**应用开发指南**

**修改记录**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **修改日期** | **版本号** | **修改内容** | **修改人** | **审核人** |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

# 目录

[目录 4](#_Toc430787345)

[修改记录 7](#_Toc430787346)

[1 概述 8](#_Toc430787347)

[2 代码生成工具 8](#_Toc430787348)

[2.1 概述 8](#_Toc430787349)

[2.2 工程说明 9](#_Toc430787350)

[2.3 开发步骤 9](#_Toc430787351)

[2.3.1 数据库设计及脚本 9](#_Toc430787352)

[2.3.2 导入demo工程 11](#_Toc430787353)

[2.3.3 增加配置文件 11](#_Toc430787354)

[2.3.4 生成文件 12](#_Toc430787355)

[2.3.5 生成的默认方法 13](#_Toc430787356)

[2.3.6 测试 13](#_Toc430787357)

[2.3.7 拷贝到实践工程 15](#_Toc430787358)

[2.4 工具其他参考信息 15](#_Toc430787359)

[2.4.1 配置示例 15](#_Toc430787360)

[2.4.2 表名前缀，简化pojo名称 16](#_Toc430787361)

[2.4.3 改变ID列 16](#_Toc430787362)

[3 开发环境搭建指南 17](#_Toc430787363)

[4 Web页面开发指南 18](#_Toc430787364)

[5 手机端开发指南 19](#_Toc430787365)

[6 附件 20](#_Toc430787366)

[6.1 开发规范 20](#_Toc430787367)

[6.1.1 Java开发规范 20](#_Toc430787368)

[6.1.2 数据库开发规范 20](#_Toc430787369)

[6.1.3 Js开发规范 20](#_Toc430787370)

# 修改记录

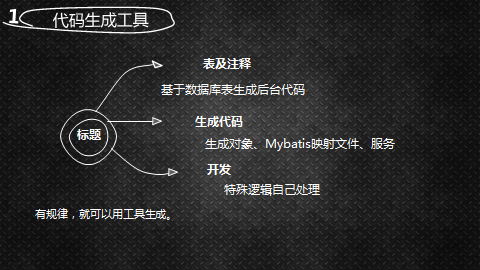
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **修改日期** | **版本号** | **修改内容** | **修改人** | **审核人** |
| 2015/9/21 | 1.0 | 创建 | 潘洪亮 |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

# 概述

本文档是主要内容帮助新加入RIIL团队的成员，自主完成开发环境搭建、配置及编译运行验证的操作手册。其中也包含团队成员在工作中常用工具的安装、配置说明。

# 代码生成工具

## 概述



根据数据表自动生成对应的pojo对象、服务接口、服务实现、mapper文件、mapper的sql文件、测试类和controller类，共7个文件

根据字段注释，生成pojo和服务的注释；

生成增删改查、批量插入，参数查询和分页查询等服务

自动生成测试用例

可以直接使用工程，生成代码；可以根据需要自己微调；

## 工程及jar说明

|  |  |  |
| --- | --- | --- |
| 序号 | 模块 | 说明 |
|  | app-base | 基础类库 |
|  | app-dev-tools | 代码生成工具 |
|  | app-model-test | 代码生成 |
|  | safe | WebDemo |

App-base 和app-dev-tools 都已经发布到 公司的仓库中。

使用公共库，需在pom.xml中加入下列内容：

<repository>

<snapshots>

<enabled>true</enabled>

</snapshots>

<id>public</id>

<name>Public Repositories</name>

<url>http://maven.glodon.com/nexus/content/groups/public/</url>

</repository>

并添加maven依赖

|  |
| --- |
| <dependency>  <groupId>com.glodon.app</groupId>  <artifactId>app-base</artifactId>  <version>1.0-SNAPSHOT</version>  </dependency> |

