServiceDateFormat = "yyyyMMddHHmmss"
let ChinaTimezone = NSTimeZone(abbreviation: "GMT+0:00")
let PredefinedColors = [UIColor.blueColor(), UIColor.greenColor(), UIColor.redColor(), UIColor.orangeColor(),
    UIColor.purpleColor(), UIColor.yellowColor(), UIColor.brownColor(), UIColor.magentaColor(), UIColor.blackColor()]

/* http://api.map.baidu.com/geocoder/v2/?ak=69ymXio7EWt0PLeGvHh&output=json&address=北京市
{"status":0,"result":{"location":{"lng":116.39564503788,"lat":39.92998577808},"precise":0,"confidence":10,"level":"\u57ce\u5e02"}}
*/
let CityList = ["北京", "南京", "上海", "天津", "西安", "深圳", "广州", "杭州", "苏州", "拉萨"]
let CityCoordinates : Dictionary<String, Coordinate> = ["北京":Coordinate(x: 116.39564503788, y: 39.92998577808),
    "南京":Coordinate(x: 118.77807440803, y: 32.05235501806),
    "上海":Coordinate(x: 113.26233333333333, y: 31.230164710015)]

```

5.8.3 应用的变量

文件“PublicVariables.swift”中包含应用中变量：



```

// PublicValues.swift
// 公用的变量
//
// Created by 玛瑞 on 16/3/31.
//
//

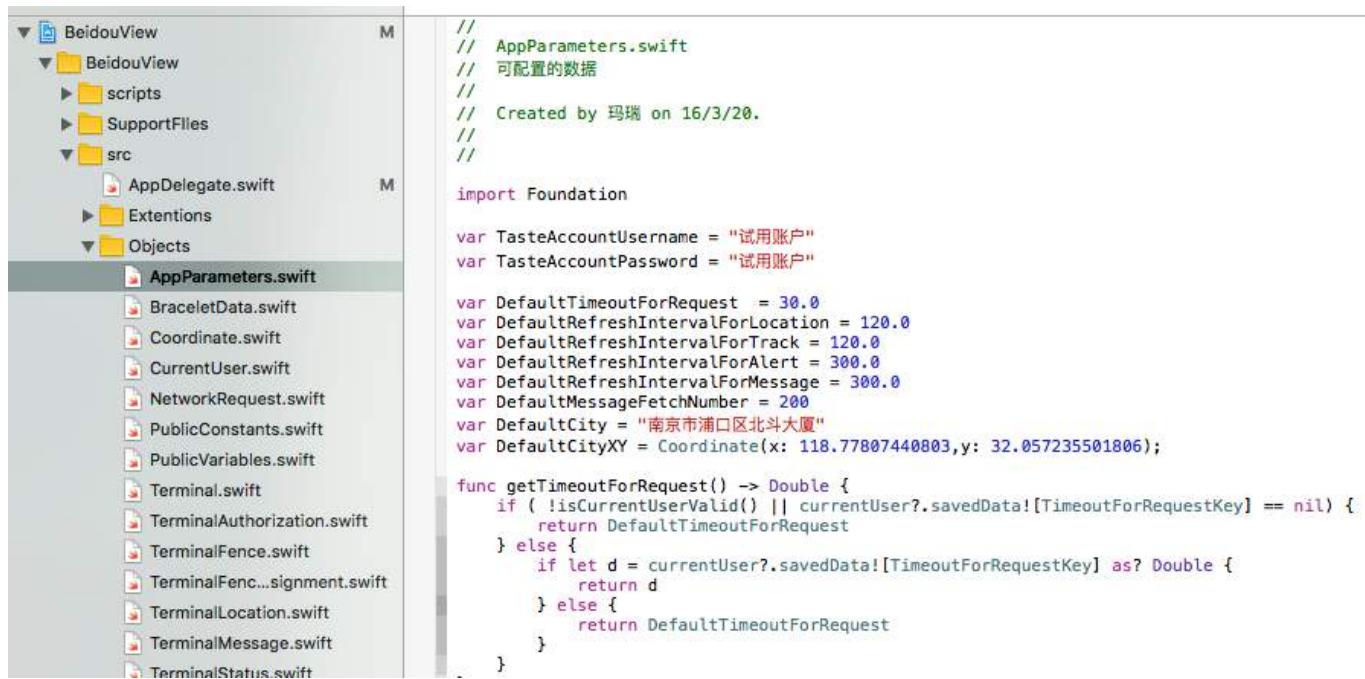
import Foundation

var currentUser : CurrentUser?

```

5.8.4 应用的参数

文件“AppParameters.swift”中包含应用中可配置的参数：



```

// AppParameters.swift
// 可配置的数据
//
// Created by 玛瑞 on 16/3/20.
//
//

import Foundation

var TasteAccountUsername = "试用账户"
var TasteAccountPassword = "试用账户"

var DefaultTimeoutForRequest = 30.0
var DefaultRefreshIntervalForLocation = 120.0
var DefaultRefreshIntervalForTrack = 120.0
var DefaultRefreshIntervalForAlert = 300.0
var DefaultRefreshIntervalForMessage = 300.0
var DefaultMessageFetchNumber = 200
var DefaultCity = "南京市浦口区北斗大厦"
var DefaultCityXY = Coordinate(x: 118.77807440803, y: 32.057235501806);

func getTimeoutForRequest() -> Double {
    if (!isValidCurrentUser() || currentUser?.savedData![TimeoutForRequestKey] == nil) {
        return DefaultTimeoutForRequest
    } else {
        if let d = currentUser?.savedData![TimeoutForRequestKey] as? Double {
            return d
        } else {
            return DefaultTimeoutForRequest
        }
    }
}

```

5.8.5 “收起键盘”

若视图中有输入域，则需要处理“键盘收起”：

