`@tiamaes/cbb-excel` 使用手册

---

1. 指南

介绍

用户手动填写 Excel 表格数据, 在 Web 端将 .xlsx 文件上传, 服务端解析 .xlsx 中的数据并导入数据库是企业应用系统中比较常见的场景. 一般可以这样定义:

> *Excel*数据导入由管理员先配置好 *Excel* 模板与数据库表之间的映射关系，填报用户在*Excel*模板中录入好数据，并一键上传、数据入库的一种方式。

功能术语

- Excel 导入模板：由一个 Excel 文件模板和一个以上的导入规则构成。一个 Excel 文件模板对应系统中的一个 Excel 导入模板。
- 导入规则：根据 Excel 文件模块，定义 Excel 中指定 sheet 页的数据导入到哪个库、哪张表、哪些字段，以及延伸的一些功能设置。
- 值域范围：可选值列表的定义，用于验证 Excel 中某一列的数据必须在指定的可选值列表内

操作流程

1. 准备要导入的 Excel 文件模板。
2. 在数据库中创建表结构。
3. 在系统的 Excel 数据导入模块中，配置导入模板、导入规则，并将导入模板授权给相关用户。
4. 普通用户填写完 Excel 文件模板后，上传 Excel 文件。如果有错误查看错误日志，导出错误数据并根据错误日志重新填写、上传。

功能的限制

- 管理员需要先在数据库中创建好表结构，导入模板不会自动创建表结构。
- 在一个导入规则中，只能将一个 Excel sheet 页的数据导入到对应一张数据库表中。如果想将一个Sheet 的数据导入到多张数据库表中，可以设置多个导入规则的方式实现

为什么要使用 Excel 组件

以上是实现 Excel 导入功能的步骤, 虽然存在多种多样的技术架构, 但其实现方法基本都是相通的.整个流程不算复杂, 一个系统如果只有一两处导入功能, 实际开发维护的工作量也在可控范围内. 然而对于一个庞大的系统而言, 复杂度更高. 实际的业务系统中, 导入功能可能至少也有十几个页面, 如果按照传统方式实现, 存在大量的重复性工作, 大量的模板文件及配置也难以维护.

因此, 导入功能的集中管理和操作简化就很有必要了, Excel 组件由此而来.

快速上手

下载安装

项目根目录, 打开命令行, 执行以下命令:

```
npm install @tiamaes/cbb-excel
```

Vue 配置

由于是源代码发布, 需要在 `vue.config.js` 中配置 babel 转译:

```
transpileDependencies: [
    "@tiamaes/cbb-excel", // 无则添加
]
```

局部引入

Excel 组件并没有提供全局注册逻辑, 只需要局部引入即可.

1. 在项目源代码目录 `src/views/` 创建一个 .vue 页面文件, 路径以 `src/views/experiment/excelList.vue` 为例. -- 实际使用自行定义
2. 局部引入

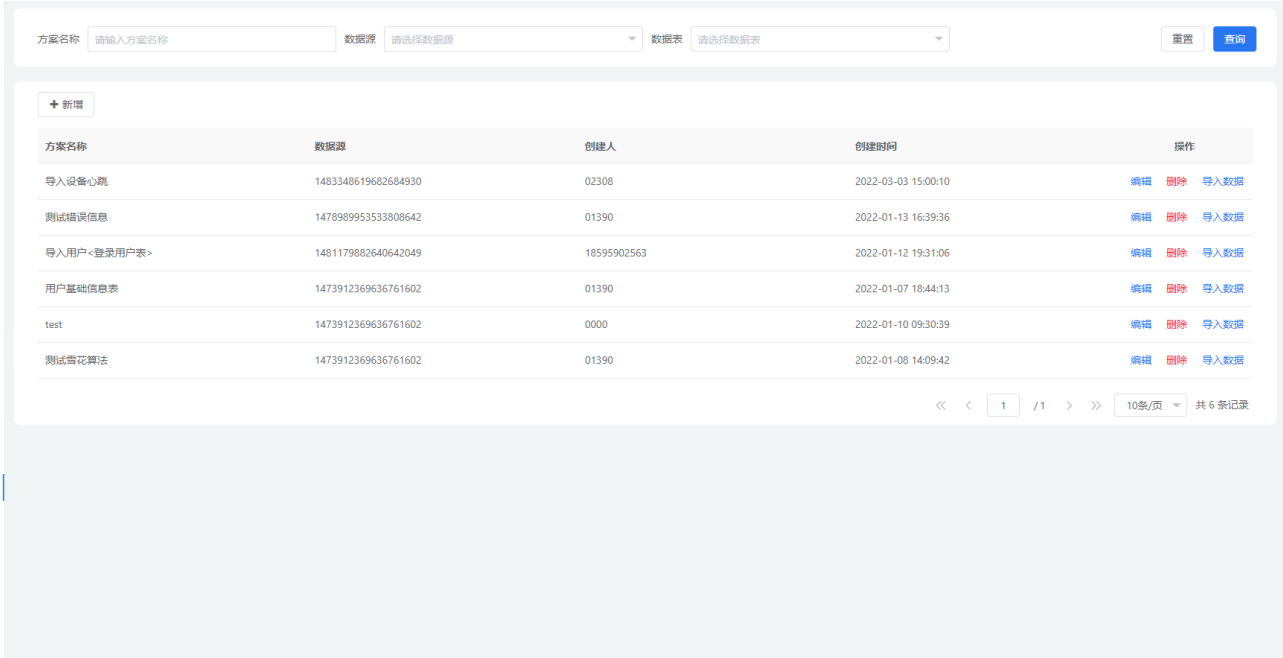
```
<template>
  <scheme-list></scheme-list>
</template>

<script>
import { SchemeList } from '@tiamaes/cbb-excel';
export default {
  components: {
    SchemeList
  },

  data() {
    return {
    }
  },

  methods: {

  }
}
</script>

<style lang="scss" scoped>

</style>
```

开始使用

要看到该页面, 需要先在系统中创建对应的菜单, 系统管理 > 资源管理:


修改资源
* 资源名称
表格导入
4/30
国际化
请输入国际化语言环境信息的键名路径
0/25
资源类型
菜单
* 访问路径
/experiment/excelList
21/400
资源路径
请输入资源路径
0/512
资源图标
请点击右侧设置图标选择资源图标
隐藏
否
缓存
否
业务扩展字段
请点击右侧设置图标编辑扩展字段
备注
请输入备注
0/100
取消
确定

列表页面如下：



2. 组件

SchemeList

导入方案列表页面, 负责导入方案的新增、修改 、删除等.

Api

属性

名称	说明	类型
custom	是否自定义编辑和导入行为	{edit: boolean, import: boolean}
appendOperations	操作列追加按钮, 配置同 `vxe-table`	array

事件

名称	说明
toolbar-add-click	点击新增按钮
operation-edit-click	点击操作列编辑按钮
operation-import-click	点击导入数据按钮

SchemeManage

导入方案的管理页面.

Api

属性

名称	说明	类型
viewId	方案 ID	string
viewIdKey	没有传递 viewId 时, 取路由参数中的哪个字段作为方案 ID	string

事件

名称	说明
save-success	保存成功
save-error	保存失败

SchemeView

数据导入页面, 接收方案 id, 展示出对应的模板, 用户下载模板文件后, 填充数据并上传文件后, 可以将数据插入数据库.