## 开发步骤

### 数据库设计及脚本

业务对象及关系分析升

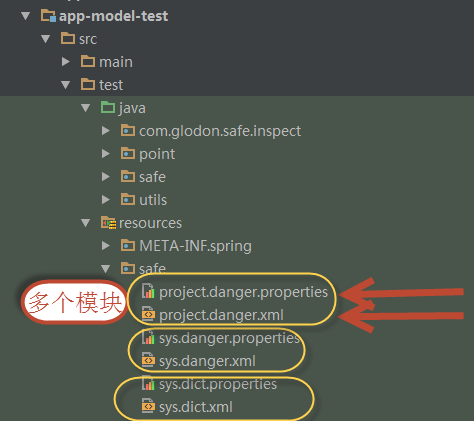
编写数据库脚本，包括注释，满足规范

|  |
| --- |
| DROP TABLE IF EXISTS `POI\_PRODUCT` CASCADE;  CREATE TABLE `POI\_PRODUCT` (  f\_id VARCHAR (32) NOT NULL COMMENT 'ID',  f\_code VARCHAR (32) NOT NULL COMMENT '产品编码',  f\_name VARCHAR (200) NOT NULL COMMENT '产品名称',  f\_desc VARCHAR (2000) COMMENT '描述',  f\_is\_deleted TINYINT(1) DEFAULT 0 COMMENT '废弃，是否删除，逻辑删除标记，非物理删除,1=删除、0=未删除',  f\_time TIMESTAMP COMMENT '修改时间',  f\_modify\_user VARCHAR (32) COMMENT '修改用户id',  f\_json VARCHAR (500) COMMENT 'json格式扩展信息',  f\_tag1 VARCHAR (32) COMMENT '冗余字段1',  f\_tag2 VARCHAR (32) COMMENT '冗余字段2',  f\_tag3 VARCHAR (32) COMMENT '冗余字段3',  f\_tag4 VARCHAR (32) COMMENT '冗余字段4',  PRIMARY KEY (f\_id)  ,UNIQUE(f\_code)  ) ENGINE = INNODB COMMENT = '产品表，如钢筋翻样、随手记'; |

### 导入demo工程

### 增加配置文件

#### 配置说明



一个模块包含两个配置文件，如项目危险源模块包括：

project.danger.properties 、project.danger.xml【文件名称相同，后缀不同】

在属性文件，配置数据库连接信息、文件生成路径、包名

在xml文件，增加表名

#### 配置示例

参见 mybatis.generator.properties、mybatis.generator.xml，修改红色部分

|  |
| --- |
| database.driver=com.mysql.jdbc.Driver  database.url=jdbc:mysql://172.17.189.75:3306/riil\_product?useUnicode=true&amp;characterEncoding=UTF-8&amp;autoReconnect=true  database.username=riil  database.password=r4rfde32wsaq1  #service package  model.package=its.dev.sys.log.model  model.src.java=src/main/java  model.src.resource=src/main/resources |

|  |
| --- |
| <table tableName="r\_report" enableCountByExample="false"  enableUpdateByExample="false" enableDeleteByExample="false" enableSelectByExample="false"  selectByExampleQueryId="false" >  </table> |

### 生成文件

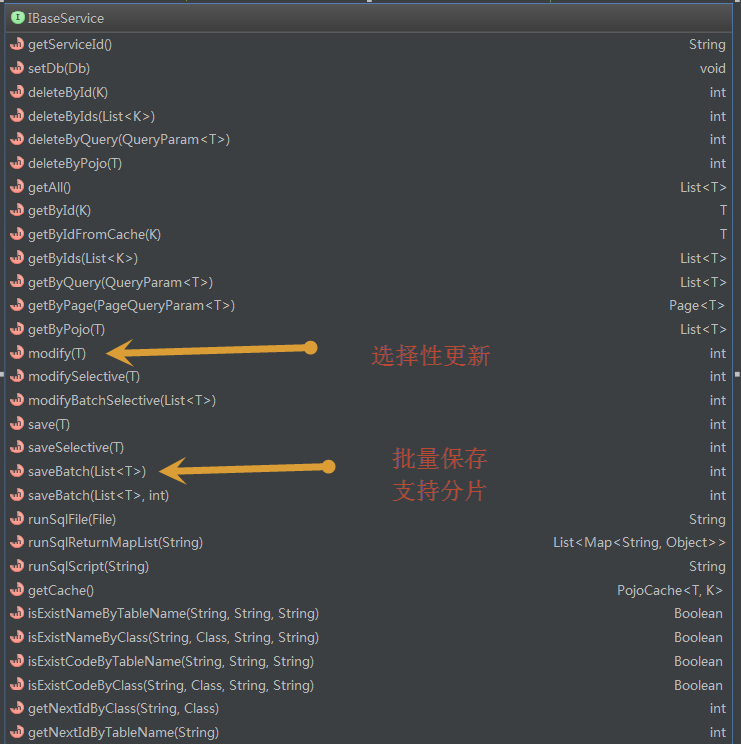
参考 safe.ServiceGeneratorMysql.java文件，修改相应参数，如：