```

// MARK: UITextFieldDelegate

func textFieldShouldReturn(textField: UITextField) -> Bool {
    // Hide the keyboard.
    textField.resignFirstResponder()
    return true
}

func textFieldDidBeginEditing(textField: UITextField) {

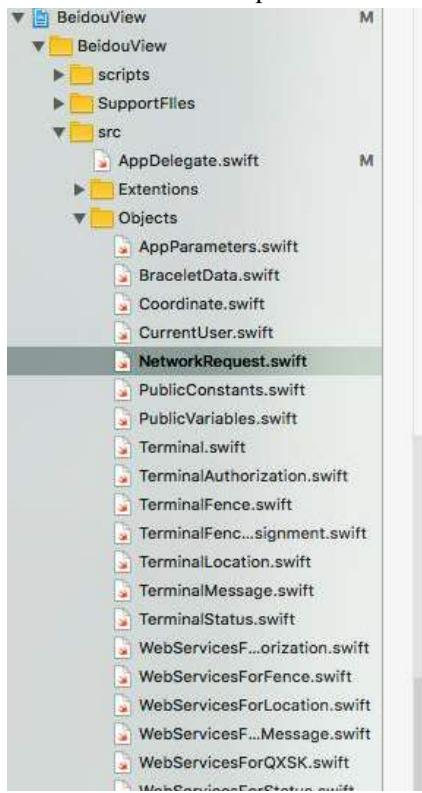
}

// 只要触摸textfield以外的地方键盘都会收起来
override func touchesEnded(touches: Set<UITouch>, withEvent event: UIEvent?) {
    emailsInput.resignFirstResponder()
    phonesInput.resignFirstResponder()
}

```

5.8.6 网络请求

文件“NetworkRequest.swift”中包含处理网络请求的数据和方法：



```

// NetworkRequest.swift
// QXSKapp
//
// Created by 玛瑞 on 16/3/24.
//
//

import Foundation

enum NetworkDataFormat : Int {
    case StringData = 1
    case IntData
    case BooleanData
    case XmlData
    case JsonData
    case DoubleData
}

let HttpReturnCodeOK = 200
let HttpReturnCodeForbidden = 403

class NetworkRequest {

    // MARK: Properties
    let method: String
    let params: Dictionary<String, String?>?
    let format: NetworkDataFormat
    let callback: (data: NSData?, response: NSURLResponse?, error: NSError?) -> Void
    let url: String
    var request: NSMutableURLRequest
    var task: NSURLSessionTask!

    // MARK: Initialization
    init(url: String, method: String,
         params: Dictionary<String, String?>?,
         acceptFormat: NetworkDataFormat,
         networkCallback: (data: NSData?, response: NSURLResponse?, error: NSError?) -> Void) {
        self.url = url
        self.request = NSMutableURLRequest(URL: NSURL(string: url)!)
    }

    private func buildRequest() {
        //PrintLog("url: \(url)")
        if self.method == "GET" {
            var tmpurl = url
            if ( self.params != nil && self.params!.count > 0 ) {
                tmpurl = url + "?" + NetworkRequest.buildParams(self.params!)
                //PrintLog("parameters: \(self.params!)")
            }
            self.request = NSMutableURLRequest(URL: NSURL(string: tmpurl)!)
        } else if method == "POST" {
            request.addValue("application/x-www-form-urlencoded", forHTTPHeaderField: "Content-Type")
            if ( self.params != nil && self.params!.count > 0 ) {
                request.HTTPBody = NetworkRequest.buildParams(params!).dataUsingEncoding(NSUTF8StringEncoding)
            }
        }
        request.HTTPMethod = self.method
        // request.addValue("application/json", forHTTPHeaderField: "Content-Type")

        switch self.format {
        case .XmlData:
            request.addValue("application/xml", forHTTPHeaderField: "Accept")
        case .JsonData:
            request.addValue("application/json", forHTTPHeaderField: "Accept")
        default:
            request.addValue("text/plain", forHTTPHeaderField: "Accept")
        }
    }