Api

属性

名称	说明	类型
viewId	方案 ID	string
viewIdKey	没有传递 viewId 时, 取路由参数中的哪个字段作为方案 ID	string

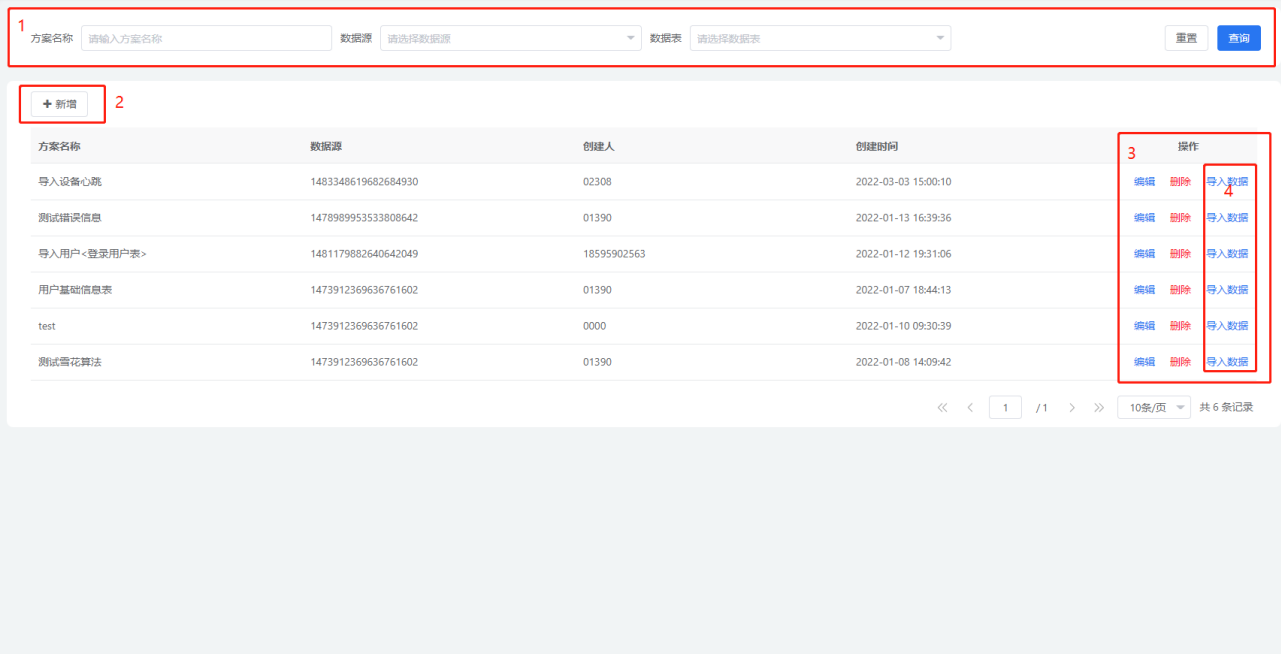
名称	说明	类型

事件

名称	说明
save-success	保存成功
save-error	保存失败

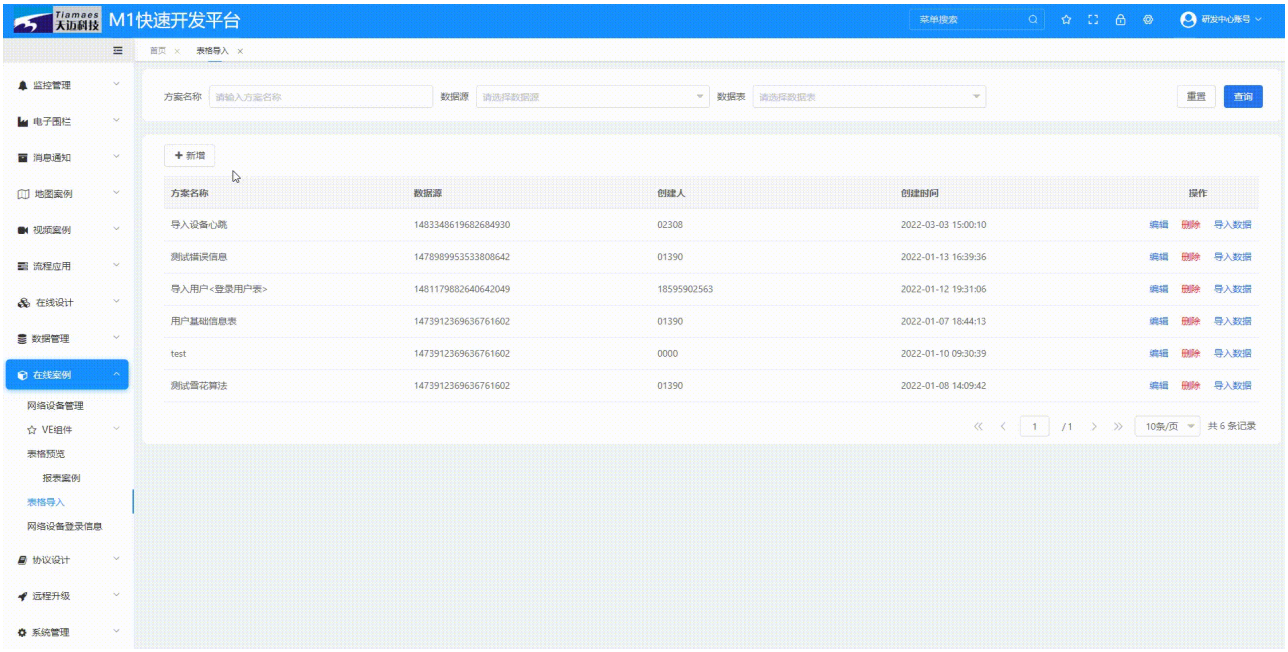
3. 使用手册

模板方案页面介绍

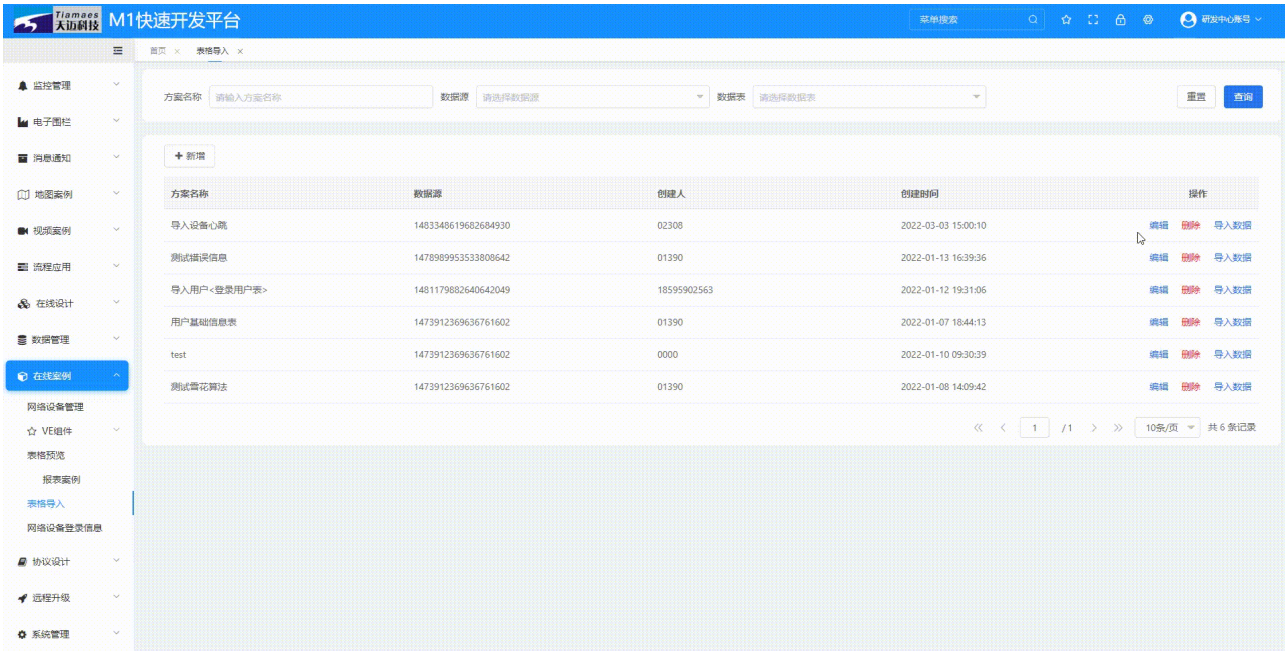


1. 查询区域, 根据条件筛选方案列表
2. 新增按钮, 默认点击打开方案管理页面
3. 操作区域, 包括编辑、删除、导入数据.
4. 导入数据按钮, 默认点击时, 以弹窗形式打开数据导入页面支持自定义

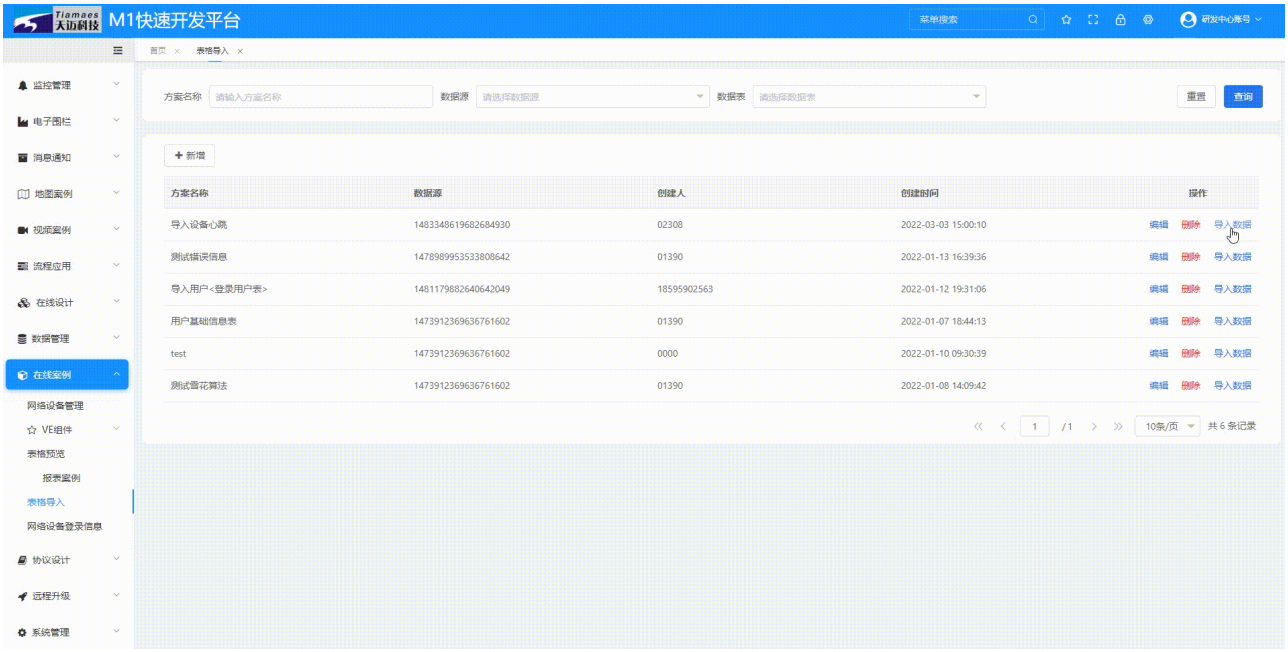
点击新增按钮,默认以弹窗形式打开模板方案编辑界面



点击编辑按钮,默认以弹窗形式打开方案编辑界面



点击导入数据按钮, 默认以弹窗形式打开导入数据界面



方案列表自定义

如果不想使用默认的弹窗交互形式, 可以 `custom` 属性和监听事件定制编辑方案和导入数据的行为.

一 定制模板方案编辑

> 从 *Excel* 组件开发期间的演示案例和 *ERP* 产线的使用情况来看, 定制方案编辑情景应该很少.