Xml文件位置、表前缀、id字段等

运行ServiceGeneratorMysql.java，就会生成相应的文件

### 生成的默认方法





### 拷贝到实践工程

根据业务需要，自行调整xcopyToBiz.bat文件

并运行批处理文件，拷贝到相应工程模块

## 测试验证

### 前提条件

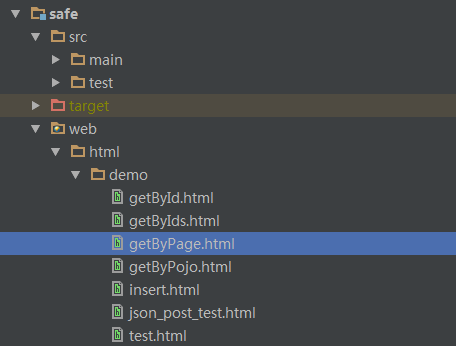
启动nginx

登录系统【用户safeUnitTestAdmin、密码password】【用户guxin、密码tenant】

### 单元测试

Test目录下会生成相应的单元测试类，可以进行业务验证测试。

### web测试demo



Demo页面访问地址

地址前缀：http://localhost:8989/safe/

<http://localhost:8989/safe/html/demo/getByIds.html>

[http://localhost:8989/safe/html/demo/insert.html](http://localhost:9001/html/demo/insert.html)

[http://localhost:8989/safe/html/demo/getByPage.html](http://localhost:9001/html/demo/getByPage.html)

[http://localhost:8989/safe/html/demo/getByPojo.html](http://localhost:9001/html/demo/getByPojo.html)

[http://localhost:8989/safe/html/demo/getByIds.html](http://localhost:9001/html/demo/getByIds.html)

页面包括url和参数信息，可以根据需要测试其他接口

服务访问地址

<http://localhost:8989/safe/services/projectProperty/getByIds?ids=1,2,3>

http://localhost:9001/services/projectProperty/getByIds?ids=1,2,3

### 测试接口

单一参数的http get请求，如deleteById、getById等

多参数的，http json post请求，如insert、getByPage等

#### 参数查询

##### url地址

../../services/projectProperty/getById ?id=1

##### 请求

用户id、产品code

|  |
| --- |
| **{**  **"userId"**: "1"**,**  **“productCode”: “sg.gj”**  **}** |

##### 响应

|  |
| --- |
| **{**http://tool.oschina.net/tools/json_format/Expanded.gif  **"code"**: "OK"**,**  **"message"**: "OK"**,**  **"body"**: **{**http://tool.oschina.net/tools/json_format/Expanded.gif  **"userId"**: "1"**,**  **"productId"**: "17"**,**  **"levelId"**: "104"**,**  **"invalidPoints"**: 80**,**  **"points"**: 85**,**  **"coins"**: 14**,**  **"invalidCoins"**: 0**,**  **"desc"**: null**,**  **"time"**: 1430795045**,**  **"levelName"**: "小兵"**,**  **"productName"**: "钢筋"**,**  **"availCoins"**: 14**,**  **"coinsDesc"**: "14-0=14"**,**  **},**  **"success"**: true  **}** |