    private func fireTask() {
        let config = NSURLSessionConfiguration.defaultSessionConfiguration()
        config.timeoutIntervalForRequest = getTimeoutForRequest()
        let session = NSURLSession(configuration : config)
        task = session.dataTaskWithRequest(request,
                                           completionHandler: { (data, response, error) -> Void in
                                                self.handleResponse(data, response: response, error: error)
                                            })
        task.resume()
    }
}

```

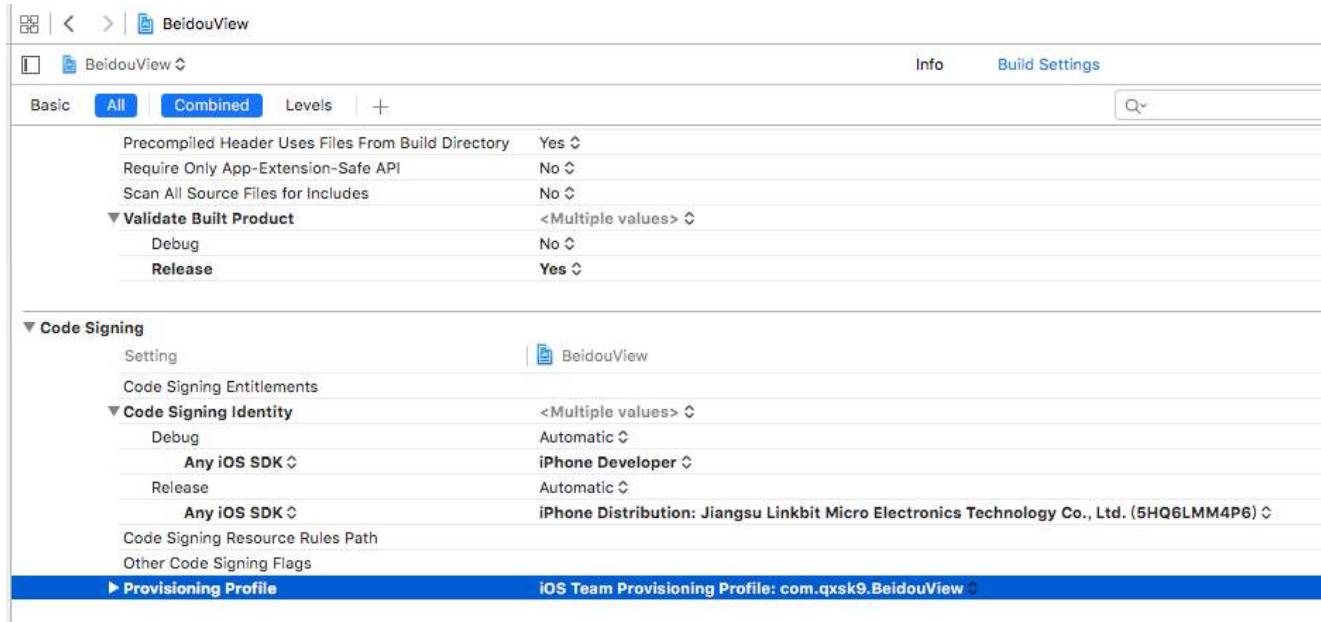
5.9 发布 iOS 应用

5.9.1 安装发布证书

将应用发布到苹果商店需要申请和安装发布证书。由于本应用账户已过期，无法截图步骤~请参考下文：

<https://www.jianshu.com/p/3c104c1150cd>

<https://www.cnblogs.com/yujinzhong/p/5800111.html>



5.9.2 在 iTunes Connect 中添加版本

在 iTunes Connect 中需要为每一种尺寸提交截图：

iTunes Connect 我的 App ▾

Jiangsu Linkbit Micro Electronics Technology Co., Ltd. 任珊虹 ▾ | ?

App Store 功能 TestFlight 活动

APP STORE 信息

App 信息

价格与销售范围

iOS APP

● 1.0 准备提交

版本或平台

+ 添加新版本

iOS App 1.0

● 准备提交

存储 提交以供审核

版本信息 简体中文 ▾ ?

App 预览和屏幕快照 ? 每种尺寸都得提供截图

4.7 英寸 5.5 英寸 4 英寸 3.5 英寸 iPad iPad Pro

7:31 PM 国栏 清除 确定

单击一点锁定圆心，长按另一点锁定半径

半径 (米) 1261.6822 检查

圆心 X 118.78087... Y 32.069176...

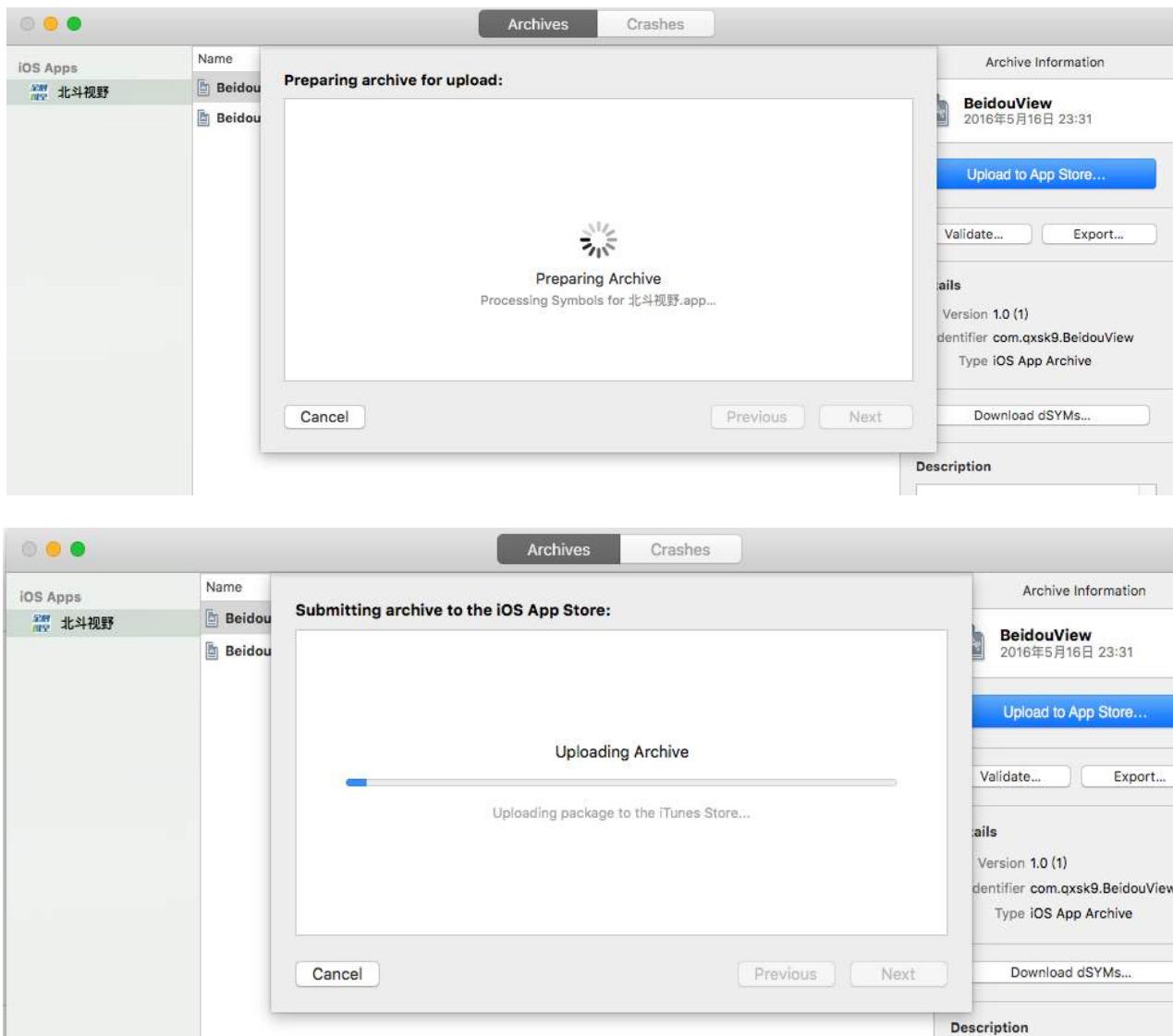
地图显示了两个重叠的圆圈，标注了圆心坐标。

编辑应用的信息，如新版本的更新：

5.9.3 上传应用

当所有条件满足后（参见上述两篇网文），选择菜单项“Product -> Archive”，开始发布流程：





5.9.4 处理反馈

在 iTunes Connect 上关注新版本的审核状态，苹果也会把审核结论发送到开发者的邮箱里。收到如下 email 意味着版本有问题：

New_message_from_App_Review_for_北斗视野 保存

发件人： iTunes Store <no_reply@email.apple.com> [添加到通讯录](#) [原信下载](#)
收件人： rshmara@sina.com
日期： 2016-05-22 02:57

邮件里的图片看不到？请开启[超文本新窗口](#)阅读

Dear 珊虹,

We've sent you a new message about your app, 北斗视野 (1113952607).
To view or reply to the message, go to Resolution Center on iTunes Connect.

Regards,
App Store Review

