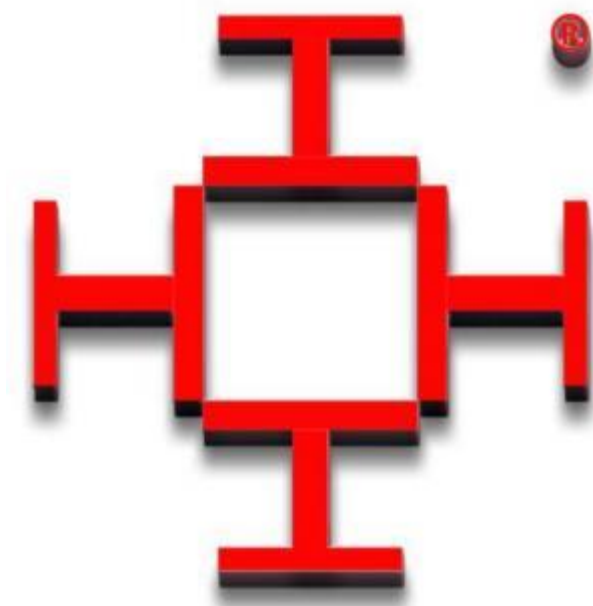




框架更新日志

V1.0.0

2022- 1-26

V3.12 更新说明

2024-02-19

【新增】远程调用新增 动态Job 支持在线Job编写与调用

【新增】移除session.type 配置，修改为 session.store；
提供配置项 Memory, Redis, JDBC

【新增】请求头或者请求参数携带 X-Access-Token :token
或UserSessionId 就可以登录

【新增】系统菜单新增 访问令牌 ； URL只要携带令牌就可以
登录

【新增】LoginHelper 类 新增方法 loginByCode() 用于企
业微信钉钉等登录

```
LoginHelper.loginByCode("code", new Function<String, String>() {
```

```
@Override
```

```
public String apply(String t) {
```

```
// TODO 自动生成的方法存根
```

```
return null;
```

```
}
```

```
});
```

【优化】资源访问不需要携带 账套信息

【删除】SessionListener类删除

【删除】AppConnector类删除

【修复】硕正报表编辑时切换选项卡没有保存修改内容问题

【修复】非主数据源行列转至报错问题

【修复】导入控制台同步字段涉及维度或者Plg新增字段或
Plg注入App表字段报错问题

【修复】新装插件，注入方法没有直接生效问题

【修复】Eclipse插件，销毁插件后，切换应用下拉插件异常
问题

【修复】安装包自动备份失效问题

【修复】打包或者升级APP 报找不到xx文件问题

【修复】控制台启动端口会多一位问题

【优化】去掉导出Excel是 如果是时间字段 携带1970/1/1前
缀问题

【优化】完善对Hql语句空格的校验

【优化】导入控制台增加对导入文件大小的限制，目前最大
5MB

【优化】Plg注入App字段的临时表 能够使用Record对象进行Set值而不是单单只能转成Map后put值

【修复】数据库拟合不存在Plg文件夹时报错问题

【修复】临时表改字段名字（sysUUID改成UUID），初始化应用 初始化全局都没用 要重启tomcat才不报错

【优化】支持跨应用调用某个类的某个Private方法

【修复】数据库拟合存在数据库生成字段带. 问题

【优化】DAOHelper.getPropertiesListAliasMap()获取的字段如果不指定别名 现在不再是0, 1, 2, 3 命名 而是对应字段名称

