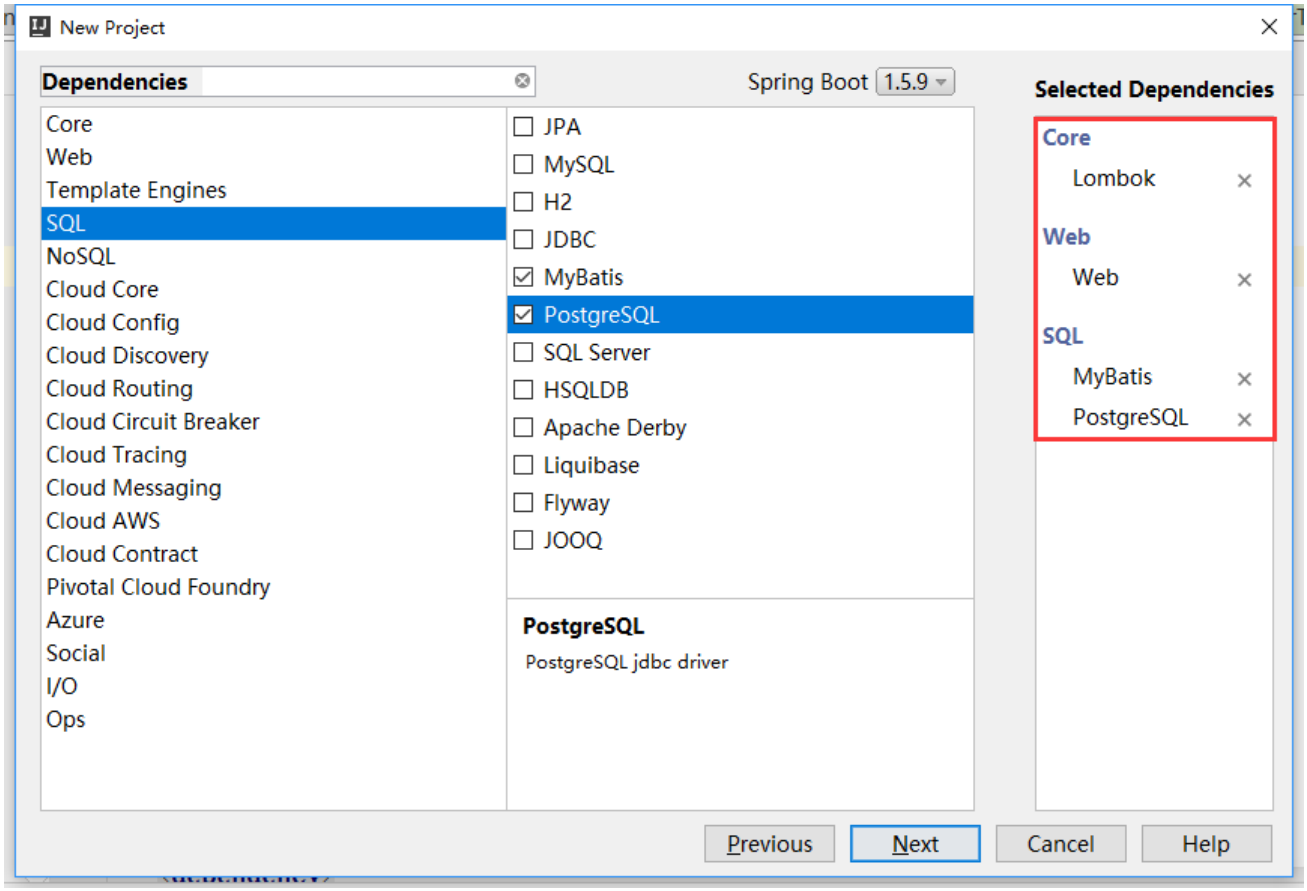
EasyPoi简单使用指南

EasyPoi是什么

EasyPoi功能就是容易让一个没见接触过POI的人员就可以方便的写出Excel导出,Excel模板导出,Excel导入,Word模板导出,通过简单的注解和模板语言(熟悉的表达式语法),完成POI原生复杂的写法。