#### Json post查询或修改、删除

##### url地址

../../services/projectProperty/getById ?id=1

##### 请求

用户id、产品code

|  |
| --- |
| **{**  **"userId"**: "1"**,**  **“productCode”: “sg.gj”**  **}** |

##### 响应

|  |
| --- |
| **{**http://tool.oschina.net/tools/json_format/Expanded.gif  **"code"**: "OK"**,**  **"message"**: "OK"**,**  **"body"**: **{**http://tool.oschina.net/tools/json_format/Expanded.gif  **"userId"**: "1"**,**  **"productId"**: "17"**,**  **"levelId"**: "104"**,**  **"invalidPoints"**: 80**,**  **"points"**: 85**,**  **"coins"**: 14**,**  **"invalidCoins"**: 0**,**  **"desc"**: null**,**  **"time"**: 1430795045**,**  **"levelName"**: "小兵"**,**  **"productName"**: "钢筋"**,**  **"availCoins"**: 14**,**  **"coinsDesc"**: "14-0=14"**,**  **},**  **"success"**: true  **}** |

### 本地数据库测试

修改app\_db\_config.json的连接参数信息

并在cloudt.properties文件增加

|  |
| --- |
| isRpc=false |

## 工具其他参考信息

### 配置示例

参见 mybatis.generator.properties、mybatis.generator.xml，修改红色部分

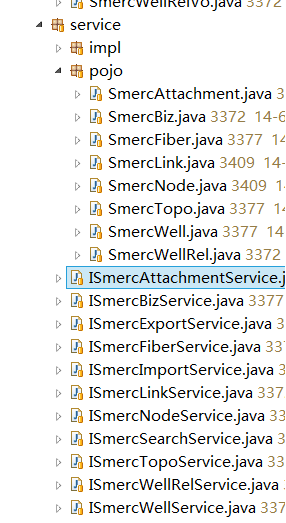
|  |
| --- |
| database.driver=com.mysql.jdbc.Driver  database.url=jdbc:mysql://172.17.189.75:3306/ product?useUnicode=true&amp;characterEncoding=UTF-8&amp;autoReconnect=true  database.username=safe  database.password=r4rfde32wsaq1  #service package  model.package=its.dev.sys.log.model  model.src.java=src/main/java  model.src.resource=src/main/resources |

|  |
| --- |
| <table tableName="t\_report" enableCountByExample="false"  enableUpdateByExample="false" enableDeleteByExample="false" enableSelectByExample="false"  selectByExampleQueryId="false" >  </table> |

### 表名前缀，简化pojo名称

|  |
| --- |
| GenUtils.*addTablePrefix*("r\_项目\_");  如 ：  代码生成生成工具指定表名前缀，生成的pojo、mapper、服务等就不包括前缀  GenUtils.addTablePrefix("t\_sys\_") ; 表 t\_sys\_danger 生成 Danger对象  GenUtils.addTablePrefix("t\_ ") ; 表 t\_sys\_danger 生成 SysDanger对象 |

错误示例



### 改变ID列

|  |
| --- |
| GenUtils.*setIdColumnName*("id"); |

## 自定义方法

### 事务及调用多个mapper

1.定义服务接口及实现

2.增加mapper接口及xml中sql语句

如：insertCustom

|  |
| --- |
| @Transactional      @Override      public int insertCustom (final ProjectProperty row) {          try {              SqlSession session = openSession();              getMapper(session).insert(row);  //第二mapper              DangerLevelBasisMapper mapper = getMapper(session, DangerLevelBasisMapper.class);              DangerLevelBasis levelBasis = buildDangerLevelBasis(row);  //插入第二个对象              mapper.insert(levelBasis);              return 1;          } catch (Exception e) {              processException(e);          }          return -1;      }      private DangerLevelBasis buildDangerLevelBasis(ProjectProperty row) {          DangerLevelBasis levelBasis = new DangerLevelBasis();          levelBasis.setId(row.getId());          levelBasis.setCode(row.getCode());          return levelBasis;      } |