【优化】导出Excel文件支持DateTime 类型

【优化】导入控制台 自定义模式增加了validateDataRef属性控制，标准导入必须校验

【优化】Plg层的数据源对openLookup 和lookup方法的支持

【修复】详细信息对Plg注入App字段的支持

【修复】查询器当查询值内容为空是 操作符显示错误

【新增】附件类型是图片时，增加预览旋转功能

【优化】时间类型字段如果有值下拉选择时间时默认是该值

【修复】在数据录入的过程中，按tab键自动带出的后面的值，焦点落在后面的值时，再按TAB键，标题异常问题

【修复】非临时表 DataSource openLookup_ 方法 setLabel 没有效果问题

【优化】支持临时表的万能查询器筛选

【修复】Plg表的详细信息，显示不全问题

【修复】万能查询器增加查询条件无反应问题

【修复】有数据源StringEditor 设置 Label 为全角空格，画面显示错误问题

【修复】插件注入App表维度字段画面显示异常问题

【优化】对Enum字段查询增加异常查询值时 在万能查询器里面不再增加错误查询条件

【修复】查询Group日期字段需要先点一下日期输入框在选择日期才有效问题

【修复】Form modifiedEditor 方法里面写了 alert，会把画面上新的值冲掉问题

【优化】currentQuery中如果已有某字段 如果原字段不是隐藏或者锁定 则默认应该替换掉

【修复】临时表delayed 关联其他正常数据源 对应查询不符合预期效果问题

【修复】临时表设置默认查询后 对应数据过滤异常问题

【优化】Dialog label 类型字段支持显示\n

【修改】右键画面调整对于Image、DataGrid设置高度无效问题

【新增】系统参数新增 新用户首次登录修改密码 配置

【修复】GOT中未提升底层Menu菜单时新建节点到菜单下，后进行恢复操作报错问题

【新增】GOT中修改表的NOTNULL属性时若原数据库中有数据且为空，增加提醒

【修复】插件开发，提升了App的form，增加一个插件的表的数据源，右键方法为空问题

【修复】App表上增加的插件字段，再画面上拖出来 和数据源重置关联节点出来的名称不一致问题

【修复】提升Plg层临时表 报错问题

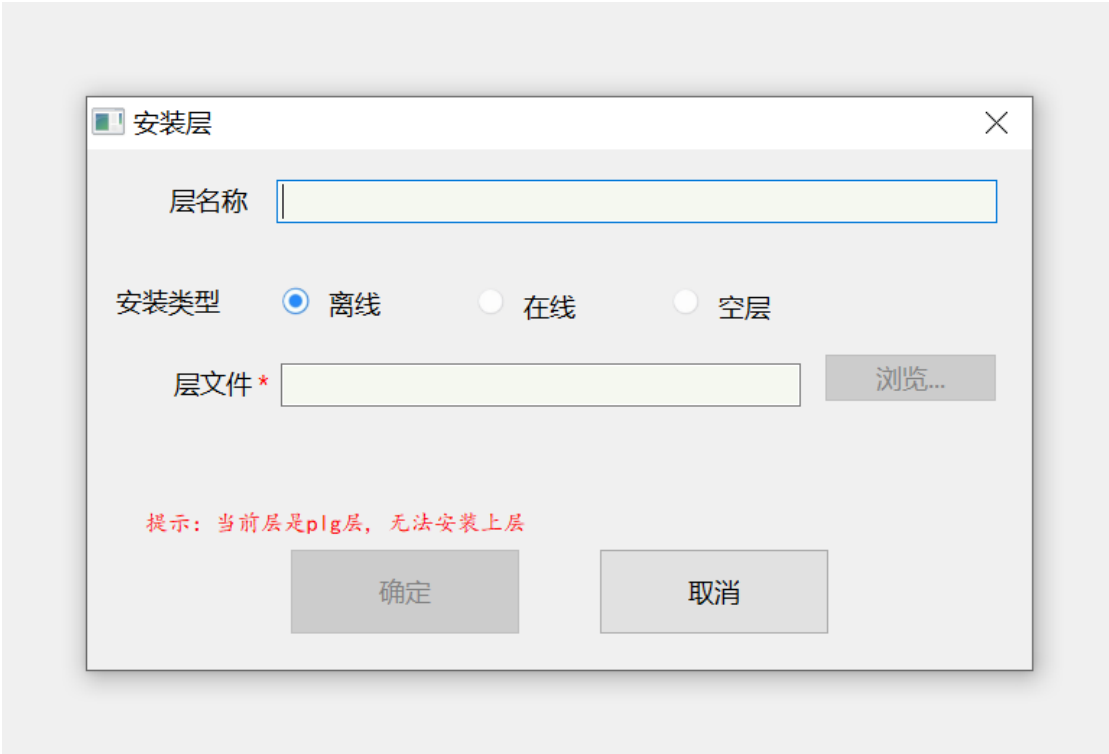
【优化】针对FormItem、TableField增加服务端二次重名校验

【优化】修改DataSource的Extend属性时如果存在returnTable returnField，增加提示

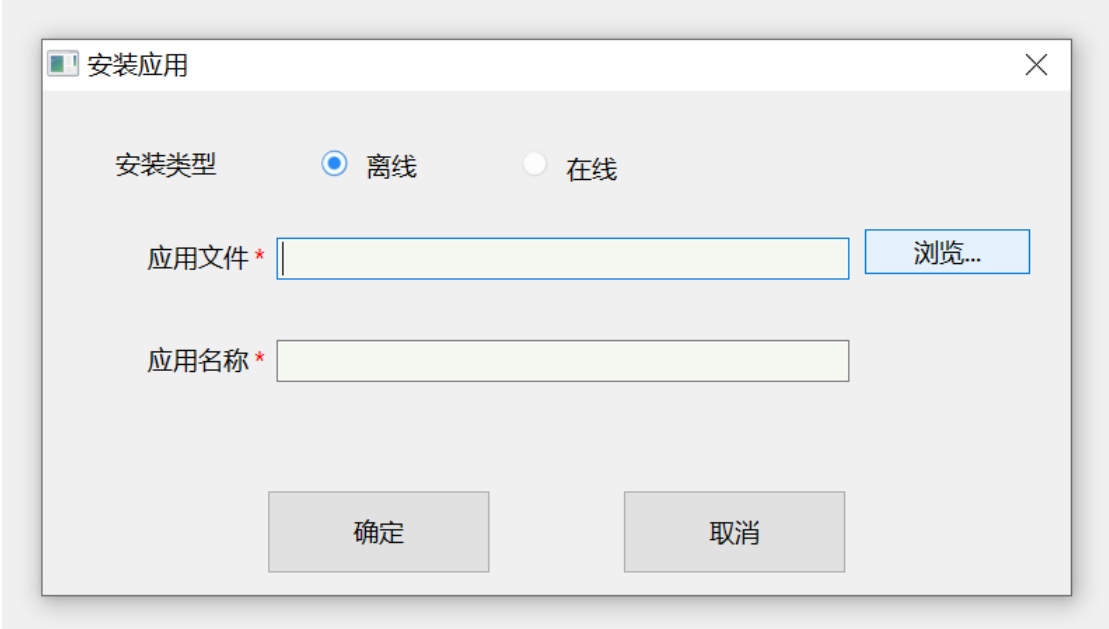
【新增】日期DisplayStyle属性 新增样式 [yyyy]

【优化】进一步增加Field字段名称的关键字校验，如：Fetch 将不能创建

【新增】Eclipse开发助手层链管理新增安装层，支持在线安装底座或安装空层



【新增】Eclipse开发助手应用管理新增应用安装，支持在线安装应用



【新增】Eclipse开发助手应用管理编辑应用，支持绑定行业层

编辑应用

应用名称 *

gongqi.df.invent

应用标签 *

库存

应用版本 *

1.0.0

部署模式

二进制

绑定对象层

☐

绑定行业层

☐

确定

取消

【新增】Eclipse开发助手插件管理安装插件，在线安装插件

安装插件

安装类型

☐ 离线

☒ 在线

应用名称 *

gongqi.df.invent

ⓘ

插件名称 *

浏览...

插件版本 *

浏览...