1. custom 属性开启编辑定制

```
custom: {
    edit: true, // 开启方案编辑定制
}
```

设置完成后, 点击新增、编辑按钮, 弹窗不再展示, 可以通过监听事件, 在项目的方案列表页面自行实现需要的处理逻辑.

2. 事件监听:

在项目方案列表页面的源代码中监听新增按钮点击、编辑按钮点击

```
<template>
  <scheme-list :custom="custom" @toolbar-add-click="add"
@operation-edit-click="edit"></scheme-list>
</template>
```

声明回调方法

```
methods: {
    add(...args) {
      console.log('add: ', args);
    },
    edit(...args) {
      console.log('edit: ', args);
    },
 }
```

3. 示例: 将弹窗改为 `el-drawer`

模板片段中增加 el-drawer, 并设置相关变量

```
<template>
  <div>
    <scheme-list :custom="custom" @toolbar-add-click="add"
@operation-edit-click="edit"></scheme-list>
    <el-drawer title="方案定制" :visible.sync="show" destroy-
on-close direction="rtl" size="50%">
      <scheme-manage></scheme-manage>
    </el-drawer>
  </div>
</template>
```

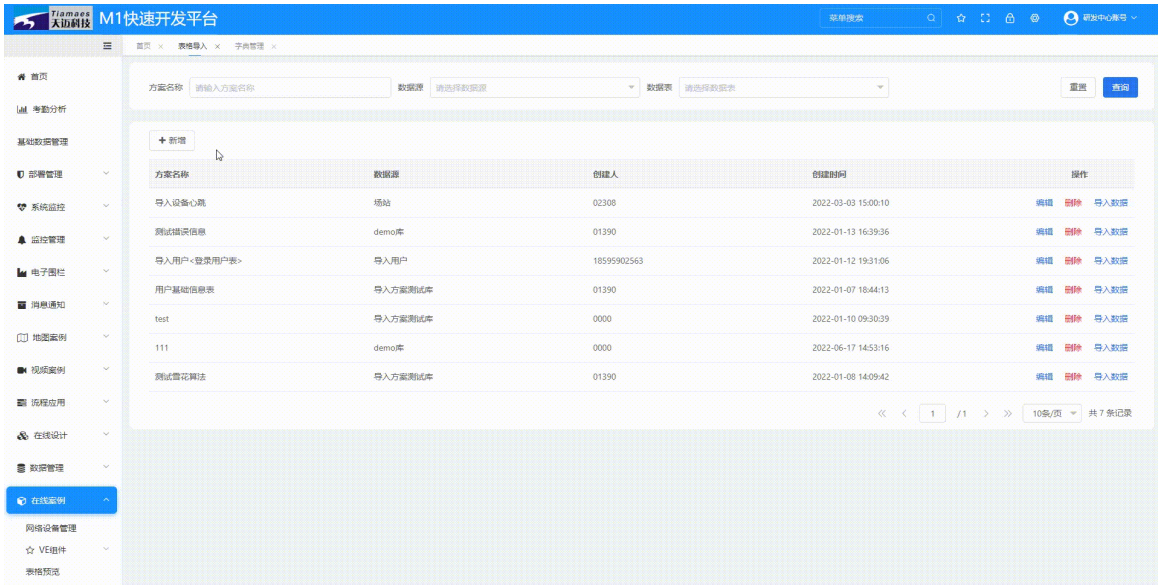
声明变量

```
data() {
    return {
        show: false,
    }
}
```

修改 add 方法

```
add(...args) {
      console.log("add: ", args);
      this.show = true;
},
```

新增效果预览



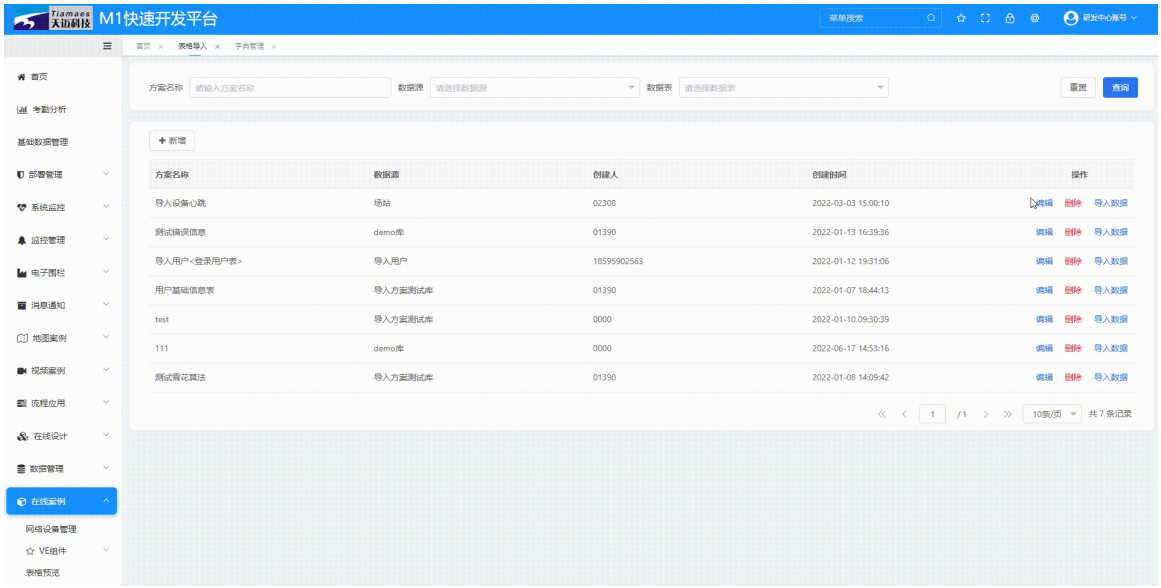
定制编辑行为，需要给方案编辑组件增加方案 ID 参数

```
<el-drawer title="方案定制" :visible.sync="show" destroy-on-close direction="rtl" size="50%">
        <scheme-manage :viewId="viewId"></scheme-manage>
</el-drawer>
```

编辑事件回调函数中接收数据并更新

```
edit({data, item}) {
        // 省略 viewId 声明
        this.viewId = data.row.id;
        this.show = true;
}
```

效果预览



二 定制数据导入

> 在演示案例中, 有定制导入数据行为的需求

1. 开启导入定制

```
custom: {
        import: true, // 开启导入定制
},
```

开启后, 默认的导入数据按钮不再展示, 可以通过属性配置自定义新的导入按钮

2. 自定义按钮

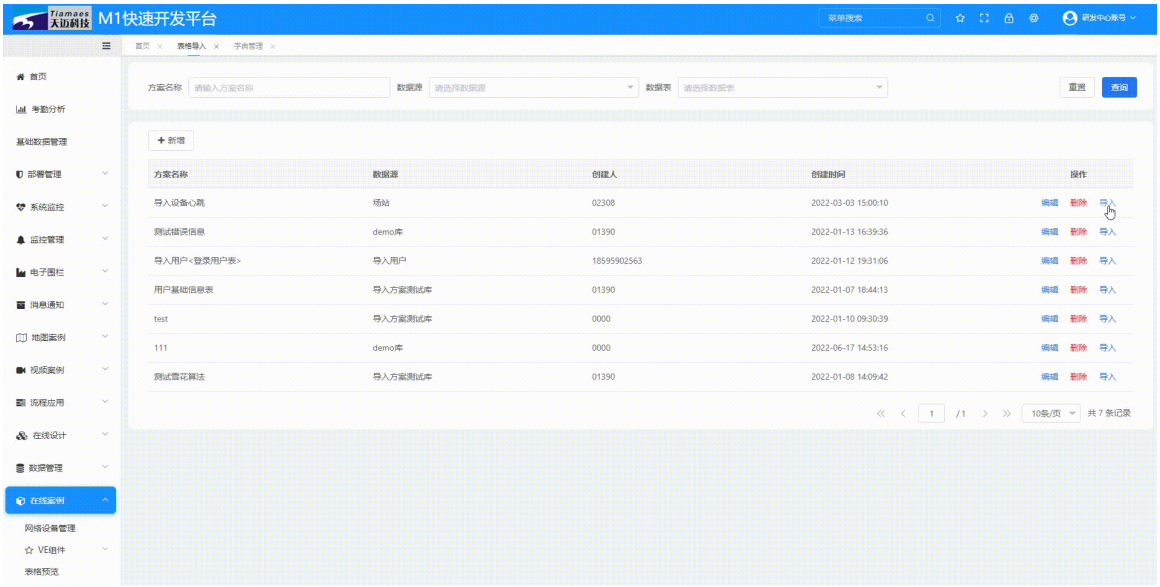
声明 `appendOperations` 按钮配置:

```
appendOperations: [
        {
          name: "导入",
          click: ({ data, item, setButtonStatus }) => {
            const { row } = data;
            this.$router.push({
              path: "/experiment/excelView",
              query: {
                id: row.id
              }
            });
          }
        }
 ]
```

以上配置的涵义是: 添加一个名称是导入的按钮, 点击时跳转到新页面 `/experiment/excelView` 并携带参数 id 为方案 id.