EasyPoi使用

创建一个Spring Boot工程study-easypoi，因为需要查询数据库的数据，需要加入以下几个模块。



在工程中增加如下依赖。

```
<!-- 父包 -->
<dependency>
 <groupId>cn.afterturn</groupId>
 <artifactId>easypoi</artifactId>
 <version>3.0.3</version>
 <type>pom</type>
</dependency>


<!-- 导入导出的工具包,可以完成Excel导出,导入,Word的导出,Excel的导出功能 -->
<dependency>
 <groupId>cn.afterturn</groupId>
 <artifactId>easypoi-base</artifactId>
 <version>3.0.3</version>
</dependency>

<!--  耦合了spring-mvc 基于AbstractView,极大的简化spring-mvc下的导出功能 -->
```

```
<dependency>
 <groupId>cn.afterturn</groupId>
 <artifactId>easypoi-web</artifactId>
 <version>3.0.3</version>
</dependency>

<!-- 基础注解包,作用于实体对象上,拆分后方便maven多工程的依赖管理 -->
<dependency>
 <groupId>cn.afterturn</groupId>
 <artifactId>easypoi-annotation</artifactId>
 <version>3.0.3</version>
</dependency>
```

EasyPoi注解

@Excel：作用到filed上面,是对Excel一列的一个描述。
@ExcelCollection：表示一个集合,主要针对一对多的导出，比如一个老师对应多个科目,科目就可以用集合表示。
@ExcelEntity：表示一个继续深入导出的实体,但他没有太多的实际意义，只是告诉系统这个对象里面同样有导出的字段。
@ExcelIgnore：和名字一样表示这个字段被忽略跳过这个导导出。
@ExcelTarget：这个是作用于最外层的对象,描述这个对象的id，以便支持一个对象可以针对不同导出做出不同处理。

@Excel注解

@Excel是必须使用的注解，如果需求简单只使用这一个注解也是可以的,涵盖了常用的Excel需求，需要大家熟悉这个功能，主要分为基础、图片处理、时间处理、合并处理几块，name_id是上面讲的id用法。

属性	类型	默认值	功能
name	String	null	列名，支持name_id
needMerge	boolean	fasle	纵向合并单元格
orderNum	String	"0"	列的排序,支持name_id
replace	String[]	{}	值的替换 导出是{a_id,b_id} ， 导入反过来
savePath	String	"upload"	导入文件保存路径,如果是图片可以填写,默认是upload/className/ IconEntity这个类对应的就是upload/Icon/
type	int	1	导出类型 1 是文本；2 是图片；3 是函数；10 是数字 默认是文本
width	double	10	列宽
height	double	10	列高,后期打算统一使用@ExcelTarget的height,这个会被废弃,注意
isStatistics	boolean	fasle	自动统计数据，在追加一行统计，把所有数据的和输出 这个处理会吞没异常,请注意这一点
isHyperlink	boolean	false	超链接,如果是需要实现接口返回对象
isImportField	String	true	校验字段，看看这个字段是不是导入的Excel中有，如果没有说明是错误的Excel，读取失败，支持name_id
exportFormat	String	""	导出的时间格式，以这个是否为空来判断是否需要格式化日期
importFormat	String	""	导入的时间格式，以这个是否为空来判断是否需要格式化日期
format	String	""	时间格式,相当于同时设置了exportFormat 和 importFormat
databaseFormat	String	""	导出时间设置，如果字段是Date类型则不需要设置，数据库如果是string 类型，需要设置这个数据库格式，用以转换时间格式输出。
numFormat	String	""	数字格式化，参数是Pattern，使用的对象是DecimalFormat
imageType	int	1	导出类型 1 从file读取 2 是从数据库中读取 默认是文件 同样导入也是一样的
suffix	String	""	文字后缀,如% 90 变成90%
isWrap	boolean	true	是否换行 即支持\n
mergeRely	int[]	{}	合并单元格依赖关系，比如第二列合并是基于第一列 则{1}就可以了
mergeVertical	boolean	fasle	纵向合并内容相同的单元格