## 与gsql开发比较

只是基本的增删改查，还没有分页、名称重复校验、编码重复校验、排序、批量存储、局部更新等业务





# 开发环境搭建指南

## Tomcat

## jetty

# Web页面开发指南

# 手机端开发指南

# 附件

## IBaseService的方法

|  |  |
| --- | --- |
| **方法** | |
| **限定符和类型** | **方法和说明** |
| int | [**deleteById**](file:///D:\tmp\com\glodon\app\base\service\IBaseService.html#deleteById(K))([**K**](file:///D:\tmp\com\glodon\app\base\service\IBaseService.html) id)  按ID删除记录 |
| int | [**deleteByIds**](file:///D:\tmp\com\glodon\app\base\service\IBaseService.html#deleteByIds(java.util.List))(java.util.List<[**K**](file:///D:\tmp\com\glodon\app\base\service\IBaseService.html)> ids)  批量删除多条记录 |
| int | [**deleteByPojo**](file:///D:\tmp\com\glodon\app\base\service\IBaseService.html#deleteByPojo(T))([**T**](file:///D:\tmp\com\glodon\app\base\service\IBaseService.html) pojo)  根据pojo属性批量删除对象 |
| int | [**deleteByQuery**](file:///D:\tmp\com\glodon\app\base\service\IBaseService.html#deleteByQuery(com.glodon.app.base.service.pojo.QueryParams))([**QueryParams**](file:///D:\tmp\com\glodon\app\base\service\pojo\QueryParams.html)<[**T**](file:///D:\tmp\com\glodon\app\base\service\IBaseService.html)> param)  根据参数批量删除对象 |
| java.util.List<[**T**](file:///D:\tmp\com\glodon\app\base\service\IBaseService.html)> | [**getAll**](file:///D:\tmp\com\glodon\app\base\service\IBaseService.html#getAll())()  获取所有记录 |
| [**T**](file:///D:\tmp\com\glodon\app\base\service\IBaseService.html) | [**getById**](file:///D:\tmp\com\glodon\app\base\service\IBaseService.html#getById(K))([**K**](file:///D:\tmp\com\glodon\app\base\service\IBaseService.html) id)  按ID获取单行记录 |
| [**T**](file:///D:\tmp\com\glodon\app\base\service\IBaseService.html) | [**getByIdFromCache**](file:///D:\tmp\com\glodon\app\base\service\IBaseService.html#getByIdFromCache(K))([**K**](file:///D:\tmp\com\glodon\app\base\service\IBaseService.html) id)  按ID获取单行记录 |
| java.util.List<[**T**](file:///D:\tmp\com\glodon\app\base\service\IBaseService.html)> | [**getByIds**](file:///D:\tmp\com\glodon\app\base\service\IBaseService.html#getByIds(java.util.List))(java.util.List<[**K**](file:///D:\tmp\com\glodon\app\base\service\IBaseService.html)> ids)  按ID获取多条记录 |
| [**Page**](file:///D:\tmp\com\glodon\app\base\service\pojo\Page.html)<[**T**](file:///D:\tmp\com\glodon\app\base\service\IBaseService.html)> | [**getByPage**](file:///D:\tmp\com\glodon\app\base\service\IBaseService.html#getByPage(com.glodon.app.base.service.pojo.PageQueryParams))([**PageQueryParams**](file:///D:\tmp\com\glodon\app\base\service\pojo\PageQueryParams.html)<[**T**](file:///D:\tmp\com\glodon\app\base\service\IBaseService.html)> pageQueryParam)  分页查询. |
| java.util.List<[**T**](file:///D:\tmp\com\glodon\app\base\service\IBaseService.html)> | [**getByPojo**](file:///D:\tmp\com\glodon\app\base\service\IBaseService.html#getByPojo(T))([**T**](file:///D:\tmp\com\glodon\app\base\service\IBaseService.html) pojo)  根据对象本身信息，进行过滤 |