> 路由跳转新页面需要先创建对应的菜单, 步骤与方案列表相同. 不再赘述

3. 预览



三 完整代码

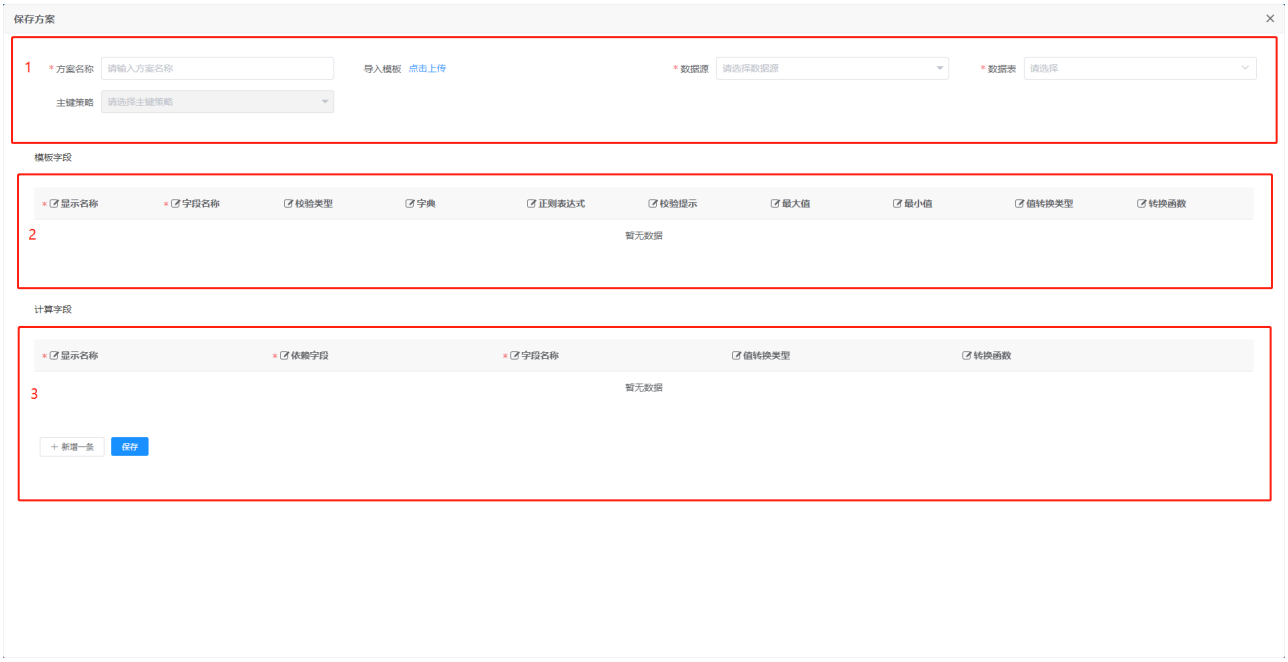
```
<template>
  <div>
    <scheme-list :custom="custom"
:appendOperations="appendOperations" @toolbar-add-click="add"
@operation-edit-click="edit"></scheme-list>
    <el-drawer title="方案定制" :visible.sync="show" destroy-on-close
direction="rtl" size="50%">
      <scheme-manage :viewId="viewId"></scheme-manage>
    </el-drawer>
  </div>
</template>

<script>
import { SchemeList, SchemeManage } from "@tiamaes/cbb-excel";

export default {
  components: {
    SchemeList,
    SchemeManage,
  },

  data() {
    return {
      custom: {
        edit: true,
        import: true,
```

```
      },

      show: false,
      viewId: '',

      appendOperations: [
        {
          name: "导入",
          click: ({ data, item, setButtonStatus }) => {
            const { row } = data;
            this.$router.push({
              path: "/experiment/excelView",
              query: {
                id: row.id
              }
            });
          }
        }
      ]
    };
  },

  methods: {
    add(...args) {
      console.log("add: ", args);
      this.show = true;
    },
    edit({data, item}) {
      this.viewId = data.row.id;
      this.show = true;
    }
  }
};
</script>

<style lang="scss" scoped>
</style>
```

新增方案

方案列表点击新增, 打开方案保存界面

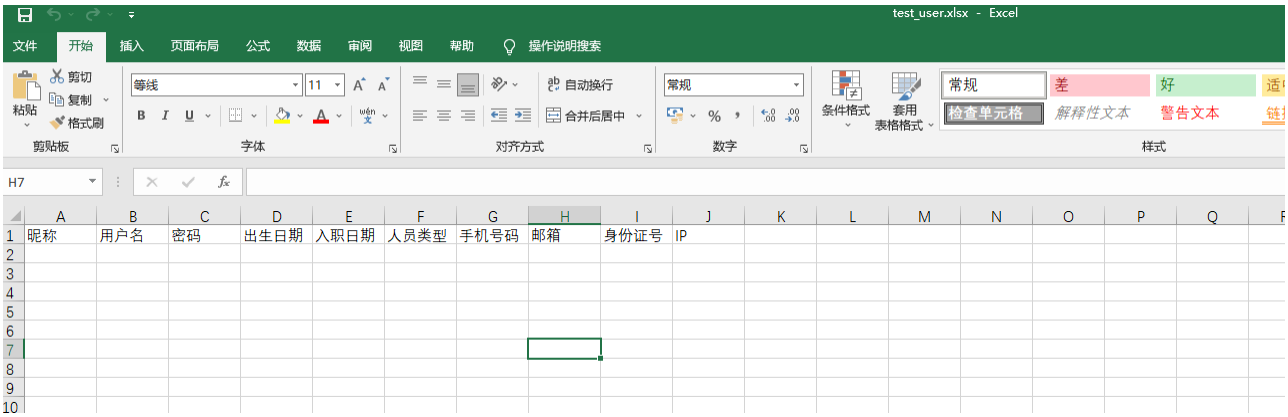
界面介绍



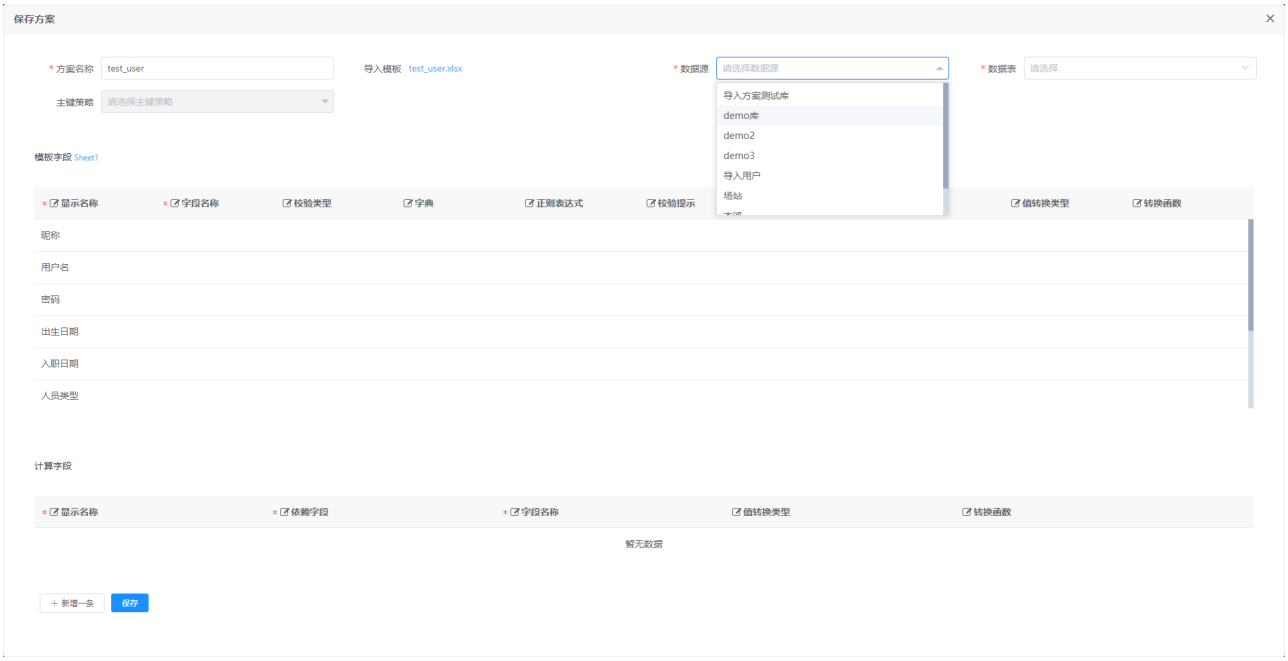
1. 方案配置表单, 包括方案名称、模板文件、数据源、数据表、主键策略
2. 模板字段区域, 展示上传的 Excel 模板文件中的所有列的信息, 可以分别配置
3. 计算字段区域, 虚拟字段, 用于根据某个模板字段的值, 通过某种规则转换成新的值插入到数据库