5.9.5 搜索 App Store

确保应用可以在苹果商店里可以被搜索到：

北斗视野

Jiangsu Linkbit Micro...

+ 打开

北斗视野 [4+]
Jiangsu Linkbit Micro Electronics Technology Co., Ltd. >

+ 打开

详情 评论 相关

内容提要

北斗视野是全息时空北斗运营服务平台的手机客户端。
全息时空北斗运营服务平台支持北斗数据网与公共数据网的双向全联通：
1) 北斗数据终端通过北斗数据通道向数据中心报告位置信息、发出SOS警报、收发北斗短消息。
2) 普通数据终端通过公共数据网利用网络服务向数据中心报告位置信息、发出SOS警报、收发手机短信、传输业务数据。... [更多](#)

信息

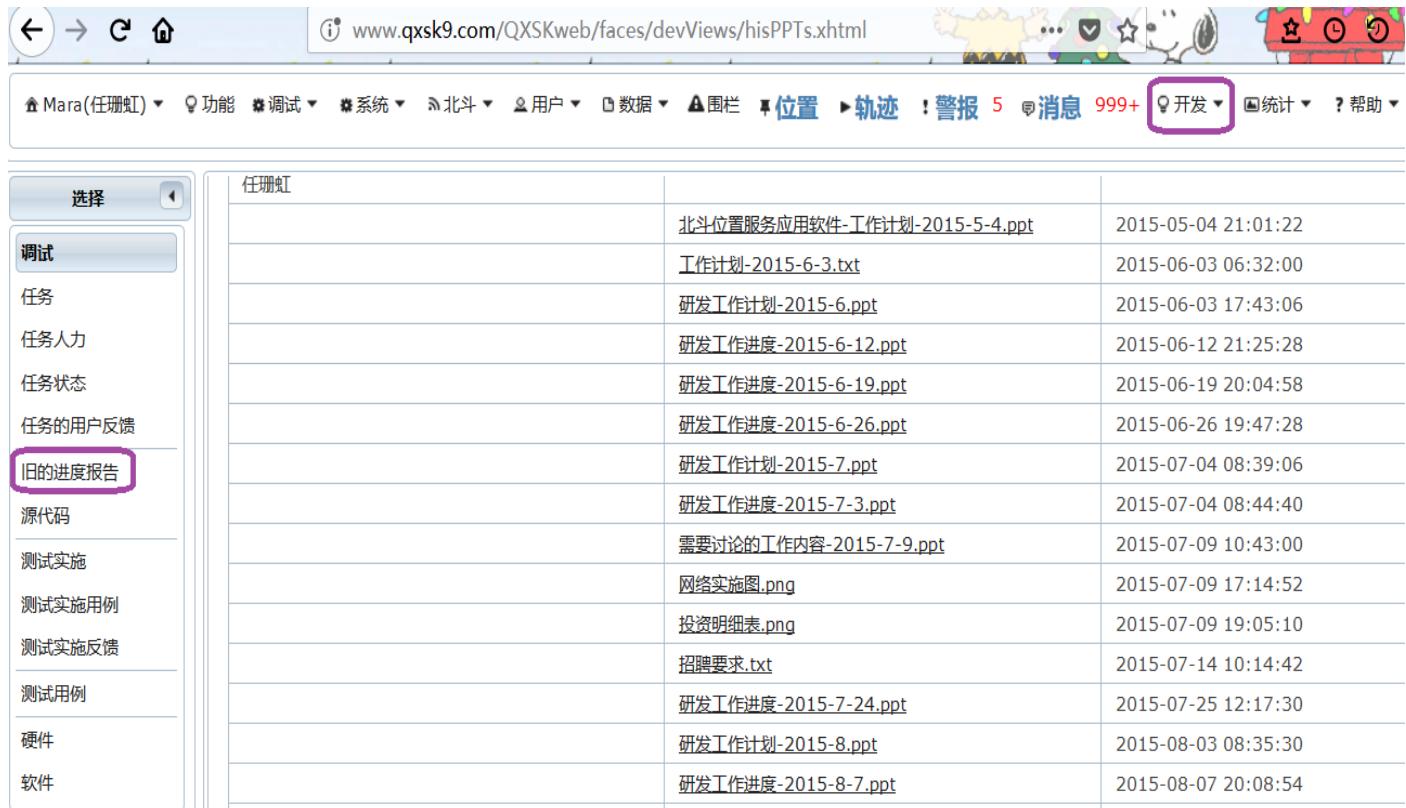
开发商	Jiangsu Linkbit Micro Electronics Technology Co., Ltd.
类别	导航
更新日期	2016年5月22日
版本	1.0
大小	12.3 MB
评级	限4岁以上
家人共享	可使用
兼容性	需要 iOS 9.3 或更高版本。与 iPhone、iPad 和 iPod touch 兼容。
语言	简体中文, 英语

附录 A 其它与开发相关的信息

系统在开发过程中碰到的问题及其解决方法，也可以通过以下信息窥知一二。

A.1 进度报告

- 1) 以管理员账户登录系统的网页客户端
- 2) 选择菜单项“开发-旧的进度报告”



The screenshot shows a web browser window for www.qxsk9.com/QXSKweb/faces/devViews/hisPPTs.xhtml. The menu bar includes '功能', '调试', '系统', '北斗', '用户', '数据', '围栏', '位置', '轨迹', '警报 5', '消息 999+', '开发' (which is highlighted with a red box), '统计', and '帮助'. On the left, a sidebar menu lists '调试' (selected), '任务', '任务人力', '任务状态', '任务的用户反馈', '旧的进度报告' (highlighted with a red box), '源代码', '测试实施', '测试实施用例', '测试实施反馈', '测试用例', '硬件', and '软件'. The main content area displays a table titled '任珊虹' with 15 rows of progress reports. The table has two columns: '文件名' and '时间'. The data is as follows:

文件名	时间
北斗位置服务应用软件-工作计划-2015-5-4.ppt	2015-05-04 21:01:22
工作计划-2015-6-3.txt	2015-06-03 06:32:00
研发工作计划-2015-6.ppt	2015-06-03 17:43:06
研发工作进度-2015-6-12.ppt	2015-06-12 21:25:28
研发工作进度-2015-6-19.ppt	2015-06-19 20:04:58
研发工作进度-2015-6-26.ppt	2015-06-26 19:47:28
研发工作计划-2015-7.ppt	2015-07-04 08:39:06
研发工作进度-2015-7-3.ppt	2015-07-04 08:44:40
需要讨论的工作内容-2015-7-9.ppt	2015-07-09 10:43:00
网络实施图.png	2015-07-09 17:14:52
投资明细表.png	2015-07-09 19:05:10
招聘要求.txt	2015-07-14 10:14:42
研发工作进度-2015-7-24.ppt	2015-07-25 12:17:30
研发工作计划-2015-8.ppt	2015-08-03 08:35:30
研发工作进度-2015-8-7.ppt	2015-08-07 20:08:54