确定

取消

【新增】Eclipse开发助手应用管理编辑插件，支持绑定行业层和对象层



编辑插件

应用名称 * gongqi.df.invent

插件名称 * gq.pl.ivPl

插件标签 * 1

插件版本 * 1.0

部署模式 Source

绑定对象层 ☐ 绑定行业层 ☐

确定 取消

【优化】OS控制台按钮样式深度优化

【优化】开辟域过程中的临时文件放到期望的临时目录下

【新增】OS控制台增加一个数字域选项卡

【新增】OS控制台针对环境对比，增加配置界面，用于配置访问令牌（支持随机生成，默认随机生成一个），配置是否允许更新（默认false）

【新增】环境对比工具在OS桌面增加一个快捷入口，类似计算器那样的入口，控制柜台也相应增加

【新增】通过域管理启动域，后台静默运行，有统一日志文件，类似服务形式

【优化】应用升级时针对WebSocket添加失败优化提示内容

【优化】应用升级时针对异常报错优化提示内容

【优化】OS控制台更新授权优化文件校验过程

【新增】OS控制台对应设置功能新增配置 关闭时：最小化到托盘、退出程序，两个选项，默认最小化到托盘

【修复】商店搜索出来显示已安装，实际没有安装问题

【修复】os配置权限，通过搜索条件所有出来，配置权限并且提交成功之后，去掉搜索条件，菜单选择没有重置问题

【修复】OS权限配置如果存在筛选条件时仅保存当前筛选条件下的配置问题

【修复】画面配置高级权限，配置按钮不可编辑时，整个画面都不可编辑问题

【修复】权限配置筛选功能多次筛选无效问题

【修复】应用权限管理独立画面根节点点了后保存不生效

【修复】商店管理中 看到的已安装插件与 应用列表中看到的已安装插件不一致问题

【修复】OS权限配置配置应用权限管理时，若该用户组已有权限，然后再配置新的权限画面时，当使用了搜索功能搜索某个画面，然后保存方案时，会把某个权限给去掉问题

【修复】os进行域的ext更新的时候，如果这个域没有安装某个插件而其他有域安装了这个插件，那么更新的时候就会报错，导致无法更新问题

【修复】应用权限管理中，给用户配权限时勾选了上方的权限，结果会把禁用的也给勾上问题

【修复】同步权限时存在云应用报错问题

【修复】域管理更新框架存在文件占用问题

【修复】os里配置用户组的时候，从用户界面给用户单独配置用户组出现空指针异常问题

【优化】商店某App不存在插件时 优化提示

【修复】商店搜索框光标位置错位问题

【修复】调整系统桌面，浏览器关掉重新登录后，调整失效问题

【修复】OS内xlg文件下载本地失效问题

【修复】OS权限配置 配置项全选打钩但其他选项不打勾问题

【修复】清空个人权限后，再勾选这个app后，自定义项的权限不清空问题

【修复】Tomcat用服务开，excel转pdf调不到office的jacob问题

【修复】APP应用右上角的隐藏，放大，关闭图标显示异常问题

【优化】域管理域更新界面新增进度描述等

【优化】域管理域调整顺序的功能

【优化】域管理安装新域时增加对授权文件的校验

【修复】更新域版本的时候，综合域有个未知版本号问题

【优化】商店升级APP时新增当前版本号

【优化】商店版本列表的版本描述样式内容

【优化】商店的详情进去默认停留在版本列表

【优化】商店应用安装/升级等页面的图标

【优化】商店界面字体样式

【优化】商店选择版本功能的界面

【新增】CommandResult增加对Decimal类型的字段调整
ShowDecimal的方法 updateControlShowDecimal

```
public CommandResult buttonOK_Button(CommandArg arg) {  
    CommandResult result = new CommandResult();  
    result.updateControlShowDecimal(TabelName._DecimalEditor,4);  
    return result;  
}
```

- 【修复】在插件层提升APP表然后新增字段并且拖入画面后，初始化画面后 提升的字段未显示 问题
- 【修复】DataSource上写openlookup, lookup方法后，初始化全局没效果，初始化画面有效果的问题
- 【优化】导入控制台，数据导入的文件类型不对时，导入历史的状态从执行中，改为失败
- 【新增】查看引用列表，增加label显示
- 【删除】删除SessionListener机制，通过事件机制实现
- 【新增】行列转置支持非主数据源

GOT DataSource 属性 DataSourceType新增选项

Pivot、新增属性PivotSource

AllowAttachment	true
DataSourceType	Pivot
PivotSource	
PivotColumn	
PivotAggregateType	
PivotAggregateColumn	

【新增】导入控制台增加事务范围控制，新增可配置选项 事务范围

- 整体：导入全部数据 要么一起成功 要么一起失败
- 逐行：单行数据判断 数据报错单行回滚

【新增】导入控制台 导入历史中增加终止导入功能，事务范围不一样，终止范围不一样，增加预计时间，增加导入时长

*	导入日期	开始时间	导入状态	导入进度	导入时长	剩余时长	进度描述
1	2023-09-22	10:22:37	成功				

【优化】优化afterinit事件，确保始终执行

【优化】优化activeRecord事件，确保始终执行

【优化】优化临时表删除记录时，能够获取到当前记录

【新增】日志增加过期自动删除功能，默认30天，

MaxHistoryDay

```

119 log4j.appender.PerformanceMemory=gongqi.erp.framework.logging.DailyRollingFileAppender
120 log4j.appender.PerformanceMemory.File=./logs/platform/PerformanceMemory.log
121 log4j.appender.PerformanceMemory.layout=org.apache.log4j.PatternLayout
122 log4j.appender.PerformanceMemory.layout.ConversionPattern=%p|%d{yyyy-MM-dd}|%d{HH:
123
124 log4j.appender.PerformanceXmlHttpRequest.MaxHistoryDay=30
125 log4j.appender.PerformanceXmlHttpRequest=gongqi.erp.framework.logging.DailyRollingFileA
126 log4j.appender.PerformanceXmlHttpRequest.File=./logs/platform/PerformanceXmlHttpReque
127 log4j.appender.PerformanceXmlHttpRequest.layout=org.apache.log4j.PatternLayout
128 log4j.appender.PerformanceXmlHttpRequest.layout.ConversionPattern=%p|%d{yyyy-MM-dd}
129
130 log4j.appender.bizlogger.MaxHistoryDay=30
131 log4j.appender.bizlogger=gongqi.erp.framework.logging.DailyRollingFileAppender
132 log4j.appender.bizlogger.File=./logs/erp/BizLog.log
133 log4j.appender.bizlogger.layout=org.apache.log4j.PatternLayout
134 log4j.appender.bizlogger.layout.ConversionPattern=%p|%d{yyyy-MM-dd}|%d{HH:mm:ss}|%m