| java.util.List<[**T**](file:///D:\tmp\com\glodon\app\base\service\IBaseService.html)> | [**getByQuery**](file:///D:\tmp\com\glodon\app\base\service\IBaseService.html#getByQuery(com.glodon.app.base.service.pojo.QueryParams))([**QueryParams**](file:///D:\tmp\com\glodon\app\base\service\pojo\QueryParams.html)<[**T**](file:///D:\tmp\com\glodon\app\base\service\IBaseService.html)> queryParam)  根据查询参数获取多条记录 |
| [**CachePojos**](file:///D:\tmp\com\glodon\app\base\service\cache\CachePojos.html)<[**T**](file:///D:\tmp\com\glodon\app\base\service\IBaseService.html),[**K**](file:///D:\tmp\com\glodon\app\base\service\IBaseService.html)> | [**getCachePojos**](file:///D:\tmp\com\glodon\app\base\service\IBaseService.html#getCachePojos())()  获取缓存对象 getCache. |
| long | [**getNextIdByClass**](file:///D:\tmp\com\glodon\app\base\service\IBaseService.html#getNextIdByClass(java.lang.String,%20java.lang.Class))(java.lang.String prefix, java.lang.Class clazz)  根据类,获取下一个id |
| long | [**getNextIdByTableName**](file:///D:\tmp\com\glodon\app\base\service\IBaseService.html#getNextIdByTableName(java.lang.String))(java.lang.String tableName)  根据表名，获取下一个id |
| java.lang.String | [**getServiceId**](file:///D:\tmp\com\glodon\app\base\service\IBaseService.html#getServiceId())()  服务id |
| int | [**insert**](file:///D:\tmp\com\glodon\app\base\service\IBaseService.html#insert(T))([**T**](file:///D:\tmp\com\glodon\app\base\service\IBaseService.html) row)  插入一条记录 |
| int | [**insertBatch**](file:///D:\tmp\com\glodon\app\base\service\IBaseService.html#insertBatch(java.util.List))(java.util.List<[**T**](file:///D:\tmp\com\glodon\app\base\service\IBaseService.html)> rows)  批量插入对象 |
| int | [**insertBatch**](file:///D:\tmp\com\glodon\app\base\service\IBaseService.html#insertBatch(java.util.List,%20int))(java.util.List<[**T**](file:///D:\tmp\com\glodon\app\base\service\IBaseService.html)> rows, int batchSize)  批量保存对象 可以分批次保存 如 1000条数据，分10次保存，每次保存100条 |
| int | [**insertSelective**](file:///D:\tmp\com\glodon\app\base\service\IBaseService.html#insertSelective(T))([**T**](file:///D:\tmp\com\glodon\app\base\service\IBaseService.html) row)  选择性插入 只保存非空字段 |
| java.lang.Boolean | [**isExistCodeByClass**](file:///D:\tmp\com\glodon\app\base\service\IBaseService.html#isExistCodeByClass(java.lang.String,%20java.lang.Class,%20K,%20java.lang.String))(java.lang.String prefix, java.lang.Class clazz, [**K**](file:///D:\tmp\com\glodon\app\base\service\IBaseService.html) id, java.lang.String code)  根据类名，判断 名称 是否存在 |
| java.lang.Boolean | [**isExistCodeByTableName**](file:///D:\tmp\com\glodon\app\base\service\IBaseService.html#isExistCodeByTableName(java.lang.String,%20K,%20java.lang.String))(java.lang.String tableName, [**K**](file:///D:\tmp\com\glodon\app\base\service\IBaseService.html) id, java.lang.String code)  根据表名判断 名称 是否存在 |
| java.lang.Boolean | [**isExistNameByClass**](file:///D:\tmp\com\glodon\app\base\service\IBaseService.html#isExistNameByClass(java.lang.String,%20java.lang.Class,%20K,%20java.lang.String))(java.lang.String prefix, java.lang.Class clazz, [**K**](file:///D:\tmp\com\glodon\app\base\service\IBaseService.html) id, java.lang.String name)  根据表名，判断 名称 是否存在 |