填写方案基本信息

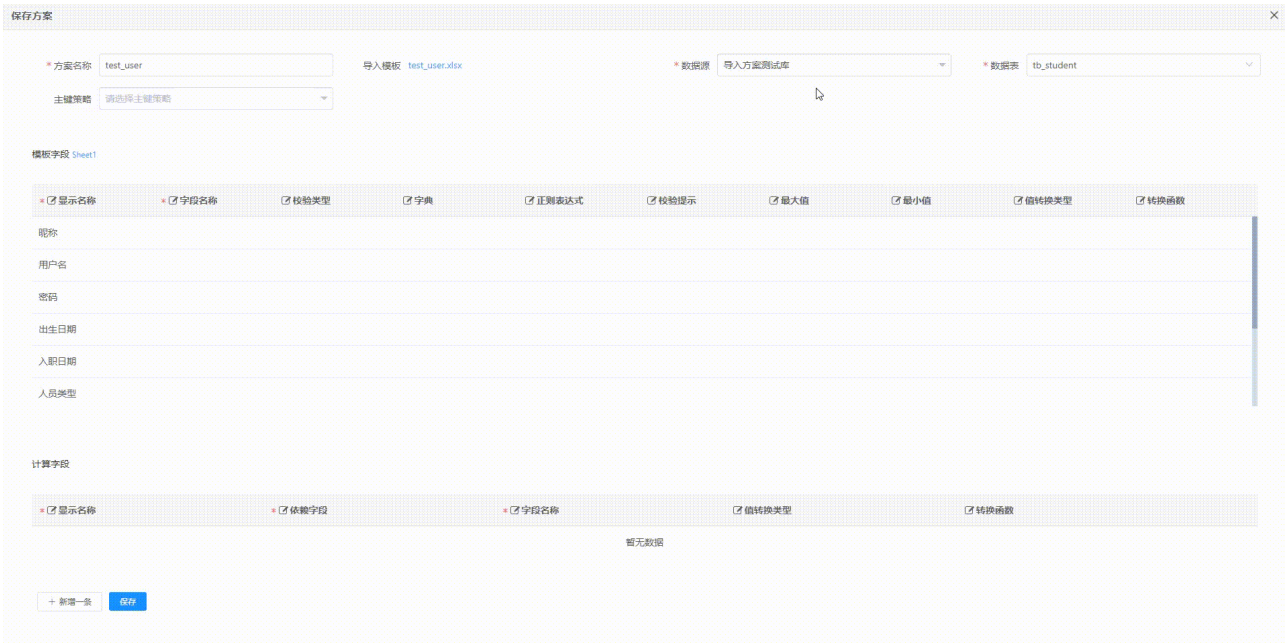
以用户表为例, 创建一个模板 Excel 文件, 填写一些字段



点击表单区域 > 导入模板 > 点击上传按钮, 选择准备好的 .xlsx 文件, 会读取响应的列名及 sheet 名, 展示在模板字段区域.



选择数据源, 会联动更新数据表的选项, 选择数据表, 会请求对应表的所有字段. 如果所选数据表包含主键, 主键策略可选, 否则主键策略禁用.



主键策略支持三种: Oracle 序列、MySql 自增、雪花算法.

本例中我们选择用户表, MySql 自增

配置模板字段

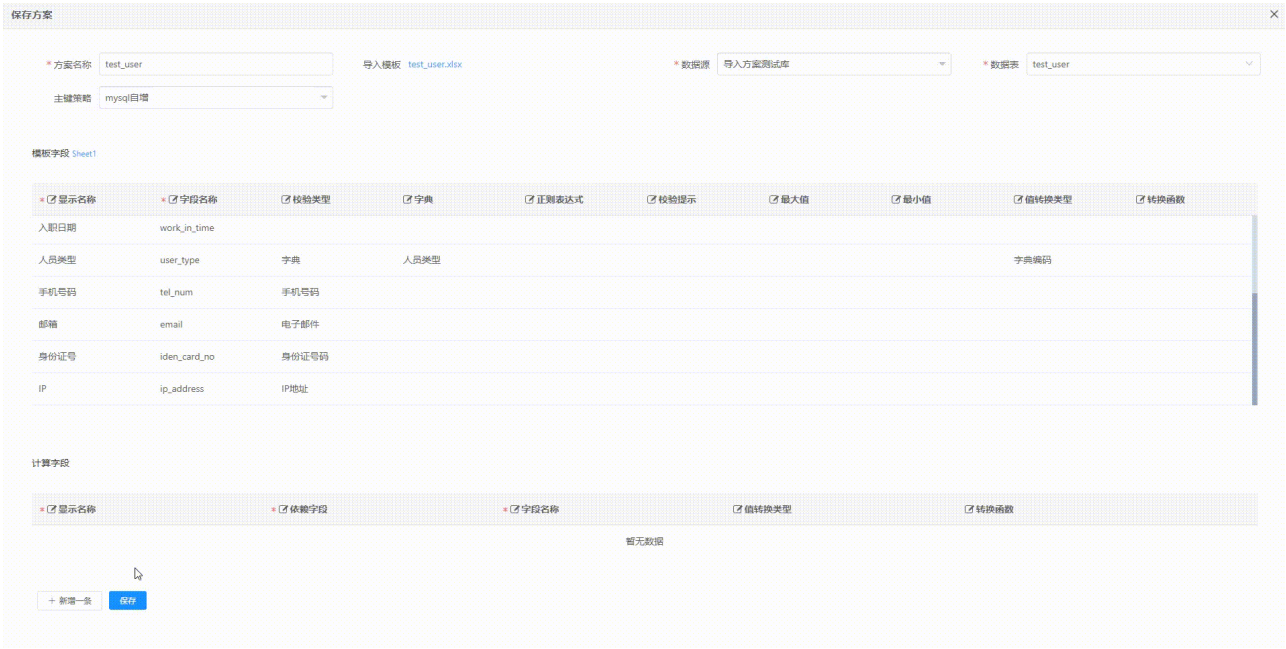
我们先从简单配置入手, 校验类型选择一些内置规则

* 显示名称	* 字段名称	校验类型	字典	正则表达式	校验提示	最大值	最小值	值转换类型
入职日期	work_in_time							
人员类型	user_type	字典	人员类型					字典编码
手机号码	tel_num	手机号码						
邮箱	email	电子邮件						
身份证号	iden_card_no	身份证号码						
IP	ip_address	IP地址						

配置项释义:

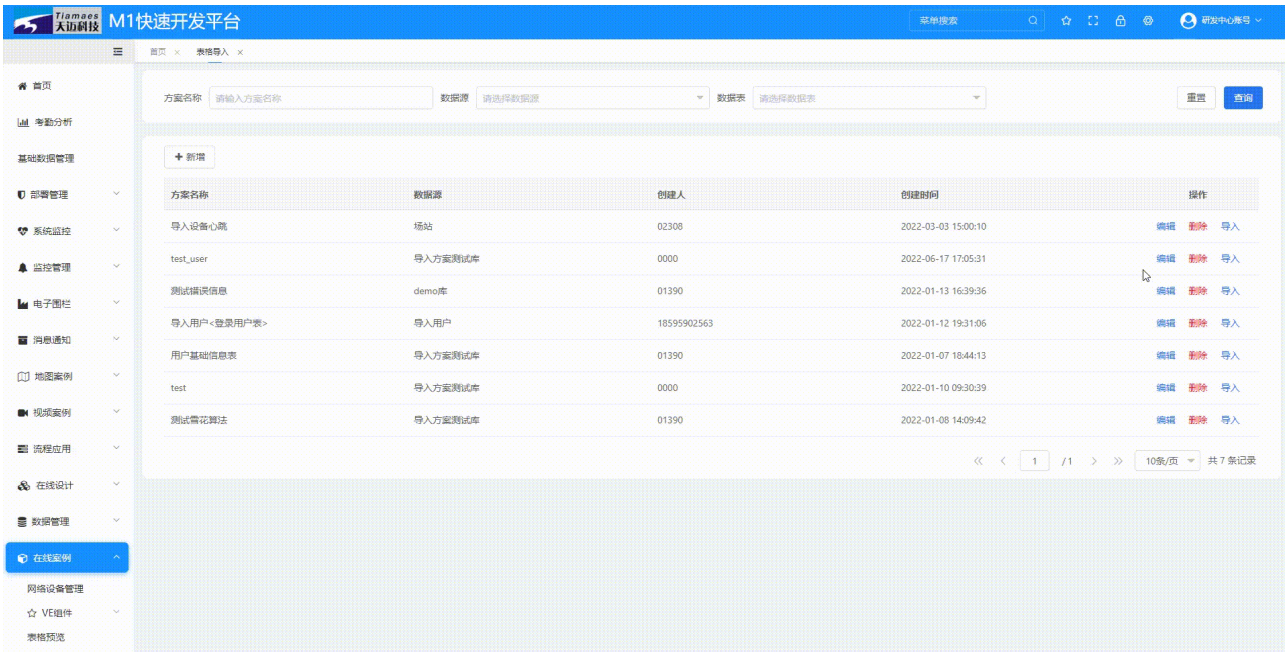
- 显示名称: 导入数据时展示在表格列的名称
- 字段名称: 数据库表的对应字段
- 校验类型: 导入数据时, 列对应数据的校验规则, 分为三类
    - 字典
    - 自定义正则表达式
    - 内置的正则表达式
- 字典: [校验类型] 选择字典时可用, 系统中选择已有字段, 导入数据时, 输入的值要在字典范围内
- 正则表达式: 如果内置的校验规则不满足需求, 可自定义正则表达式
- 校验提示: 校验类型为正则表达式的提示信息
- 最大值: 数值类型最大值
- 最小值: 数值类型最小值
- 值转换类型: 包含函数和字典编码
    - 函数: 导入数据时, 将函数的返回值写入数据库对应字段
    - 字典编码: 只有[校验类型]为字典时, 才可用. 将用户输入的名称转换为编码写入数据库
- 转换函数: [值转换类型]选择函数时可用.