@ExcelTarget注解

@ExcelTarget限定一个导出实体的注解，以及一些通用设置，作用于最外面的实体。

属性	类型	默认值	功能
value	String	null	定义ID

@ExcelEntity注解

@ExcelEntity标记是不是导出excel标记为实体类，一般是一个内部属性类，标记是否继续穿透，可以自定义内部id。

属性	类型	默认值	功能
id	String	""	定义ID
name	String	""	设置一级表头的名称
show	boolean	fasle	是否显示两层表头，如果为true，必须设置name的值，name的值为一级表头

@ExcelCollection注解

@ExcelCollection一对多的集合注解，用于标记集合是否被导出以及集合的整体排序。

属性	类型	默认值	功能
id	String	""	定义ID
name	String	null	定义集合列名,支持nanm_id
orderNum	int	0	排序,支持name_id
type	Class<?>	ArrayList.class	导入时创建对象使用

@ExcelIgnore注解

@ExcelIgnore忽略这个属性，被该注解标注的属性不导出，多使用需循环引用中。

注解导出Excel

简单实体导出（仅使用@Excel注解）

下面介绍如何使用注解导出数据库表中的数据，首先定义一个Menu类，并使用相关的注解。

Menu类：

```
@Getter
@Setter
public class Menu {

    @Excel(name = "菜单ID", height = 10, width = 15, isImportField = "true_st")
    private Integer mid;

    @Excel(name = "菜单名称", height = 10, width = 15, isImportField = "true_st")
    private String menuName;

    @Excel(name = "父级菜单", height = 10, width = 15, isImportField = "true_st")
    private Integer menuParentId;

    @Excel(name = "菜单层级", height = 10, width = 15, isImportField = "true_st")
```

```
    private String menuLevel;

    @Excel(name = "菜单路径", height = 10, width = 15, isImportField = "true_st")
    private String menuPath;

    @Excel(name = "创建时间", databaseFormat = "yyyyMMddHHmmss", format = "yyyy-MM-dd")
    private Date createDate;

}
```

数据查询接口：

查询数据库的语句为，此处使用的是mybatis-spring-boot-starter模块的写法

```
@Select("SELECT dm.* FROM domp_menu dm ORDER BY menu_level , menu_sort limit 20  ")
@Results({
        @Result(property = "mid",  column = "mid", javaType = Integer.class , jdbcType= JdbcType.NUMERIC),
        @Result(property = "menuName",  column = "menu_name", javaType = String.class),
        @Result(property = "menuParentId",  column = "menu_parent_id", javaType = Integer.class),
        @Result(property = "menuLevel",  column = "menu_level", javaType = String.class),
        @Result(property = "menuPath",  column = "menu_path", javaType = String.class),
        @Result(property = "createDate",  column = "create_date", javaType = Date.class , jdbcType = Jdbc
})
List<Menu> getMenu();
```

测试类：

```
@Test
public void testExcelExport4(){
    List<Menu> menuList=menuMapper.getMenu();//查询数据
    ExportParams exportParams=new ExportParams();//定义导出参数
    exportParams.setSheetName("菜单信息");
    //将数据和参数转为为POI的Workbook对象
    Workbook workbook = ExcelExportUtil.exportExcel(exportParams ,Menu.class, menuList);
    exportExcelFile(workbook);//导出
}

public static void exportExcelFile(Workbook workbook){
    FileOutputStream os=null;
    try {
        os= new FileOutputStream( name: "G:\\easypoi\\test.xls");
        workbook.write(os);
        os.flush(); os.close();
        workbook.close();
    } catch (FileNotFoundException e) {
        e.printStackTrace();
```