3) 这些进度报告包含了开发过程中技术选型、问题讨论、团队建设的内容。

凌比特-研发进度报告-2015-09-17

工作内容	说明	当前进度
研发方向和总体框架	基于北斗定位服务的各类应用 占领互补市场而非与已存在的成熟产品竞争	100%
资质认证	可行性报告、提交申请	100%
原型研发	数据存储层的设计与实现 业务逻辑的设计与实现	100%
	集团用户的权限模型与使用视图	100%
	个人用户的权限模型与使用视图	100%
	网页客户端	100%
	手机客户端应用	30%
数据链路	北斗数据的注入、分发、与应用 公网通道：网络服务、短信收发	80% 90%
产品研发	项目XXXX的调研、立项、与研发 项目YYYY的调研、立项、与研发	80% 30%
数据中心	筹备、拨款、设计、实施	40% 2

“北斗位置服务应用软件” 2015年5月的工作计划

工作内容	工作重点和难点
搭建环境： 1) 北斗终端与指挥机 2) 服务器: Linux + MySql 3) 工作站: iOS + NetBeans	1) 连通终端与指挥机 2) 读取指挥机的输出
以追踪旅游区的人员和车辆为需求示例，设计位置服务应用的原型	简化需求，确定基本功能
设计原型的数据表	用户模型、节点类型、节点属性、位置信息
北斗数据入库：产生示例数据	根据指挥机数据协议，写入数据库
编写原型的MVC代码	1) 实践各种Java框架的最新版本 2) 取舍框架：Spring + JPA (+ Hibernate) ?
修正系统框架	

团队建设-网管的招聘要求

职位: 网络管理员

职责: 维护公司的网络系统，保障网络和主机的运行：

- 1) 设计并实现系统安全管理方案
- 2) 设计并实现系统备份方案
- 3) 设计并实现机房管理方案。
- 4) 配合其它部门进行网络建设，提出规划、标准。

基本要求:

- 1) 无学历要求，无性别要求，无加班要求，只要能够把事情做顺溜了。
- 2) 曾经设计并实现过网络管理方案，能清晰描述细节。
- 3) 熟悉Linux和Windows的网管工具，能自行设计和实现网管工具。
- 4) 具有文档编写能力。

加分条件:

- 1) 在网络上共享过自己的经验。
- 2) 快速阅读英语原文资料，快速学习技术并且实际应用。
- 3) 会java或者c编程
- 4) 积极乐观。

面试方式:

- 1) 针对公司的网络现状，让其描述自己的网络管理方案，包括安全、备份、机房等。
- 2) 询问其会什么编程语言、编写过什么代码、解决过什么问题

目前应聘者的面试评价

职位	姓名	知识	技能	经验	认真	诚恳	好学	表达	目前月薪(税前)	结论
平台		8	7	6	6	6	6	5	6000	谈崩
网管		5	4	4	8	8	9	7	4000	落选
		6	7	7	9	9	9	9	4200	选定
		5	5	5	7	7	5	5	5000	落选
		6	6	7	7	5	6	7	兼职	落选
		8	9	9	8	8	7	8	9000	落选
App		3	3	3	8	8	7	5	创业	待定

A.2 功能“开发”

- 1) 以管理员账户登录系统的网页客户端
- 2) 选择菜单项“开发”
- 3) 这些数据可以用来记录开发过程的任务、人力、状态、和问题追踪。（目前并没有实际应用）

A.2 功能“测试”

- 1) 以管理员账户登录系统的网页客户端
- 2) 选择菜单项“开发”
- 3) 这些数据可以用来记录测试过程的资源、用例、实施、和问题追踪。（有过实际应用）

A.3 用户反馈

1) 以任意账户登录系统的网页客户端

2) 选择菜单项“帮助-用户反馈”

数据编号	用户反馈的对象	用户反馈类型	标题	报告者	报告时间	用户反馈状态	响应时间	响应者	用户反馈的来源	解决的版本
441	安卓客户端	改进建议	"轨迹"图增加颜色	任珊虹	2017-12-13 08:46	已解决	2017-12-22 16:39:43	任珊虹	网页客户端	安卓客户端-2.000
440	安卓客户端	功能需求	"位置"图上区分终端状态	任珊虹	2017-12-13 08:35	已解决	2017-12-22 16:39:54	任珊虹	网页客户端	安卓客户端-2.000