```

【优化】 优化应用加载失败逻辑，加载失败再次执行加载

【修复】 HQL语句where (A条件 and B条件) or C条件 转义的SQL不对的问题

【优化】 调试模式下，启动ERP如果加载失败会抛出异常，阻止启动。

【修复】 修复生成行号的时候如果没有filterfield会报错的情况

【修复】 修复批量插入、更新对于维度字段不支持的缺陷；

【优化】 废弃ServiceGuard、CommandHelper，建议通过ServiceDispatch.runXXX实现

【新增】 编号序列支持获取父级记录字段值，当前表字段[filter字段]/父级表字段， filter字段可选

如：{ABC}、{[yyyy]Date}、{ABC/EFG}、{ABC[HJK]/EFG}

【修复】 修复对于存在应用插件扩展等情况时，域定时器启动后，任务无法自动触发

【修复】 Sys层系统菜单下的-数据库会话，打开就报错问题

【优化】 二次开发Css 层链加载结构 上层优先

【修复】 GOT插件-功能里调整名称后，重新加载got报安全模式，xml没改问题

【优化】 GOT插件-对Ids重复用户进行校验

【新增】 GOT插件-AutoLookup 下节点增加 visable 属性

【优化】 GOT插件-queryGroup的按钮，选择buttonType 自动带label

【修复】 GOT插件-显示标签状态下，打开menu新窗口，不显示标签的问题

【修复】 GOT插件-Form移至MenuItems 未保存情况下 随便点击一个地方 报空指针，报错问题

【修复】 GOT插件- AutoLookUp下新增节点 没有选继承字段提示不允许

【修复】 GOT插件- 数据库同步，字段数据类型不一致，同步失败问题

【修复】JDT插件- 查看层里，修改层标签，确定了，重新打开查看 还是没效果的问题

【修复】GOT插件- GOT表两次建同名不同类型报错问题

【修复】GOT插件- 插件提升APP 添加字段保存后在修改字段，字段名保存错误(没有带插件前缀)问题

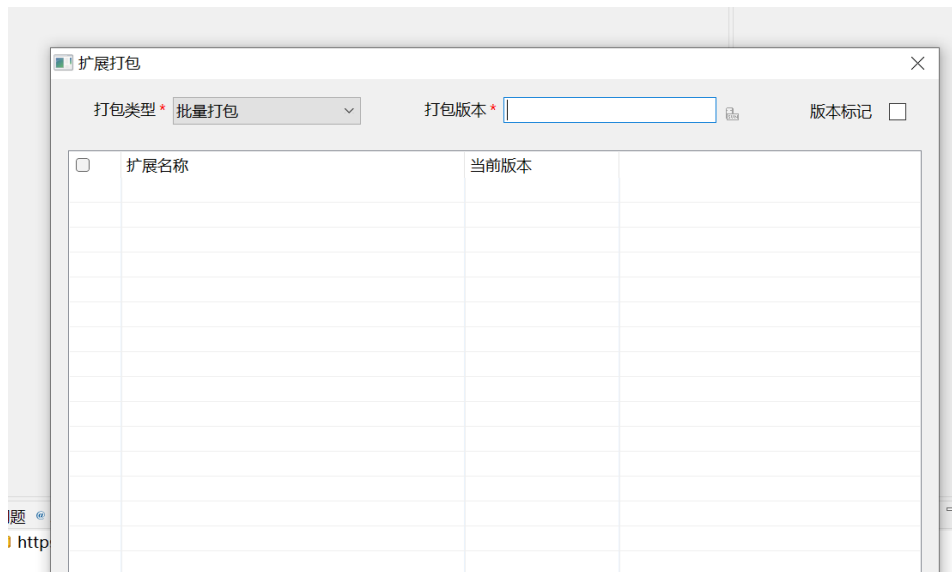
【优化】GOT插件-Table新增字段后 修改名字后 选择继承属性 不再进行修改字段名称

【修复】GOT插件-MenuItem节点每次修改Lable, EntityName 都会发生变化的问题

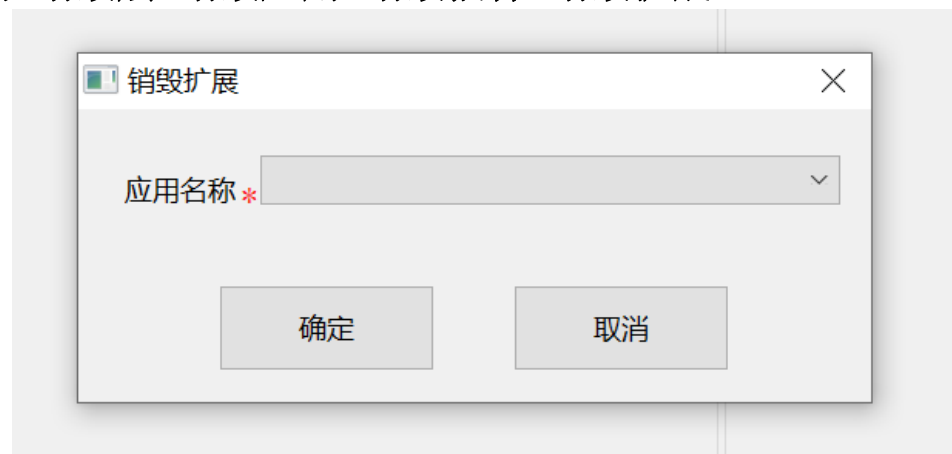
【优化】GOT插件-MenuItem节点的SourceType为Form的时候，如果未输入Source，不允许保存

【新增】JDT插件-打包层、打包应用、打包插件、打包扩展
新增版本管理 显示当前版本 查看历史版本 版本打包规则
强制 x.y.z 多层打包时版本号统一

版本号	提交人	提交日期	版本说明
1.0.20	yanglei	2023-09-21 17:24	ddd
1.0.19	yanglei	2023-09-21 17:17	qaee
1.0.18	yanglei	2023-09-21 17:15	qaa123
1.0.17	yanglei	2023-09-21 17:14	azzzz
1.0.16	yanglei	2023-09-21 17:12	qad123
1.0.15	yanglei	2023-09-21 17:11	qwee
1.0.14	yanglei	2023-09-21 16:32	1aad
1.0.13	yanglei	2023-09-21 11:20	qadddd
1.0.12	yanglei	2023-09-21 11:18	qaf
1.0.11	yanglei	2023-09-21 11:14	1qaz
1.0.10	yanglei	2023-09-13 13:58	qweasd



新增 销毁层 销毁应用 销毁插件 销毁扩展



【新增】新增对dataMethod字段写mainForm_方法的支持

【新增】新增time字段的下拉秒的选择

【修复】Dialog 下Time类型字段无法获取问题

【修复】datagrid里面，双击点到不可编辑的字段里面，然后直接点击删除，删除成功后，没法新增记录，和关闭画面的问题

【修复】回车切换字段后toolTip无法正常消失问题

【修复】DataGrid最后一行 回车无法进入编辑状态的问题

【新增】多次打开同一个标签页 允许每次打开都刷新

【优化】优化datagrid 工具栏 在不同宽度下的显示

【优化】优化显示画面使得部分页面不出现滚动条

【优化】优化 输入字段大小写未区分

【新增】新增 String 和Decimal 类型的数字小键盘

- 【修复】无数据源Time Date 类型无法复制值的问题
- 【修复】画面调整无法对Image控件调整高度问题
- 【修复】临时表过滤查询时候部分条件失效 如 x**, 空格.00等
- 【修复】下面画面支持 updateControlLable 对DataGrid内字段修改
- 【优化】Dialog 下 label 支持对\n 的识别
- 【修复】FadeInfoArg命令cornerRadius无效; labelColor无效; 的问题
- 【修复】GOT Form的plugin控件为html, 隐藏没有效果的问题
- 【修复】自定义下拉画面 支持 x** 自动过滤筛选
- 【修复】频繁切换tabpage datagrid 高度变小 问题