导出的数据：

菜单ID	菜单名称	父级菜单	菜单层级	菜单路径	创建时间
15	数据报表	0	1		2016-01-06
929261158	测试显示	0	1	1	2017-05-24
929261258	测试图标配置	0	1		2017-09-13
1063	移动报表	0	1		2016-03-24
929261159	测试显示2	0	1	2	2017-05-24
741715504	精准营销	0	1		2017-02-06
6	专题工具	0	1	x	2017-03-16
929261174	客服质检	0	1		2017-06-21
929261150	测试2223	0	1	1	2017-05-18
2	系统设置	0	1		2016-01-04
1085634120	报表权限	0	1		2017-10-25
929261096	测试1750	929261095	2	123	2017-03-22
929261309	生意参谋计算模型	6	2	http://dompapi.bluemoon.com.cn/	2017-12-27
929261175	质检首页	929261174	2		2017-06-21
1064	规划与组织	1063	2		2016-03-24

集合定义导出（同时使用@Excel、@ExcelEntity、@ExcelCollection注解）

学会了使用@Excel注解做最基本的导出后，下面在使用@ExcelEntity注解关联实体、@ExcelCollection导出集合属性复杂的表格。

下面导出的数据表的逻辑大体为：数据库有张用户表存储用户的基本信息，用户的登录表（存储密码），一个用户会有多个角色，一个角色会有多个菜单。

用户基本信息UserInfo类：

定义一个用户信息的类，注意标红的两个注解

```java
@Getter
@Setter
public class UserInfo {

    @Excel(name = "员工编号", needMerge=true, isImportField = "true_st" , isStatistics=true)
    private Integer id;

    @Excel(name = "用户名", needMerge=true,  isImportField = "true_st")
    private String userName;

    @Excel(name = "手机号码",needMerge=true,  isImportField = "true_st")
    private String mobile;

    @ExcelEntity(id = "user" , name = "用户密码", show = true)
    private User user;

    @ExcelCollection(name = "角色信息", orderNum = "0" ,type=Role.class)
    private List<Role> roleList;
}
```

用户登录信息User类：

```java
@ExcelTarget(value = "user")
@Getter
@Setter
public class User {
    @Excel(name = "主键" ,needMerge=true, orderNum = "0")
    private Integer id;

    private String userId;

    @Excel(name = "密码" ,needMerge=true, orderNum = "1")
    private String password;
}
```

用户角色信息Role类：

```
@Getter
@Setter
public class Role {

    @Excel(name = "角色ID", needMerge=true,    isImportField = "true_st")
    private Integer id;

    @Excel(name = "角色名称", needMerge=true,    isImportField = "true_st")
    private String roleName;

    @Excel(name = "角色描述",needMerge=true,    isImportField = "true_st")
    private String roleDes;

    @ExcelCollection(name= "菜单" , type=Menu.class)
    private List<Menu> menuList;

}
```

角色下的Menu类：

```
public class Menu {
    @Excel(name = "菜单ID", height = 10, width = 15, isImportField = "true_st")
    private Integer mid;
    @Excel(name = "菜单名称", height = 10, width = 15, isImportField = "true_st")
    private String menuName;
    @Excel(name = "父级菜单", height = 10, width = 15, isImportField = "true_st")
    private Integer menuParentId;

    @Excel(name = "菜单层级", height = 10, width = 15, isImportField = "true_st")
    private String menuLevel;

    @Excel(name = "菜单路径", height = 10, width = 15, isImportField = "true_st")
    private String menuPath;

    @Excel(name = "菜单选中状态下的图标", type = 2 ,width = 60 , height = 90,imageType = 1)
    private String menuActiveIcon;

    @Excel(name = "创建时间", databaseFormat = "yyyyMMddHHmmss", format = "yyyy-MM-dd")
    private Date createDate;
```