439	安卓客户端	功能需求	对称的数据加密算法	任珊虹	2017-12-08 10:44	已解决	2018-02-28 19:56:13	任珊虹	网页客户端	安卓客户端-2.100
438	安卓客户端	功能需求	向服务器提供日志数据	任珊虹	2017-12-08 10:37	已解决	2017-12-22 16:40:07	任珊虹	网页客户端	安卓客户端-2.000
437	安卓客户端	功能需求	解析数据时注意标签可能被强	任珊虹	2017-12-08 10:27	无需解决	2017-12-09 21:58:22	任珊虹	网页客户端	-
436	平台服务器	功能需求	系统日志写入数据库	任珊虹	2017-11-29 20:58	已解决	2017-12-22 16:41:02	任珊虹	网页客户端	平台服务器-5.000
435	接口	改进建议	Web Service和WebSocket	任珊虹	2017-11-28 17:19	已解决	2017-12-22 16:42:10	任珊虹	网页客户端	平台服务器-5.000
434	平台服务器	性能优化	读写分离、实时同步、缓存、	任珊虹	2017-11-24 19:29	已解决	2017-12-22 16:42:19	任珊虹	网页客户端	平台服务器-5.000
433	平台服务器	功能需求	终端状态按重要性排序	任珊虹	2017-11-24 18:40	已解决	2017-12-22 16:42:29	任珊虹	网页客户端	平台服务器-5.000
432	数据	功能需求	衍生数据：步长、速度、和行	任珊虹	2017-11-24 09:11	以后解决	2017-12-29 11:31:47	任珊虹	网页客户端	-
431	安卓客户端	功能需求	采用okhttp库实现数据实时传	任珊虹	2017-11-23 11:03	已解决	2017-12-22 16:43:31	任珊虹	网页客户端	安卓客户端-2.000
430	安卓客户端	功能需求	本地数据全部存入Sqlite	任珊虹	2017-11-23 10:19	已解决	2017-12-22 16:43:43	任珊虹	网页客户端	安卓客户端-2.000
429	平台服务器	性能优化	数据表中的布尔字段均改为非	任珊虹	2017-11-23 10:01	已解决	2017-12-22 16:43:56	任珊虹	网页客户端	平台服务器-5.000
428	数据	功能需求	数据：相对距离	任珊虹	2017-11-22 09:02	以后解决	2017-12-29 11:31:54	任珊虹	网页客户端	-
427	接口	功能需求	为手机客户端实现新的webS	任珊虹	2017-11-22 07:49	已解决	2017-12-22 16:44:07	任珊虹	网页客户端	平台服务器-5.000

3) 这些用户反馈包含了系统开发的问题、需求、改进、分析。

查看 用户反馈	
数据编号	412
用户反馈的对象	数据
用户反馈类型	性能优化
标题	调整数据定义以适应数据表分区
描述	mysql分区不支持外键，且对主键有要求。对于需要分区的数据表，需要删除外键并且重新定义主键。
报告者	任珊虹
报告时间	2017-11-02 09:05:25
响应说明	在数据逻辑层（ER图）和代码持久层（JPA）不必变动，仍然保持原有定义。而是在物理层直接删除外键并且重新定义主键，用触发器保证外键数据的一致性。
响应时间	2017-11-06 17:46:28
响应者	任珊虹
用户反馈状态	已解决
用户反馈的来源	网页客户端
解决的版本	平台服务器-4.500

关闭

A.3 版本历史

- 1) 以任意账户登录系统的网页客户端
- 2) 选择菜单项“帮助-版本历史”

列表 - 版本历史					
小提示 (鼠标放这里)					
	软件	版本号	已发布	版本时间	描述
<input type="checkbox"/>	平台服务器	5.200	<input checked="" type="checkbox"/>	2018-04-06 13:38:44	修正几个问题。添加WebSocket调试功能
<input type="checkbox"/>	平台《用户手册》	4.000	<input checked="" type="checkbox"/>	2018-04-06 13:38:10	添加新功能的描述