【优化】 validateRecordFieldRef方法，增加校验引用值是否有效

【修复】 SysTimeHelper.diff 算时差，time为空值时，算的时差不对

【优化】 QueryTable中的findOrCreateWhereItem方法，针对维度whereItem的情况，增加一个新方法 实现传整个维度对象进去

【优化】 应用的加载机制改进，应用为单位进行加载（优先加载底座，加载账套，后逐个加载单个应用包括该应用插件与扩展）

重点说明 （1）dataArealistener加载时机移到所有应用加载之前

（2）应用的加载统使得对应用初始化可以写在application不需要写监听

【优化】 form增加proxy机制

```
@Override  
public void registerObjects() {  
    ObjectContainer.getInstance(GongqiFormProxyManager.class).addProxy(new GongqiFormProxyImpl());  
}
```

```

public class GongqiFormProxyImpl extends GongqiFormProxy{

    @Override
    protected boolean supports(GOT_Form arg0) {
        // TODO 自动生成的方法存根
        return true;
    }

    @Override
    public CommandResult activeRecord(FormDataSourceRoot dataSource, CommandArg arg) {
        // TODO 自动生成的方法存根
        return super.activeRecord(dataSource, arg);
    }

    @Override
    public CommandResult afterInit(GOT_Form form, CommandArg arg) {
        // TODO 自动生成的方法存根
        return super.afterInit(form, arg);
    }

    @Override
    public CommandResult radioItemSelect(FormItem_Plugin radioPlugin, CommandArg arg) {
        // TODO 自动生成的方法存根
        return super.radioItemSelect(radioPlugin, arg);
    }
}

```

【优化】 针对OS里面的应用、插件、扩展的安装 进行深度优化

【修复】 GOT插件-修复TableField上设置AllowEdit为Always, 数据源设置never, 但是在FormItem上还是显示Table上的Always

【修复】 GOT插件-修复层链升级要支持增量部分的提升

【修复】 GOT插件-修复数据源改BigDecimal的显示小数位数后, 数据源里没有变化, 右键table进行同步也没修改的问题

【修复】 GOT插件-修复从A表将字段拖入到B表中, StringSize属性没有带入到B表中的问题

【修复】 GOT插件-修复ext的表调整创建层, 调整至ent, java代码存在异常的问题

【修复】 GOT插件-修复将TableField往下拖动到某个索引下的两个索引值之间, 报错的问题

【修复】 GOT插件-修复ent层提升sys层能够提升, 但是回滚报错问题

【修复】 GOT插件-修复层升级-升级操作, plg在ext提升的画面, 查找文件路径不对的问题

【修复】 GOT插件-修复进入插件，got点击错的表，不会弹出画面的问题

【修复】 GOT插件-表字段改名后，索引没有跟着改的问题

【优化】 客户端框架UI全面升级优化

【修复】 客户端框架修复了部分已知bug

2023-02-10 更新说明

【新增】 OS更新APP时，前端仍在线操作的，友好提示，说明APP在更新，请等待。

【修复】 数据源设置了left join关系，且relation设置了Fixed类型，打开画面报错问题。

【新增】 DialogFieldGroup支持设置layout属性

【修复】 通过plg修改app的表，新加的字段，画面上点排序会报错问题。

【修复】修复result.fileupload方法上传文件url带.会报错问题。

【修复】Left join 两表关联 之后 关联表加空值条件 SQL 语句报错问题。

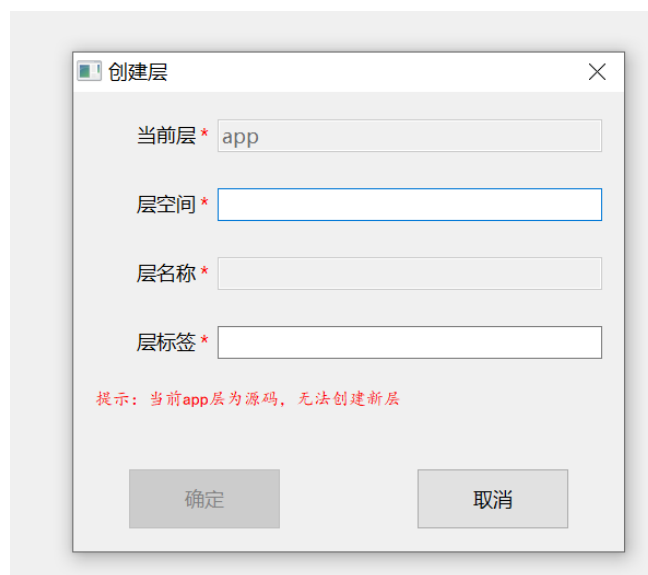
【新增】无数据源的String类型的FormItem，当StringSize为零时，输入字符后，不检查长度。

【新增】新增报表入口，工具栏【报表】



报表

【新增】层空间、层名称配置

A screenshot of a 'Create Layer' (创建层) dialog box. It contains four input fields: 'Current Layer' (当前层) with 'app' entered, 'Layer Space' (层空间), 'Layer Name' (层名称), and 'Layer Tag' (层标签). All fields are marked with a red asterisk. Below the fields is a red提示 (提示) message: '提示: 当前app层为源码, 无法创建新层' (提示: 当前app层为源码, 无法创建新层). At the bottom are '确定' (Confirm) and '取消' (Cancel) buttons.