查询数据库的用户信息Mapper UserInfoMapper

```java
@Mapper
public interface UserInfoMapper {

    /**
     * 查询
     * @return
     */
    @Select("SELECT * FROM domp_user_info WHERE id in ('80486838' , '80497024')")
    @Results({
            @Result(property = "id",  column = "id", javaType = Integer.class),
            @Result(property = "userName",  column = "user_name", javaType = String.class),
            @Result(property = "mobile",  column = "mobile", javaType = String.class),
    })
    List<UserInfo> findUserInfo();

}
```

查询数据库的用户信息Mapper UserMapper

```java
@Mapper
public interface UserMapper {

    /**
     *  查询
     * @return
     */
    @Select("SELECT * FROM domp_user WHERE user_id='${userId}' limit 1 ")
    @Results({
            @Result(property = "userId",  column = "login_name", javaType = String.class ),
            @Result(property = "password",  column = "pass_word", javaType = String.class)})
    User getByUserId(@Param("userId") Integer userId );

}
```

查询数据库的用户角色信息Mapper RoleMapper

```java
@Mapper
public interface RoleMapper {

    /**
     * 查询
     * @return
     */
    @Select("SELECT dr.* FROM domp_role dr ,  domp_role_user_ref drur WHERE dr.rid=drur.role_id  \n"
            "AND drur.user_id='${userId}' ORDER BY rid")
    @Results({
            @Result(property = "id",  column = "rid", javaType = Integer.class , jdbcType= JdbcType.NUMERIC),
            @Result(property = "roleName",  column = "role_name", javaType = String.class),
            @Result(property = "roleDes",  column = "role_des", javaType = String.class),
    })
    List<Role> findRole(@Param("userId") Integer userId);
}
```

查询数据库的角色下的菜单信息Mapper MenuMapper

```java
@Mapper
public interface MenuMapper {

    /**...*/
    @Select("SELECT dm.* FROM domp_menu dm , domp_role_menu_ref drmr WHERE dm.mid=drmr.menu_id\n" +
            "AND drmr.role_id=${roleId} ORDER BY menu_level , menu_sort limit 10  ")
    @Results({
            @Result(property = "mid",  column = "mid", javaType = Integer.class , jdbcType= JdbcType.NUMERIC),
            @Result(property = "menuName",  column = "menu_name", javaType = String.class),
            @Result(property = "menuParentId",  column = "menu_parent_id", javaType = Integer.class),
            @Result(property = "menuLevel",  column = "menu_level", javaType = String.class),
            @Result(property = "menuPath",  column = "menu_path", javaType = String.class),
            @Result(property = "menuActiveIcon",  column = "menu_icon_activated", javaType = String.class),
            @Result(property = "createDate",  column = "create_date", javaType = Date.class , jdbcType = JdbcTyp
    })
    List<Menu> findMenuByRoleId(@Param("roleId") Integer roleId);
```