| java.lang.Boolean | [**isExistNameByTableName**](file:///D:\tmp\com\glodon\app\base\service\IBaseService.html#isExistNameByTableName(java.lang.String,%20K,%20java.lang.String))(java.lang.String tableName, [**K**](file:///D:\tmp\com\glodon\app\base\service\IBaseService.html) id, java.lang.String name)  根据表名，判断 名称 是否存在 |
| int | [**modify**](file:///D:\tmp\com\glodon\app\base\service\IBaseService.html#modify(T))([**T**](file:///D:\tmp\com\glodon\app\base\service\IBaseService.html) row)  更新单条记录 |
| int | [**modifyBatchSelective**](file:///D:\tmp\com\glodon\app\base\service\IBaseService.html#modifyBatchSelective(java.util.List))(java.util.List<[**T**](file:///D:\tmp\com\glodon\app\base\service\IBaseService.html)> rows)  批量选择性保存 只保存非空字段 |
| int | [**modifySelective**](file:///D:\tmp\com\glodon\app\base\service\IBaseService.html#modifySelective(T))([**T**](file:///D:\tmp\com\glodon\app\base\service\IBaseService.html) row)  选择性保存 只保存非空字段 |
| int | [**replace**](file:///D:\tmp\com\glodon\app\base\service\IBaseService.html#replace(T))([**T**](file:///D:\tmp\com\glodon\app\base\service\IBaseService.html) row)  替换一条记录 |
| int | [**replaceBatch**](file:///D:\tmp\com\glodon\app\base\service\IBaseService.html#replaceBatch(java.util.List))(java.util.List<[**T**](file:///D:\tmp\com\glodon\app\base\service\IBaseService.html)> rows)  批量替换对象 |
| int | [**replaceBatch**](file:///D:\tmp\com\glodon\app\base\service\IBaseService.html#replaceBatch(java.util.List,%20int))(java.util.List<[**T**](file:///D:\tmp\com\glodon\app\base\service\IBaseService.html)> rows, int batchSize)  批量替换对象 可以分批次保存 如 1000条数据，分10次保存，每次保存100条 |
| java.lang.String | [**runSqlFile**](file:///D:\tmp\com\glodon\app\base\service\IBaseService.html#runSqlFile(java.io.File))(java.io.File sqlScriptFile)  执行sql脚本文件 |
| java.util.List<java.util.Map<java.lang.String,java.lang.Object>> | [**runSqlReturnMapList**](file:///D:\tmp\com\glodon\app\base\service\IBaseService.html#runSqlReturnMapList(java.lang.String))(java.lang.String sql) |
| java.lang.String | [**runSqlScript**](file:///D:\tmp\com\glodon\app\base\service\IBaseService.html#runSqlScript(java.lang.String))(java.lang.String sqlScript)  执行sql脚本文件 |
| void | [**setDb**](file:///D:\tmp\com\glodon\app\base\service\IBaseService.html#setDb(com.glodon.app.base.db.Db))([**Db**](file:///D:\tmp\com\glodon\app\base\db\Db.html) db)  setDb. |

## Mvn创建工程

|  |
| --- |
| 创建一个Project  mvn archetype:create -DgroupId=com.glodon.safe -DartifactId=project-area -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false  cd project-area  mvn archetype:generate -DgroupId=com.glodon.safe -DartifactId=safe-danger-api -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false  mvn archetype:generate -DgroupId=com.glodon.safe -DartifactId=safe-danger-impl -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false  　　mvn archetype:generate　　固定格式  　　-DgroupId　　　　　　　　　组织标识（包名）  　　-DartifactId　　　　　　　　项目名称  　　-DarchetypeArtifactId　　  指定ArchetypeId，maven-archetype-quickstart，创建一个Java Project；maven-archetype-webapp，创建一个Web Project  　　-DinteractiveMode　　　　　　是否使用交互模式 |

## 开发规范

### Java开发规范

### 数据库开发规范

### Js开发规范