<input type="checkbox"/>	平台安装配置手册	4.000	<input checked="" type="checkbox"/>	2018-04-04 13:01:41	加入从库事件处理步骤。并入安全管理手册
<input type="checkbox"/>	网络服务接口	2.100	<input checked="" type="checkbox"/>	2018-03-28 11:03:35	添加PBE密码参数。Token暂停支持
<input type="checkbox"/>	WebSocket接口	1.000	<input checked="" type="checkbox"/>	2018-03-27 19:03:04	描述WebSocket接口的使用方式、及每个接口的地址与数据
<input type="checkbox"/>	安卓客户端	4.100	<input checked="" type="checkbox"/>	2018-03-23 17:58:52	修正北斗伴侣的小问题。
<input type="checkbox"/>	安卓客户端《用户手册》	2.000	<input checked="" type="checkbox"/>	2018-03-23 10:22:48	描述三合一的应用：客户端、本地定位、北斗伴侣
<input type="checkbox"/>	安卓客户端	4.000	<input checked="" type="checkbox"/>	2018-03-22 22:12:51	三合一：客户端、本地定位、北斗伴侣。
<input type="checkbox"/>	安卓客户端	3.100	<input checked="" type="checkbox"/>	2018-03-01 16:08:08	调整界面；加入百度定位方式
<input type="checkbox"/>	安卓客户端	3.000	<input checked="" type="checkbox"/>	2018-02-28 19:50:22	实现手机定位终端
<input type="checkbox"/>	平台服务器	5.120	<input checked="" type="checkbox"/>	2018-02-28 19:49:45	调整位置数据的定义以适应多角度数据分析
<input type="checkbox"/>	安卓客户端	2.100	<input checked="" type="checkbox"/>	2018-01-16 13:09:43	密码以密文存储和传输。修正若干小问题。
<input type="checkbox"/>	平台服务器	5.110	<input checked="" type="checkbox"/>	2018-01-16 13:07:18	补充网页元素id以便于自动化功能测试。处理安卓客户端的密文
<input type="checkbox"/>	平台服务器	5.100	<input checked="" type="checkbox"/>	2017-12-29 11:30:29	解决重复日志和日志显示的问题
<input type="checkbox"/>	平台服务器	5.000	<input checked="" type="checkbox"/>	2017-12-22 16:35:34	明确读写分离的标准。优化数据接口。日志写入数据库，页面

- 3) 这些用户反馈包含了系统的进化过程。

	软件	版本号	已发布	版本时间	描述	记录者
<input type="checkbox"/>	平台服务器	2.300	<input checked="" type="checkbox"/>	2016-04-21 13:02:34	新增网络服务以支持手机客户端查询与操作消息和报警数据	任珊虹
<input type="checkbox"/>	平台服务器	2.230	<input checked="" type="checkbox"/>	2016-04-19 13:00:29	增加报警类型：落水报警。根据手环数据的逻辑，当收到手环数据时，触发落水报警。	任珊虹
<input type="checkbox"/>	平台服务器	2.130	<input checked="" type="checkbox"/>	2016-04-15 16:32:20	增加手环数据的接收、解析、存储、处理、和显示	任珊虹