测试类：

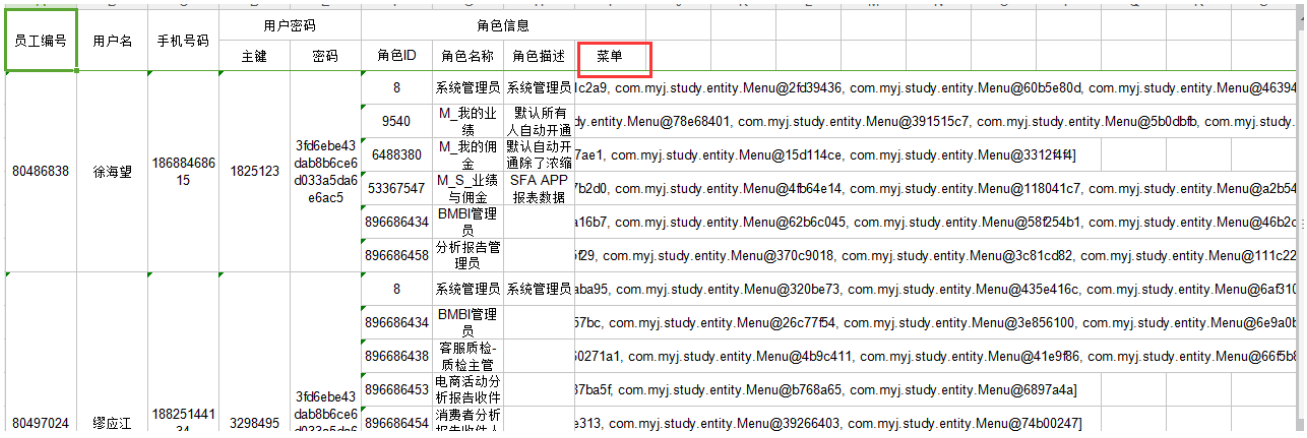
运行测试类

```java
    @Test
    public void testExcelExport2(){
        List<UserInfo> listUserInfo = userInfoMapper.findUserInfo();//查询用户信息
        for(UserInfo userInfo : listUserInfo){
            userInfo.setUser(userMapper.getByUserId(userInfo.getId()));//查询用户密码

            List<Role> roleList = roleMapper.findRole(userInfo.getId());//查询用户角色
            userInfo.setRoleList(roleList);
            for(Role role : roleList){
                List<Menu> menuList = menuMapper.findMenuByRoleId(role.getId());//查询用户菜单
                role.setMenuList(menuList);
            }
        }
        ExportParams exportParams=new ExportParams();
        exportParams.setSheetName("员工信息");
        Workbook workbook = ExcelExportUtil.exportExcel(exportParams ,UserInfo.class, listUserInfo);

        exportExcelFile(workbook);
    }
```

结果如下图：



员工编号	用户名	手机号码	用户密码		角色信息			
			主键	密码	角色ID	角色名称	角色描述	菜单
80486838	徐海望	18688468615	1825123	3fd6ebe43dab8b6ce6d033a5da6e6ac5	8	系统管理员	系统管理员	c2a9, com.myj.study.entity.Menu@2fd39436, com.myj.study.entity.Menu@60b5e80d, com.myj.study.entity.Menu@46394
					9540	M_我的业绩	默认所有人自动开通	y.entity.Menu@78e68401, com.myj.study.entity.Menu@391515c7, com.myj.study.entity.Menu@5b0dbfb, com.myj.study.
					6488380	M_我的佣金	默认自动开通除了浓缩	7ae1, com.myj.study.entity.Menu@15d114ce, com.myj.study.entity.Menu@3312f4f4]
					53367547	M_S_业绩与佣金	SFA APP报表数据	7b2d0, com.myj.study.entity.Menu@4fb64e14, com.myj.study.entity.Menu@118041c7, com.myj.study.entity.Menu@a2b54
					896686434	BMBI管理员		a16b7, com.myj.study.entity.Menu@62b6c045, com.myj.study.entity.Menu@58f254b1, com.myj.study.entity.Menu@46b2c
					896686458	分析报告管理员		f29, com.myj.study.entity.Menu@370c9018, com.myj.study.entity.Menu@3c81cd82, com.myj.study.entity.Menu@111c22
80497024	缪应江	18825144134	3298495	3fd6ebe43dab8b6ce6d033a5da6e6ac5	8	系统管理员	系统管理员	aba95, com.myj.study.entity.Menu@320be73, com.myj.study.entity.Menu@435e416c, com.myj.study.entity.Menu@6af310
					896686434	BMBI管理员		57bc, com.myj.study.entity.Menu@26c77f54, com.myj.study.entity.Menu@3e856100, com.myj.study.entity.Menu@6e9a0b
					896686438	客服质检-质检主管		0271a1, com.myj.study.entity.Menu@4b9c411, com.myj.study.entity.Menu@41e9f86, com.myj.study.entity.Menu@66f5b6
					896686453	电商活动分析报告收件		7ba5f, com.myj.study.entity.Menu@b768a65, com.myj.study.entity.Menu@6897a4a]
					896686454	消费者分析报告收件人		e313, com.myj.study.entity.Menu@39266403, com.myj.study.entity.Menu@74b00247]