保存



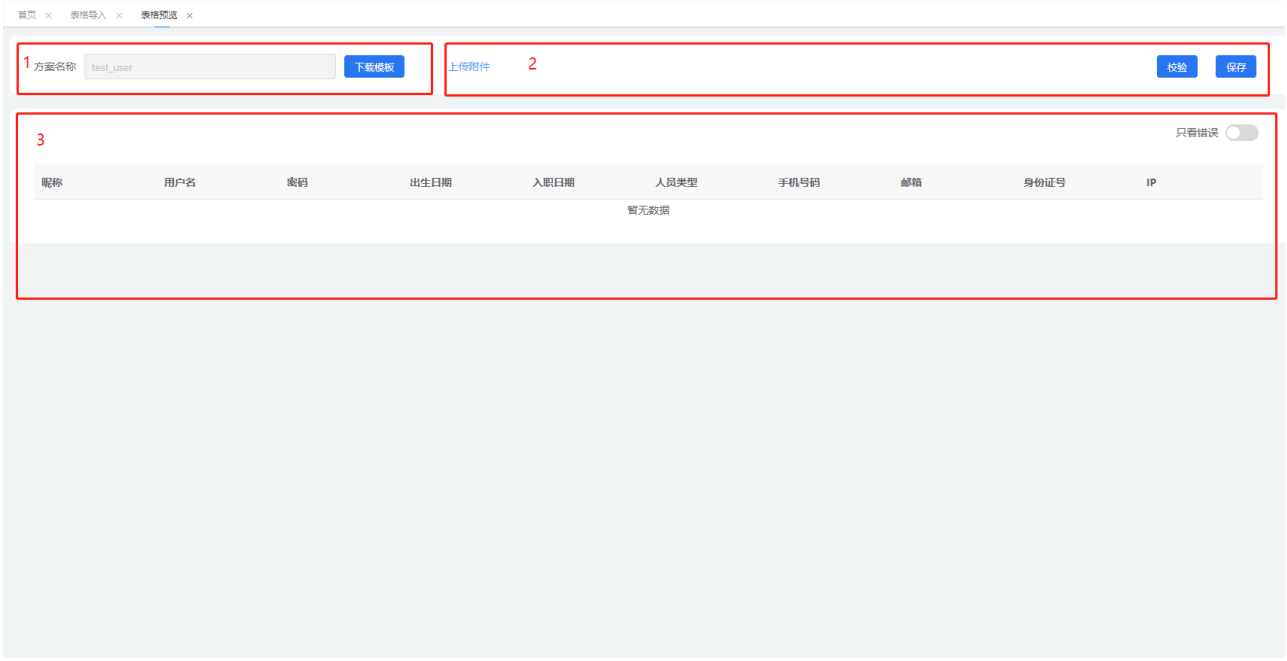
编辑方案

点击编辑时会打开方案保存界面并回显已有的方案信息. 所有操作都和新增方案一致, 不再赘述.



导入数据

点击导入按钮, 打开上一步创建好的用户导入方案



界面介绍

1. 方案信息: 包含方案名称(不可编辑)和模板文件

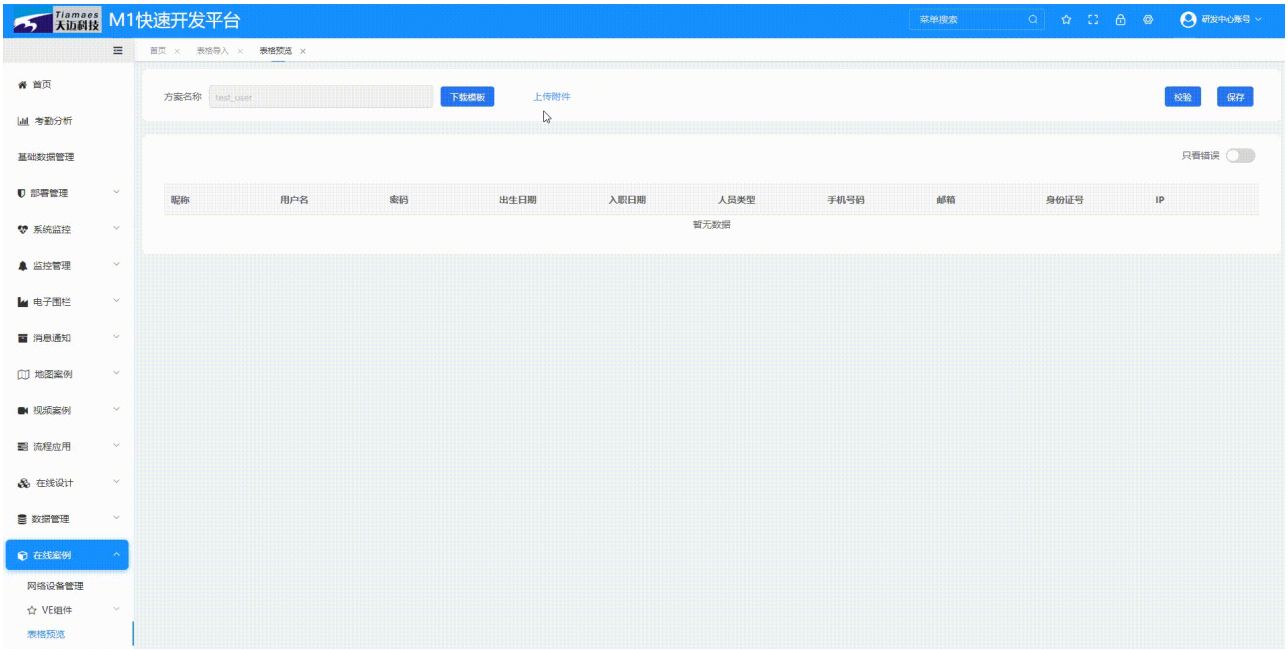
> 模板文件即新增方案时上传的 *.xlsx* 文件. 点击下载模板文件按钮, 会直接下载保存在服务器上的文件.
>
> 但是不使用下载模板, 本地重新创建一个 *.xlsx* 文件, 列名按照数据区域展示出来的列名称录入, 也是可用的.

2. 操作区域:
    - 上传附件: 上传填写完数据的 .xlsx 文件
    - 校验: 按照导入方案配置好的校验规则对导入的数据进行全量校验
    - 保存: 将处理好的数据通过接口插入数据库
3. 数据区域: 展示用户上传的 .xlsx 文件中的数据

下载模板并填写数据

	A	B	C	D	E	F	G	H	I	J	K	L
1	昵称	用户名	密码	出生日期	入职日期	人员类型	手机号码	邮箱	身份证号	IP		
2	啊啊啊	abc	123123	d	d	1	123123	aaa	123123	111		
3	等等	bbb	123123			2						
4												
5												

导入 .xlsx 文件后, 会展示出用户录入的数据, 并按照字段类型, 展示编辑组件



在上例中, 出生日期和入职日期的数据库表字段类型是日期时间, 因此表格编辑单元格以日期时间选择器展示; 人员类型选择了字典, 以下拉框展示, 其选项为对应的字典项. 其他默认为文本输入框.

校验

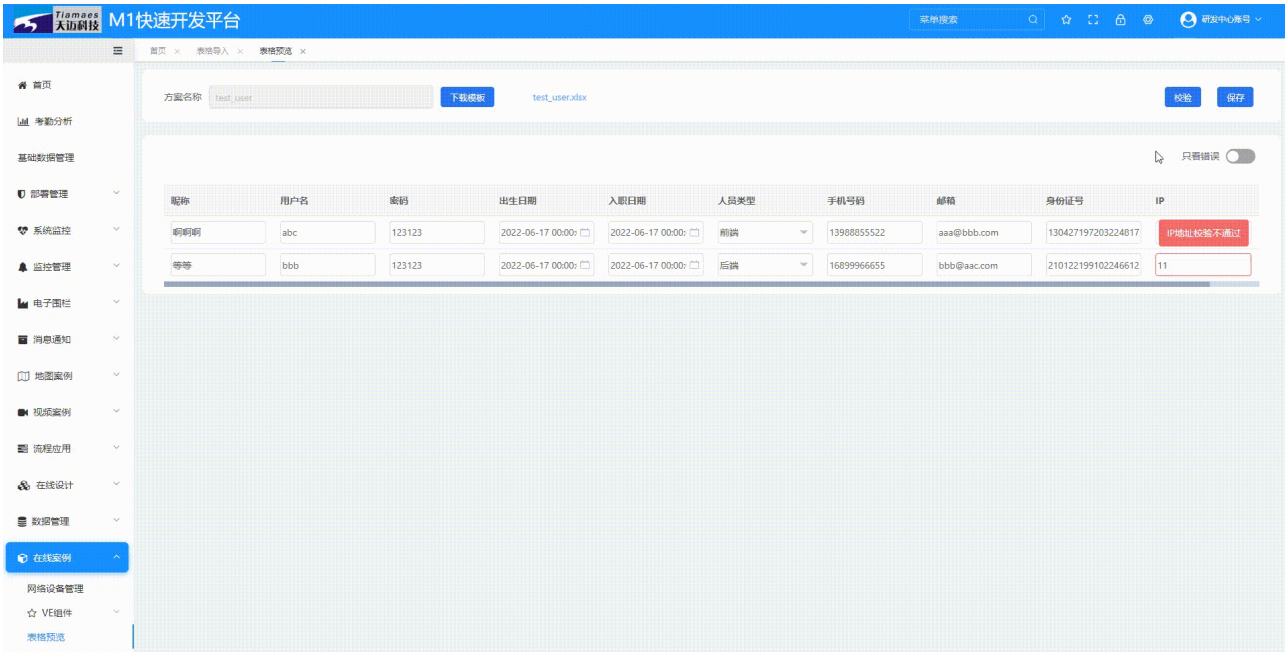
在数据保存之前, 需要先执行校验操作.