<input type="checkbox"/>	平台服务器	2.120	<input checked="" type="checkbox"/>	2016-04-14 21:41:27	增加更多的网络服务	任珊虹
<input type="checkbox"/>	平台服务器	2.110	<input checked="" type="checkbox"/>	2016-04-12 13:01:19	增强网络服务：指明操作类型来访问终端数据	任珊虹
<input type="checkbox"/>	平台服务器	2.100	<input checked="" type="checkbox"/>	2016-04-11 13:16:09	继续完善数据权限模型：增加网络服务，并且调整功能权限。	任珊虹
<input type="checkbox"/>	平台服务器	2.000	<input checked="" type="checkbox"/>	2016-04-10 21:10:38	数据权限模型被重新设计和实现了。现在用户对终端的权限可以是以下的组合：管理、监视	任珊虹
<input type="checkbox"/>	平台服务器	1.340	<input checked="" type="checkbox"/>	2016-04-04 18:45:01	添加网络服务：修改终端的邮箱列表和手机号列表	任珊虹
<input type="checkbox"/>	平台服务器	1.330	<input checked="" type="checkbox"/>	2016-03-27 00:50:49	修正问题：由于时间函数变化而导致若干功能失效。	任珊虹
<input type="checkbox"/>	平台服务器	1.320	<input checked="" type="checkbox"/>	2016-03-15 17:37:27	修正北斗伴侣位置报告 (F1) 的时间解析错误。	任珊虹
<input type="checkbox"/>	平台服务器	1.310	<input checked="" type="checkbox"/>	2016-03-12 16:34:38	统一按世界标准时 (UTC) 处理北斗数据和保存数据。	任珊虹
<input type="checkbox"/>	平台服务器	1.300	<input checked="" type="checkbox"/>	2016-03-11 22:24:13	修正北斗短消息的构建和解析算法，保证短消息在各种模式下被正确发送和接收。	任珊虹
<input type="checkbox"/>	平台服务器	1.200	<input checked="" type="checkbox"/>	2016-03-08 14:37:24	处理“用户反馈”：65~72	任珊虹
<input type="checkbox"/>	平台服务器	1.100	<input checked="" type="checkbox"/>	2016-03-05 14:36:35	处理“用户反馈”：1~64	任珊虹
<input type="checkbox"/>	平台服务器	1.000	<input checked="" type="checkbox"/>	2015-12-12 14:35:15	系统上线	任珊虹

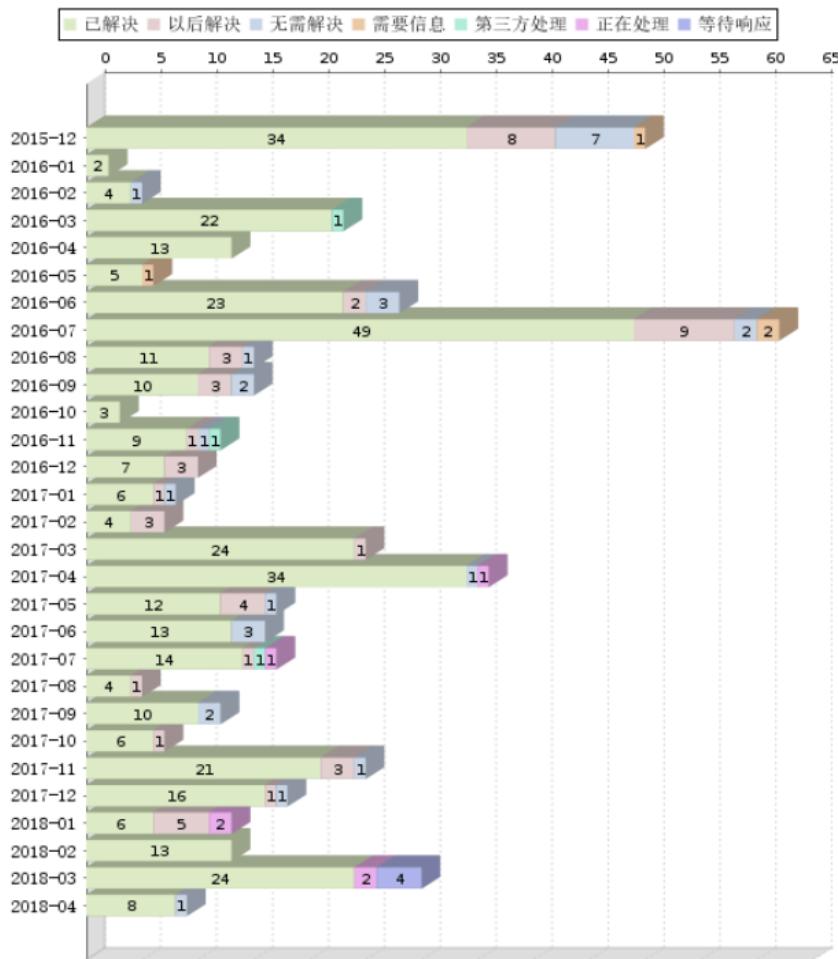
总计:105 页数:7/7

A.4 统计报告

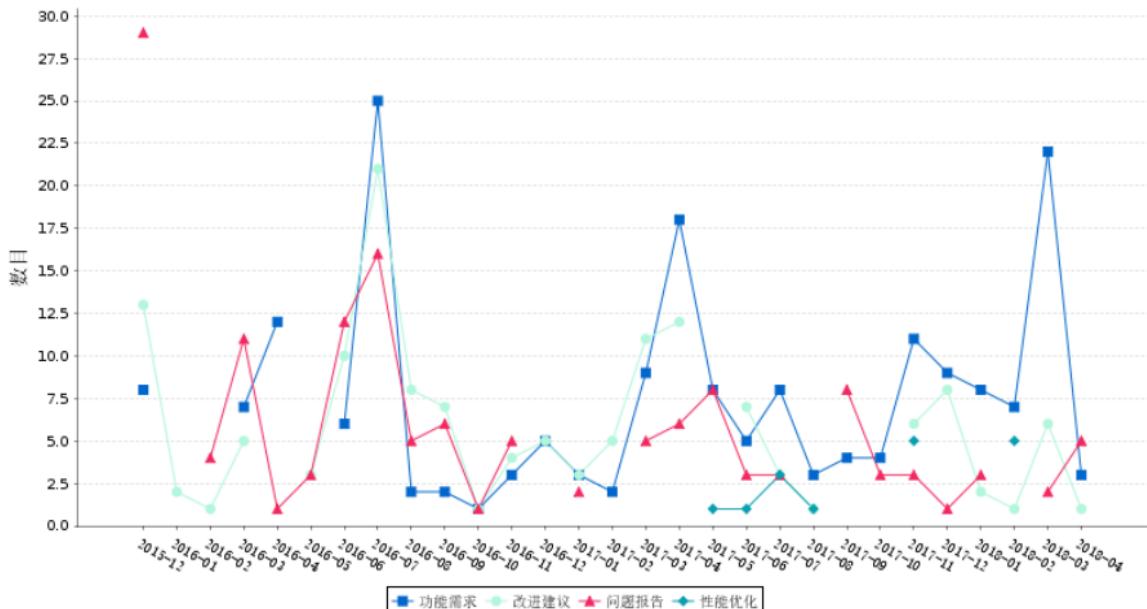
1) 以任意账户登录系统的网页客户端

2) 选择菜单项“统计 → 用户反馈”，可以从多个角度查看用户反馈的统计信息：

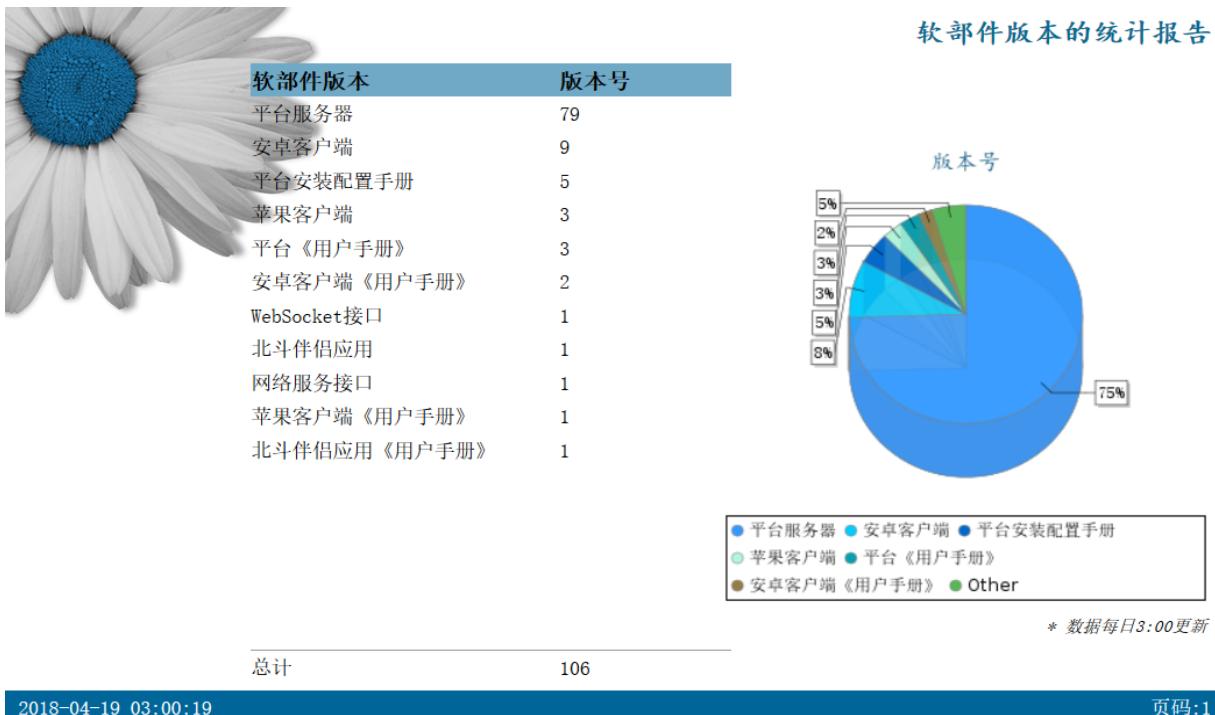
用户反馈状态的月统计报告



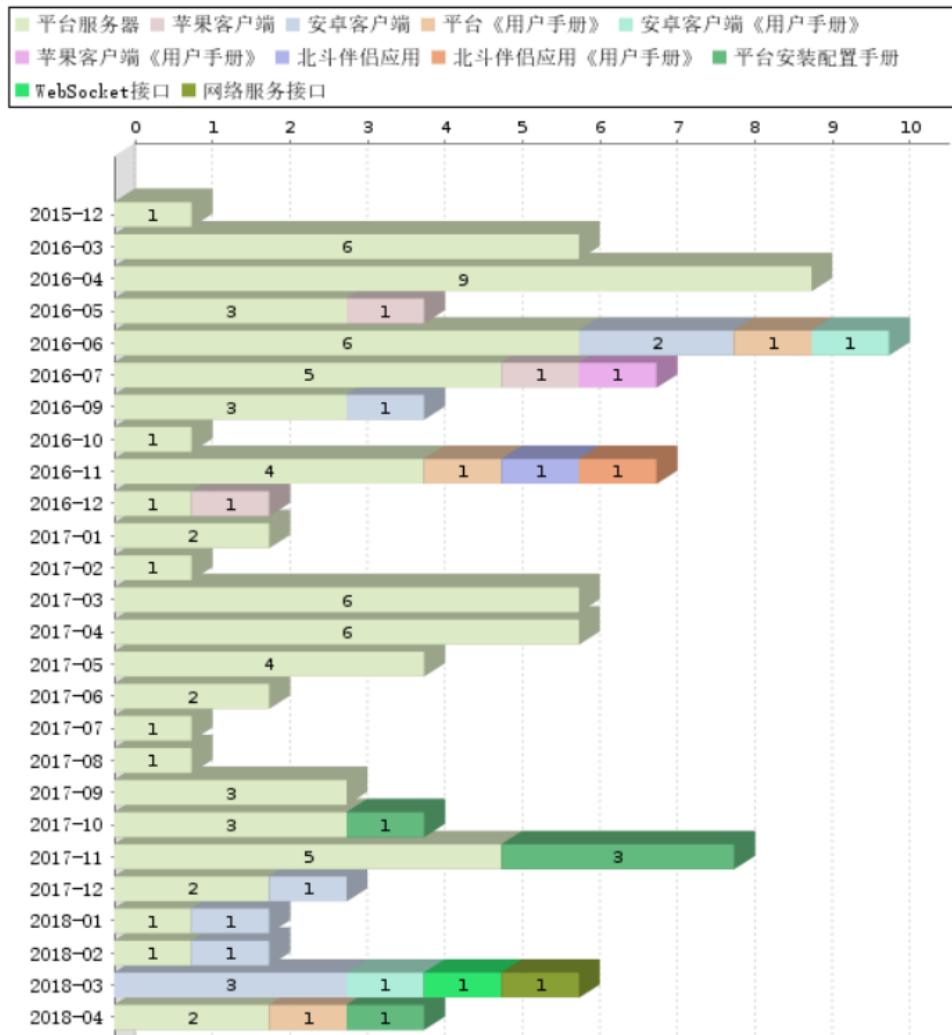
用户反馈类型的月统计报告



2) 选择菜单项“统计 → 版本历史”，可以从多个角度查看版本历史的统计信息：



软部件版本的月度统计报告



<文档结束>