发现用户信息下的菜单导出不对，用户>>多个角色>>多个菜单，怀疑代码问题，继续抛弃最外层用户信息测试。

只查询角色和菜单测试类：

```
@Test
public void testExcelExport3(){
    List<Role> roleList = roleMapper.findRole( userId: 80497024);
    for(Role role : roleList){
        List<Menu> menuList = menuMapper.findMenuByRoleId(role.getId());
        role.setMenuList(menuList);
    }
    System.out.println(JSONObject.toJSONString(roleList));
    ExportParams exportParams=new ExportParams();
    exportParams.setSheetName("角色菜单信息");
    Workbook workbook = ExcelExportUtil.exportExcel(exportParams ,Role.class, roleList);

    exportExcelFile(workbook);
}
```

运行测试结果：

```
@Test
public void testExcelExport3(){
    List<Role> roleList = roleMapper.findRole( userId: 80497024);
    for(Role role : roleList){
        List<Menu> menuList = menuMapper.findMenuByRoleId(role.getId());
        role.setMenuList(menuList);
    }
    System.out.println(JSONObject.toJSONString(roleList));
    ExportParams exportParams=new ExportParams();
    exportParams.setSheetName("角色菜单信息");
    Workbook workbook = ExcelExportUtil.exportExcel(exportParams ,Role.class, roleList);

    exportExcelFile(workbook);
}
```

结果如下图：

A	B	C	D	E	F	G	H	I	J
角色ID	角色名称	角色描述	菜单						
			菜单ID	菜单名称	父级菜单	菜单层级	菜单路径	菜单选中状态下的图标	创建时间
			15	数据报表	0	1			2016-01-0
			1063	移动报表	0	1			2016-03-2
			741715504	精准营销	0	1			2017-02-0

发现不是问题问题，说明只能支持两层表头

图片导出

在日常运作中不可避免的会遇到图片的导入导出，这里提供了两种类型的图片导出方式：

1、图片路径导出

```
@Excel(name = "公司LOGO", type = 2 ,width = 40 , height = 20,imageType = 1)
private String companyLogo;
```

表示type =2 该字段类型为图片，imageType=1 (默认可以不填)，表示从file读取，字段类型是个字符串类型 可以用相对路径也可以用绝对路径，绝对路径优先依次获取。

示例建上面类Menu类中的：

```
public class Menu {
    @Excel(name = "菜单ID", height = 10, width = 15, isImportField = "true_st")
    private Integer mid;
    @Excel(name = "菜单名称", height = 10, width = 15, isImportField = "true_st")
    private String menuName;
    @Excel(name = "父级菜单", height = 10, width = 15, isImportField = "true_st")
    private Integer menuParentId;

    @Excel(name = "菜单层级", height = 10, width = 15, isImportField = "true_st")
    private String menuLevel;

    @Excel(name = "菜单路径", height = 10, width = 15, isImportField = "true_st")
    private String menuPath;

    @Excel(name = "菜单选中状态下的图标", type = 2 ,width = 20 , height = 40,imageType = 1)
    private String menuActiveIcon;

    @Excel(name = "创建时间", databaseFormat = "yyyyMMddHHmmss", format = "yyyy-MM-dd")
    private Date createDate;
```

2、图片路径导出

@Excel(name = "公司LOGO", type = 2 ,width = 40 , height = 20,imageType = 1)
private byte[] companyLogo;

表示type =2 该字段类型为图片，imageType=2 ，表示从数据库或者已经读取完毕，字段类型是个字节数组直接使用，同时image类型的cell最好设置好宽和高，会百分百缩放到cell那么大，不是原尺寸。