创建层

当前层 * app

层空间 *

层名称 *

层标签 *

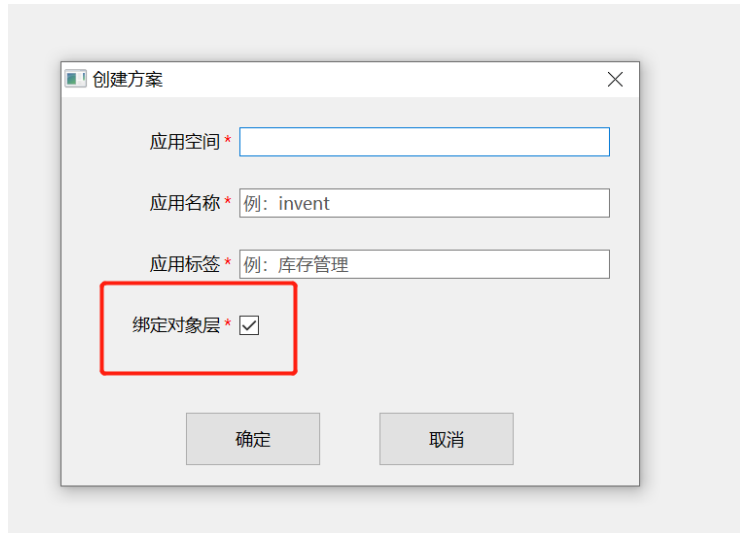
提示: 当前app层为源码, 无法创建新层

确定 取消

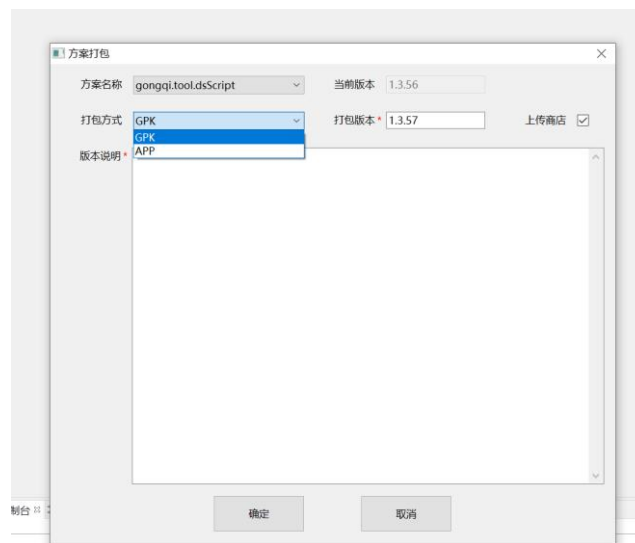
【新增】查看层配置



【新增】 绑定对象层



【新增】打包方案、插件 优化



【新增】导入Excel 支持 dateTime 类型

【新增】支持硕正报表后台导出PDF/JPG等格式

2022-09-15 更新说明

【新增】维度项支持建立 DataSourceRelation

Relation	
ChildTable. ItemDim. Color=ParentTable. Config	
Name	Value
Field	ItemDim. Color
ParentField	Color

【优化】Event事件处理异常消息-优先提示ERPException

【新增】多种方法 支撑 代码内多线程执行

同步执行无返回值

```
ServiceDispatcher.runAsyncService("dynamic", "admin", new Runnable() {  
    @Override  
    public void run() {  
        SysUser find = SysUser.util().find(SysUser.class, "admin");  
        System.out.println(find.castToMap().toString());  
    }  
});
```

异步执行无返回值

```
ServiceDispatcher.runService("dynamic", "admin", new Runnable() {  
    @Override  
    public void run() {  
        SysUser find = SysUser.util().find(SysUser.class, "admin");  
        System.out.println(find.castToMap().toString());  
    }  
});
```

同步执行有返回值

```
ServiceDispatcher.supplyAsyncService("dynamic", "admin", new Supplier<String>() {  
    @Override  
    public String get() {  
        return null;  
    }  
});
```

异步执行有返回值

```
ServiceDispatcher.supplyService("dynamic", "admin", new Supplier<String>() {  
    @Override  
    public String get() {  
        return null;  
    }  
});
```

【新增】HtmlRefesArg新增方法 changeURL(url)

```
HtmlRefreshArg.changeURL("url");
```

callIFrameMethod(methodName, params)

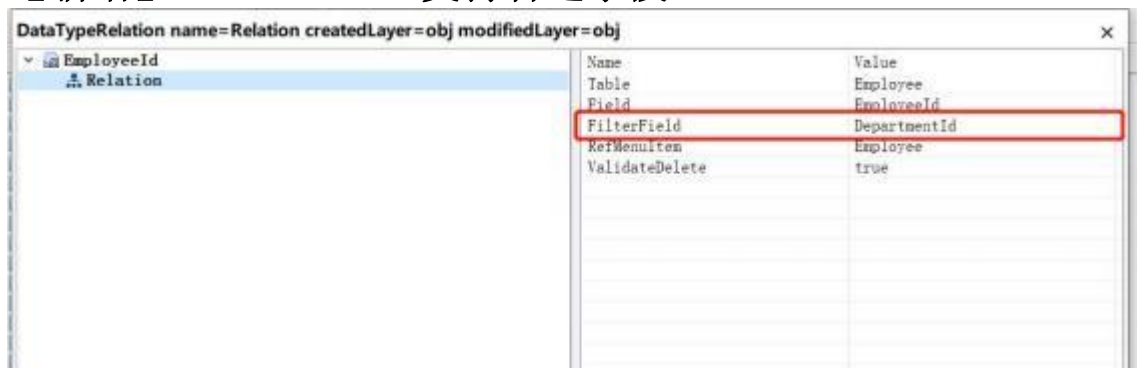
```
HtmlRefreshArg.callIFrameMethod("methodName", new Object[] {});
```

【修复】SQLServer下对日期时间类型数据的过滤排序异常处理

【修复】debug模式为false 时 beforeInitEvent 事件没触发

【修复】获取无数据源的EnumEditor字段值异常问题

【新增】FilterField 支持普通字段



【优化】日期时间类型字段为null时 toString() 方法统一返回 空字符串、getTime() 方法统一返回 -1

【优化】对JSON 序列化 建议使用 JSONUtils

【新增】可被跨应用调用的方法 需要在方法上写注解 @ApplicationMethod 否则将调用不到

【新增】在config.xml <general-configuration> 下面可配置项目环境基础APP 不配置 默认为: gongqi.df.database

```
1 <?xml version="1.0" encoding="UTF-8"?>
2
3<framework-configuration>
4<general-configuration>
5<property name="gongqi.framework.core.debug">false</property>
6<property name="gongqi.framework.data.stateless_session">true</property>
7<property name="gongqi.framework.security.httpsession">true</property>
8<property name="gongqi.framework.web.jsoncompress_length">102400</property>
9<property name="gongqi.framework.query.wildcard_character">*</property>
10<property name="gongqi.framework.core.maintenance">true</property>
11<property name="gongqi.framework.objects.solutionname">gongqi.df.basedata</property>
12</general-configuration>
13<session-configuration>
14<property name="gongqi.framework.security.session.type">Local</property>
15<property name="gongqi.framework.security.session.redis.hostname">127.0.0.1</property>
16<property name="gongqi.framework.security.session.redis.port">6379</property>
17<property name="gongqi.framework.security.session.redis.password"></property>
18</session-configuration>
19<logging-configuration>
20<property name="gongqi.framework.logging.enable">false</property>
```

【修复】requestUUID 和 gongqiForm 日志不对应的关系

【修复】plg提升app的表并新增字段，再安装plg后新增字段不及时生效的问题

【新增】项目启动时增加对 .xml 和.class文件进一步校验

【新增】方案卸载前事件

【新增】注解 @ScheduleJob (value= “名称” , autoImport=true,cron=”) 用于 定时任务 下拉选择和 自动导入

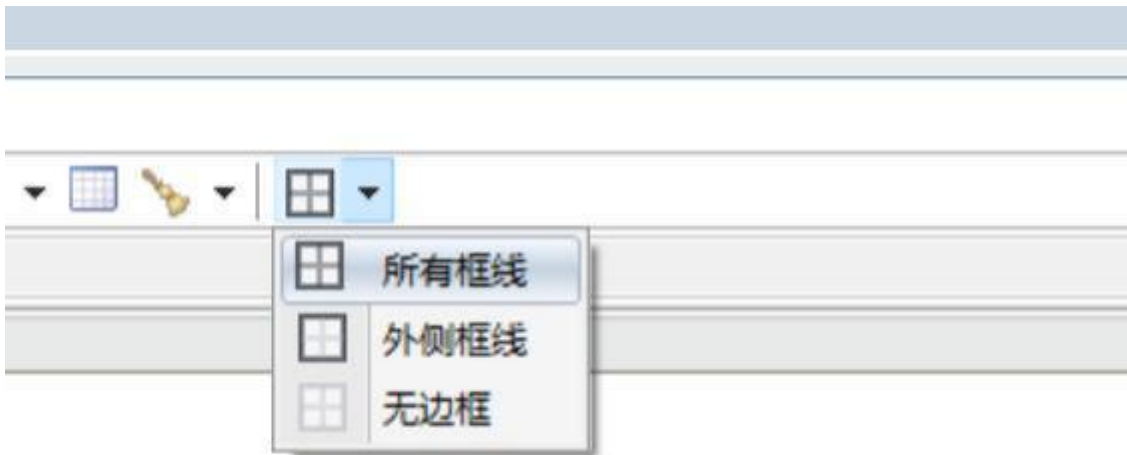
【新增】plg层提升app表后 modifiedField_ 方法写法
modifiedField_plgId\$字段Id

【修复】导入控制台 报错信息 单元格【0】 改为单元格A0
日期-单元格格式，如果是文本 日期 支持 -和/ ，数值类型支持保留几位小时，优化新建导入 新建字段序号，排序，修复 dateTime 的导入异常

【新增】 报表中心 增加导入内置报表 功能，开发人员可在在 resource 目录下 建 reports 目录 然后把报表存入开发好的.report文件



【新增】 设计报表 边框样式



【新增】 报表新增函数 `singleParmValue(entityName(单例表名称), filedName(字段英文名))`

【新增】 导入控制台 导入详情，提示必填

2022-03-03 更新说明

【新增】新增 APP 首页功能

可在 Application 与 Extension 可复写方法 onClientAfterLoading() 之后可执行 openForm 命令

包资源管理器

- Client
- GongqiERP
- LayerProject
 - layer_obj
 - app_gongqi.erp.mrp
 - app_gongqi.erp.qc
 - gongqi.erp.qc.layers.app
 - classes
 - Application.java
 - datatypes
 - enums
 - forms
 - imports
 - jobs
 - maps.base
 - menuitems
 - queries
 - reports
 - services
 - tables

Application.java

```
1
2 package gongqi.erp.qc.layers.app.core;
3
4 import gongqi.erp.gotmodelform.remote.CommandResult;
5 import gongqi.erp.gotmodelform.remote.OpenFormArg;
6
7 /**
8  * {@link Application} 是应用的定义类，用于管理应用的初始化与销毁逻辑。
9  */
10 public class Application extends gongqi.erp.framework.core.solution.Solution {
11
12
13     @Override
14     public CommandResult onClientAfterLoading() {
15         CommandResult result = new CommandResult();
16         OpenFormArg arg = new OpenFormArg();
17         arg.menuItemId = "";
18         result.openForm(arg);
19         return result;
20     }
21 }
```

【新增】新增 APP 业务导图

业务导图 现 可配至 App

业务导图

应用编号	名称	描述
1	gongqi.erp.basedata	001 123
2		

应用编号

1

gongqi.message.engine

2

gongqi.erp.subreqpurch

3

gongqi.erp.subpurch

4

gongqi.erp.sales

5

gongqi.erp.reqpurch

6

gongqi.erp.qc

7

gongqi.erp.purch

8

gongqi.erp.prod

9

gongqi.erp.arp

10

gongqi.erp.master

11

gongqi.erp.inventAccoun

12

gongqi.erp.invent

13

gongqi.erp.custinvoice

14

gongqi.erp.basedata

- 用户设置
- 编号序列
- 消息通知

【新增】账户安全说明



【新增】SysTmp 新增工具类 提供 newRecordMap() 方法

```
14
15= @Override
16 public CommandResult afterInit(CommandArg arg) {
17     CommandResult result = new CommandResult();
18     Map<String, Object> newRecordMap = SysTmp.util().newRecordMap();
19     newRecordMap.put("name", "张三");
20     return result;
21 }
22
23
```

【新增】新增 AfterInitEvent 事件

【优化】优化主表校验的提示，增加对 FilterFiled 的提示

【优化】优化某表存在多个当前表主表字段时的字段校验

【优化】优化唯一键值在 SQLServer 内的报错提示

【优化】优化 GOT 内操作的流畅性和稳定性

【优化】对原 clientCommandAfterLogin 进行作废过期处

理**【优化】**账户密码强度最低要求从 3 位提升到 4 位

【优化】修改部分已知 BUG

新 framework 连 sqlServer 启动报错 自动添加了参数
useOldAliasMetadataBehavior=true

GOT 启动错误显示具体错误信息

当表上有字段取名为“SysActive”时（未继承 SysActive），当数据被引用时，还是会检查数据启用的问题。

主表两次 leftjoin 子表报错

...

...

具体可查看 任务管理系统

2022-01-25 更新说明

【新增】 账户安全

- 1.1 设置密码强度校验，密码强度设置三级： 高、 中、低

高：

密码长度必须大于 8 位；
必须包含大小写字母及数字；
不能包含用户名

中：

密码长度必须大于 6 位；
必须包含字母及数字；

低：

密码长度必须大于 3 位

- 1.2 设置密码最长使用期限(天) :超过使用期限系统自动提醒 修改密码

- 1.3 设置登录失败锁定阈值(次数)：登录失败多次将锁定账户 禁止登录

- 1.4 设置登录失败锁定时间

- 1.5 设置新用户初始密码 123456

- 1.6 设置新用户首次登录且为默认密码时 必须修改密码

- 1.7 设置用户锁定： 管理员新增 锁定用户 和 解锁用户 功能（锁定：禁止用户登

录）

【新增】服务端新增监听 Table 操作事件

```
ApplicationListener<GongqiRecordAfterInitEvent>  
ApplicationListener<GongqiRecordAfterInsertEvent>  
ApplicationListener<GongqiRecordAfterUpdateEvent>  
ApplicationListener<GongqiRecordAfterDeleteEvent>  
ApplicationListener<GongqiRecordBeforeInitEvent>  
ApplicationListener<GongqiRecordBeforeInsertEvent>  
ApplicationListener<GongqiRecordBeforeUpdateEvent>  
ApplicationListener<GongqiRecordBeforeDeleteEvent>
```