注解导入Excel

有导出就有导入，基于注解的导入导出，配置上是一样的，只是方式反过来而已，比如类型的替换 导出的时候是1替换成男，2替换成女，导入的时候则反过来，男变成1 ，女变成2，时间也是类似导出的时候date被格式化成 2017-8-25 ，导入的时候2017-8-25被格式成date类型。

导出的基本参数如下：

属性	类型	默认值	功能
titleRows	int	0	表格标题行数,默认0
headRows	int	1	表头行数,默认1
startRows	int	0	字段真正值和列标题之间的距离 默认0
keyIndex	int	0	主键设置,如何这个cell没有值,就跳过 或者认为这个是list的下面的值 这一列必须有值,不然认为这列为无效数据
startSheetIndex	int	0	开始读取的sheet位置,默认为0
sheetNum	int	1	上传表格需要读取的sheet 数量,默认为1
needSave	boolean	false	是否需要保存上传的Excel
needVerfiy	boolean	false	是否需要校验上传的Excel
saveUrl	String	"upload/excelUpload"	保存上传的Excel目录,默认是 如 TestEntity 这个类保存路径就是 upload/excelUpload/Test/yyyyMMddHHmss* 保存名称上传时间五位随机数
verifyHanlder	IExcelVerifyHandler	null	校验处理接口,自定义校验
lastOfInvalidRow	int	0	最后的无效行数,不读的行数
readRows	int	0	手动控制读取的行数
importFields	String[]	null	导入时校验数据模板,是不是正确的Excel
keyMark	String	":"	Key-Value 读取标记,以这个为Key,后面一个Cell 为Value,多个改为ArrayList
readSingleCell	boolean	false	按照Key-Value 规则读取全局扫描Excel,但是跳过List读取范围提升性能 仅仅支持titleRows + headRows + startRows 以及 lastOfInvalidRow
dataHanlder	IExcelDataHandler	null	数据处理接口,以此为主,replace,format都在这后面

假如有如下的Excel,如何读取Excel的文件：

菜单ID	菜单名称	父级菜单	菜单层级	菜单路径	菜单选中状态下的图标	创建时间
15	数据报表	0	1			2016-01-06
929261158	测试显示	0	1	1		2017-05-24
929261258	测试图标配置	0	1			2017-09-13
1063	移动报表	0	1			2016-03-24
929261159	测试显示2	0	1	2		2017-05-24
741715504	精准营销	0	1			2017-02-06
6	专题工具	0	1	x		2017-03-16
929261174	客服质检	0	1			2017-06-21
929261150	测试2223	0	1	1		2017-05-18
2	系统设置	0	1			2016-01-04
1085634120	报表权限	0	1			2017-10-25
929261096	测试1750	929261095	2	123		2017-03-22
929261309	生意参谋计算模	6	2	http://dompapi.bl		2017-12-27
929261175	质检首页	929261174	2			2017-06-21
1064	规划与组织	1063	2			2016-03-24
929261084	测试317ff	741715504	2	123		2017-03-17
3	用户管理	2	2	https://dompapi.		2016-01-04
929261278	报表角色管理	1085634120	2	https://dompapi.		2017-10-25
929261321	营销决策	15	2			2018-01-22
929261235	消费者档案	741715504	2			2017-08-11

下面说下导入的基本代码，注解都在上面使用过

```
@Test
public void testImport1() {
    ImportParams params = new ImportParams();
    params.setHeadRows(1);
    long start = new Date().getTime();
    List<Menu> list = ExcelImportUtil.importExcel(
            new File( pathname: "G:\\easypoi\\menu.xls"),
            Menu.class, params);
    System.out.println("处理时间："+(new Date().getTime() - start));
    System.out.println("数据条数："+list.size());
    System.out.println("数据如下：");
    System.out.println(JSONObject.toJSONString(list));
}
```

运行测试后的结果如下：