如：

```
public class RecordOnAfterInitListener implements ApplicationListener<GongqiRecordAfterInitEvent> {  
    @Override  
    public void onApplicationEvent(GongqiRecordAfterInitEvent event) {  
        GongqiRecord gongqiRecord = event.getGongqiRecord(); // 获取工企Record  
        String tableId = event.getTableId(); // 获取 表ID  
    }  
}
```

特殊：框架提供特殊类 GongqiRecordEventListener 可整体实现所有 table 监听事件

```
public class GongqiRecordEventListenerImpl extends GongqiRecordEventListener {  
    //删除记录 在table上super.delete()之后执行  
    @Override  
    protected void onAfterDelete(GongqiRecordAfterDeleteEvent event) {  
    }  
    //初始化记录 在table上super.init()之后执行  
    @Override  
    protected void onAfterInit(GongqiRecordAfterInitEvent event) {  
    }  
    //新增记录 在table上super.insert()之后执行  
    @Override  
    protected void onAfterInsert(GongqiRecordAfterInsertEvent event) {  
    }  
    //修改记录 在table上super.update()之后执行  
    @Override  
    protected void onAfterUpdate(GongqiRecordAfterUpdateEvent event) {  
    }  
    //删除记录 在table上super.delete()之前执行  
    @Override  
    protected void onBeforeDelete(GongqiRecordBeforeDeleteEvent event) {  
    }  
    //初始化记录 在table上super.init()之前执行  
    @Override  
    protected void onBeforeInit(GongqiRecordBeforeInitEvent event) {  
    }  
    //新增记录 在table上super.insert()之前执行  
    @Override  
    protected void onBeforeInsert(GongqiRecordBeforeInsertEvent event) {  
    }  
    //新增记录 在table上super.update()之前执行  
    @Override  
    protected void onBeforeUpdate(GongqiRecordBeforeUpdateEvent event) {  
    }  
}
```

【新增】服务端新增监听无数据源下拉事件

ApplicationListener<FormLookupEvent>

```
public class FormFormLookupEvent implements ApplicationListener<FormLookupEvent>{  
    @Override  
    public void onApplicationEvent(FormLookupEvent event) {  
        CommandArg arg = event.getCommandArg();//获取 CommandArg  
        String editorName = event.getEditorName();//获取 下拉字段名称  
        String formId = event.getFormId();// 获取画面Form 编号ID  
        QueryTable queryTable = event.getQueryTable();// 获取 QueryTable  
        Map<String, Object> editors = event.getEditors();// 获取 下拉字段列表  
    }  
}
```

【新增】服务端新增监听有数据源下拉事件

ApplicationListener<FormDataSourceLookupEvent>

```
public class FormDataSourceLookupListener implements ApplicationListener<FormDataSourceLookupEvent>{  
    @Override  
    public void onApplicationEvent(FormDataSourceLookupEvent event) {  
        CommandArg commandArg = event.getCommandArg();// 获取 CommandArg  
        String fieldName = event.getFieldName();// 获取 字段名称  
        String formDataSourceId = event.getFormDataSourceId();// 获取 数据源编号ID  
        QueryTable queryTable = event.getQueryTable();// 获取 QueryTable  
        GongqiRecord record = event.getRecord();// 获取 当前记录  
        Object source = event.getSource();// 获取数据源  
    }  
}
```

【新增】服务端新增监听客户端选中记录事件

ApplicationListener<ClientActiveRecordEvent>

```
public class ClientActiveRecordEventListener implements ApplicationListener<ClientActiveRecordEvent> {  
    @Override  
    public void onApplicationEvent(ClientActiveRecordEvent event) {  
        CommandArg commandArg = event.getCommandArg(); // 获取 CommandArg  
        CommandResult commandResult = event.getCommandResult(); // 获取 CommandResult  
        String formDataSourceId = event.getFormDataSourceId(); // 获取 数据源ID  
        String formId = event.getFormId(); // 获取 画面ID  
    }  
}
```

【删除】删除底层 MixedServiceImpl 下 部分 injection

开头的方法（原来用于全局性事件的监听）

```
injectionBeforeInit  
injectionClientActiveRecord  
injectionModifiedField  
injectionLookup  
injectionLookup  
injectionExecuteQuery  
injectionSaveRecord  
injectionDeleteRecord  
injectionInitRecord
```

【优化】GenericInvoke 类添加到 Framework jar 包里面

```
GenericInvoke.getClassesInvoker(namespaceId, className);
```

【新增】新增 PLG 注入方法

按钮注入：支持注入到 APP 的画面上， 按钮包括：

普通按钮、 按钮组、 按钮 Menu

菜单注入： 支持注入菜单到 APP Menu 内

```
public void initialize() {  
    new ButtonDefinitionBuilder(Form_FormTest.class, "ButtonA").setLabel("注入按钮A").addFormItemMethod(A_Class.class, "buttonOK_ButtonA", CommandArg.class)  
        .register();  
    |  
    new ButtonDefinitionBuilder(Form_FormTest.class, "ButtonB").setLabel("注入按钮B").addFormItemMethod(A_Class.class, "buttonOK_ButtonB", CommandArg.class)  
        .register();  
  
    new MenuItemButtonDefinitionBuilder(Form_FormTest.class, "MenuItemA")  
        .setMenuItem(MenuItem_SysUser.class)  
        .setLabel("注入MenuItemButton").register();  
  
    ButtonGroupDefinitionBuilder buttonGroupDefinitionBuilder = new ButtonGroupDefinitionBuilder(Form_FormTest.class, "BG001");  
    buttonGroupDefinitionBuilder.obtainButtonDefinitionBuilder("ButtonA2").setLabel("BGB01");  
    buttonGroupDefinitionBuilder.obtainButtonDefinitionBuilder("ButtonB2").setLabel("BGB02");  
    buttonGroupDefinitionBuilder.register();  
  
    ButtonMenuDefinitionBuilder buttonMenuDefinitionBuilder = new ButtonMenuDefinitionBuilder(Form_FormTest.class, "BM001");  
    buttonMenuDefinitionBuilder.setLabel("按钮菜单");  
    buttonMenuDefinitionBuilder.obtainButtonDefinitionBuilder("ButtonA3").setLabel("按钮一");  
    buttonMenuDefinitionBuilder.obtainButtonDefinitionBuilder("ButtonB3").setLabel("按钮二");  
    buttonMenuDefinitionBuilder.register();  
  
    new MenuRefMenuItemDefinitionBuilder("gongqi.erp.basedata.60400010", "菜单").setLabel("菜单内容").setMenuItem(MenuItem_A_Form.class).register();  
}
```