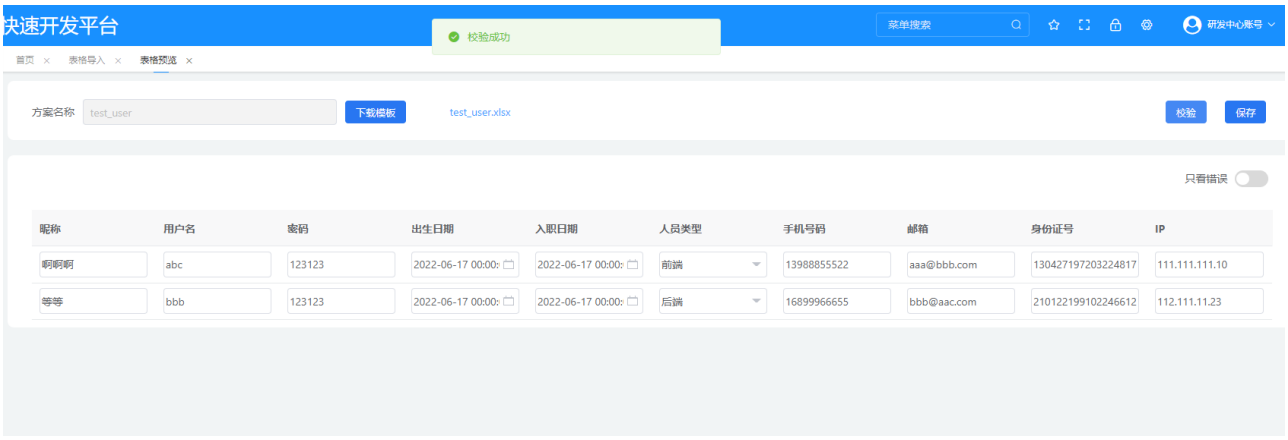
点击校验, 会标记不符合规则的单元格

本案例配置的规则如下:

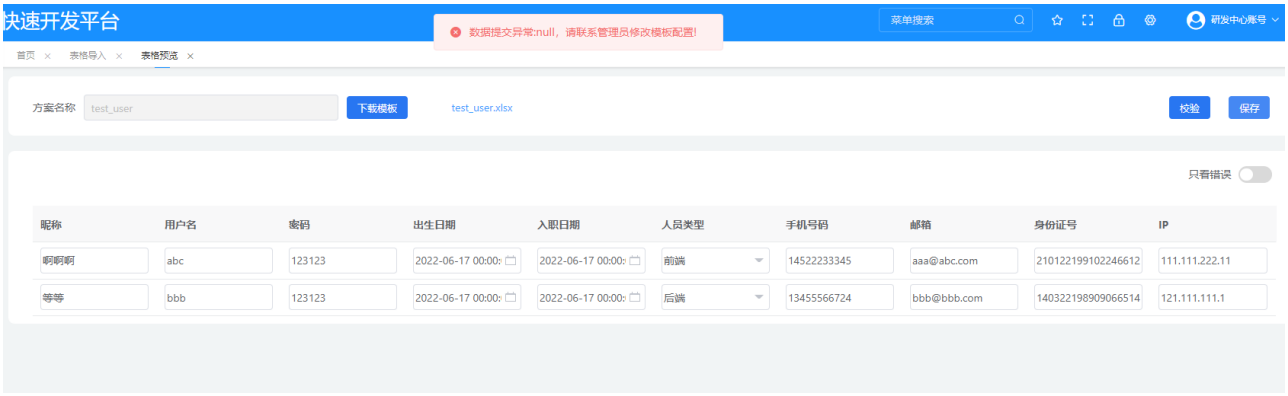
人员类型	user_type	字典	人员类型
手机号码	tel_num	手机号码	
邮箱	email	电子邮件	
身份证号	iden_card_no	身份证号码	
IP	ip_address	IP地址	

如果数据过多, 可以点击只看错误, 只展示包含错误的行



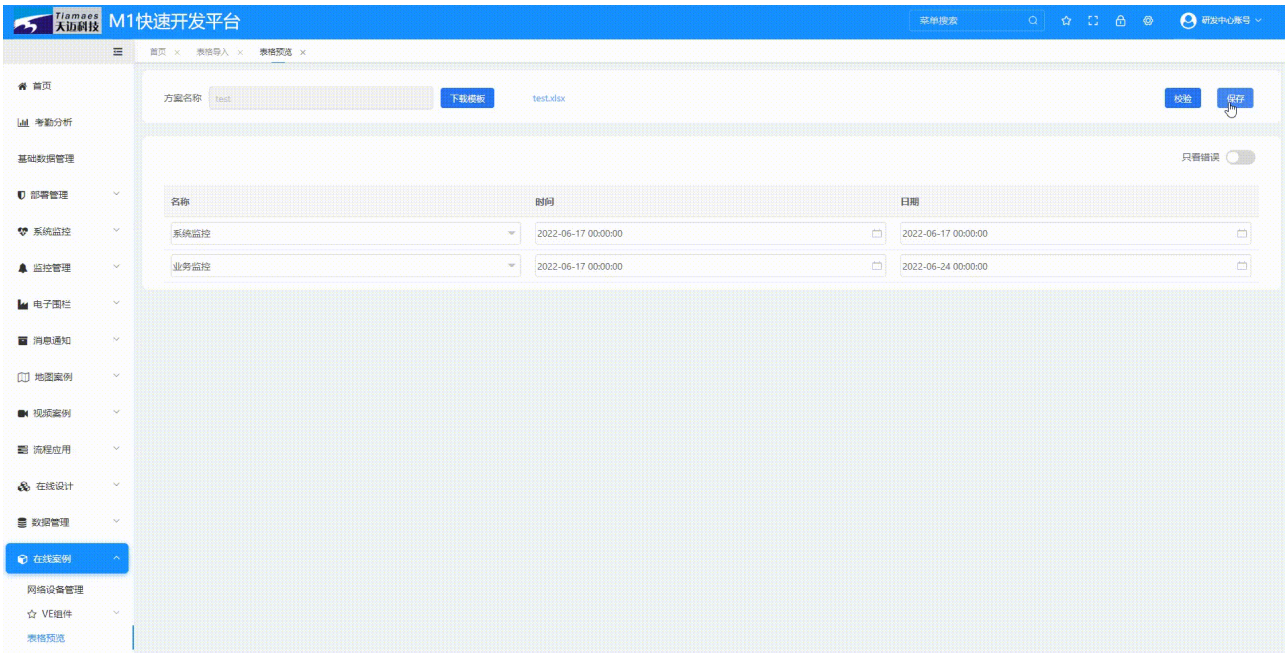


校验成功后, 可以点击保存



意外出现错误, 且没有明显提示, 需要后台开发人员协助或重新设计.

下面是一个成功的示例:



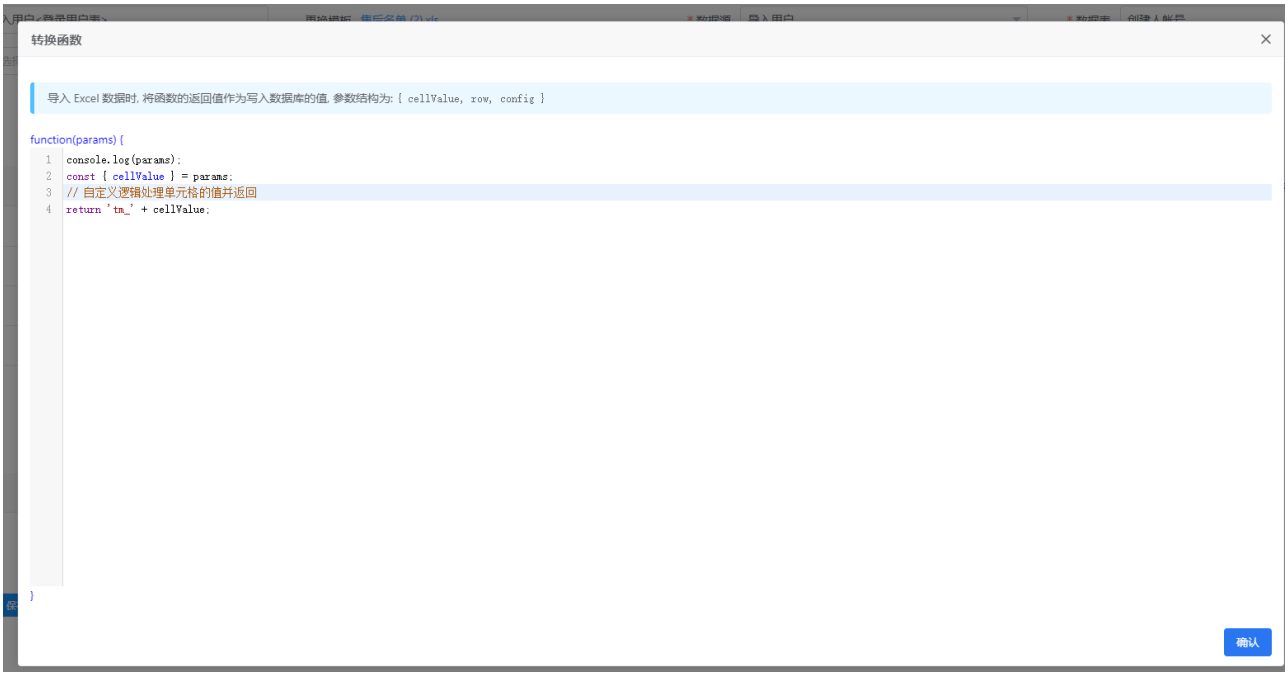
扩展

这一节介绍一些使用稍微复杂的扩展功能

转换函数

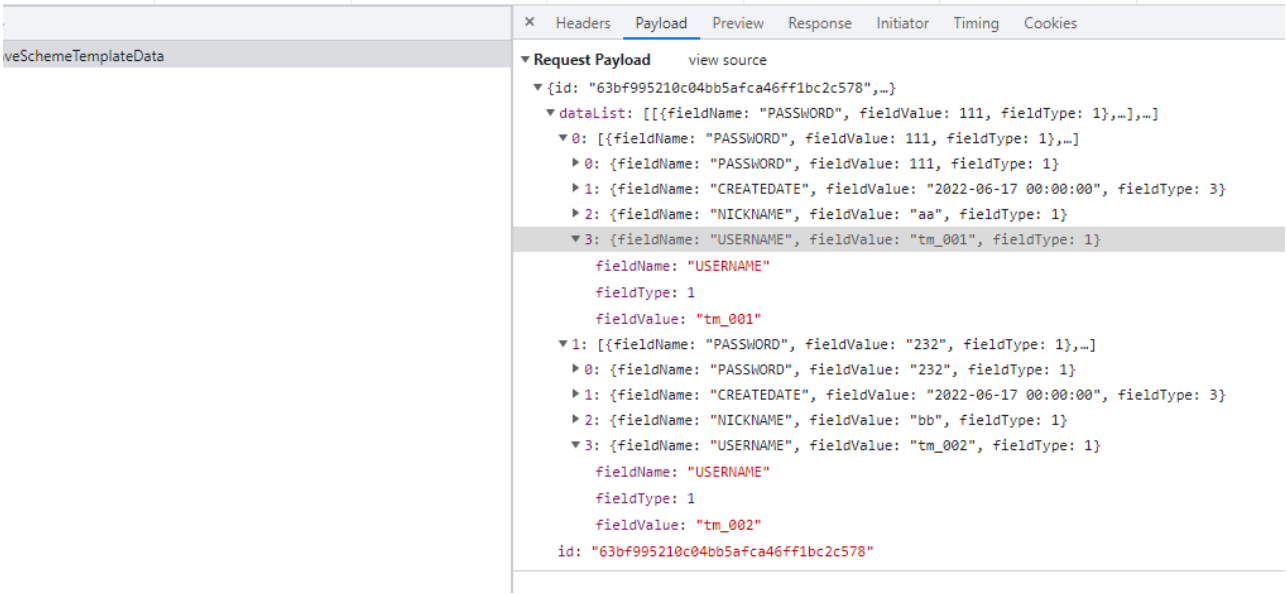
选择一个可以导入成功的方案进行编辑, 在模板字段选择一列, 值转换类型选择函数:

点击输入代码按钮, 打开函数编辑窗口, 根据业务编写代码. 图中只做演示



方案修改后保存, 重新点击导入, 上传 Excel 数据:

通过网络工具观察, 插入数据库的值已经转换为我们期望的值:

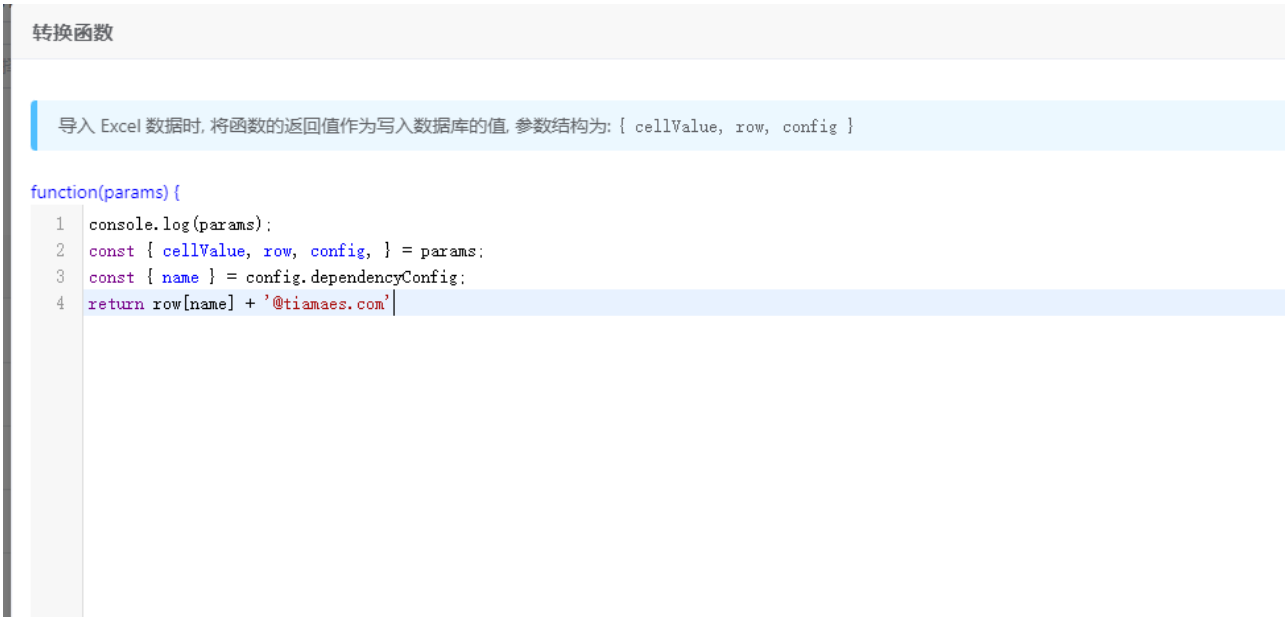


计算字段

在计算字段区域点击新增一条, 可以插入一项虚拟列, 不在数据导入列中展示, 但该列实际计算后的值会插入到数据库.

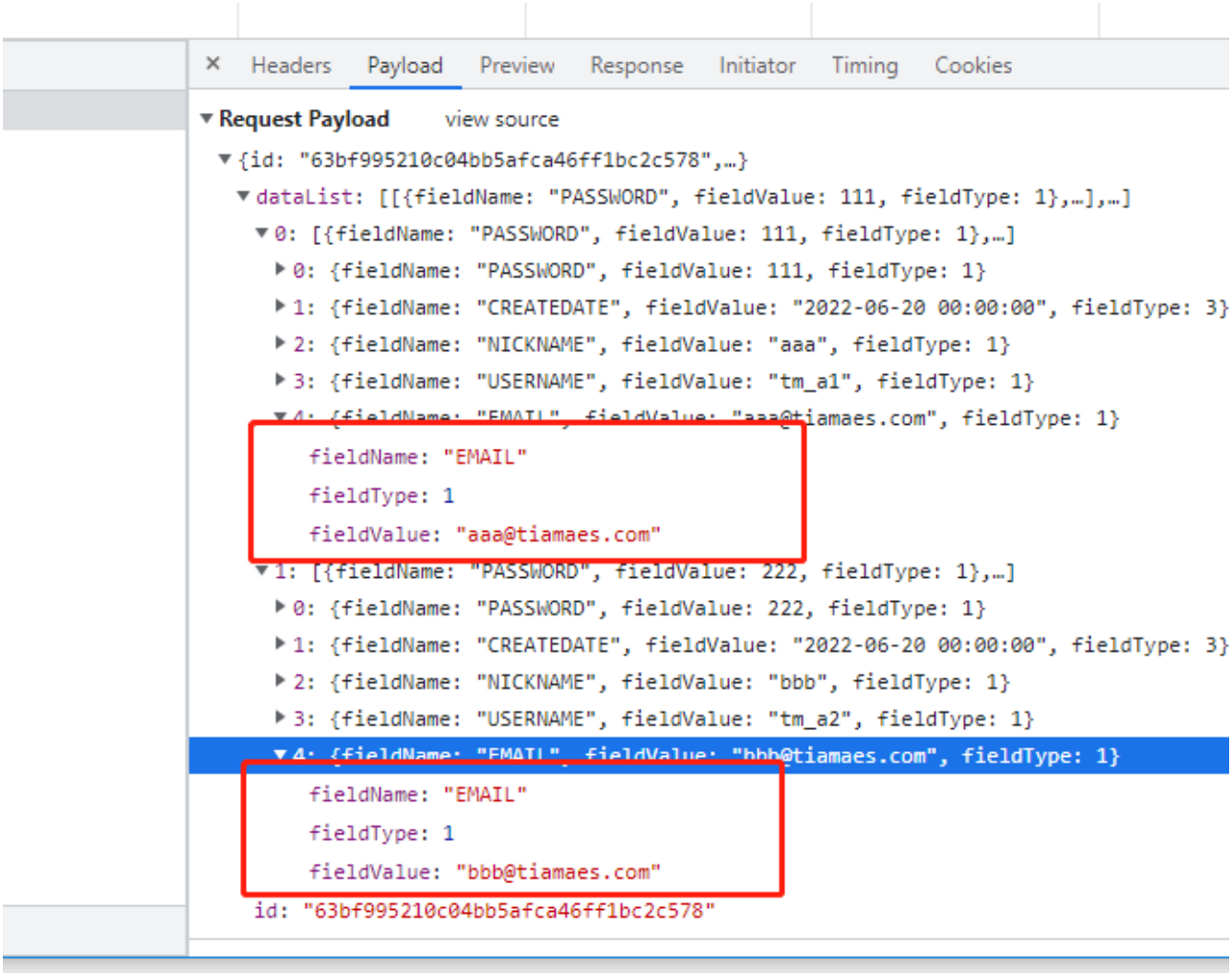
例如: 新增一个计算字段, 依赖字段选择姓名, 字段名称 `EMAIL`

校验类型选择函数, 并点击转换函数, 输入代码:



> 解释一下以上代码:
>
> 函数执行时从参数中解构出三个变量
>
> - *cellValue:* 单元格的值. 计算列在表格中并不存在, 因此 ***cellValue*** 也不存在
> - *row:* 行数据
> - *config:* 行对应配置
>
> 从依赖字段配置中取出依赖列名称 `name`, 则依赖字段值即为 `row[name]`
>
> 根据依赖字段的值进行计算并返回.

在表格导入界面, 校验数据并保存, 观察请求参数:


Headers
Payload
Preview
Response
Initiator
Timing
Cookies
Request Payload
view source
{id: "63bf995210c04bb5afca46ff1bc2c578",…}
dataList: [[{fieldName: "PASSWORD", fieldValue: 111, fieldType: 1},…],…]
0: [{fieldName: "PASSWORD", fieldValue: 111, fieldType: 1},…]
0: {fieldName: "PASSWORD", fieldValue: 111, fieldType: 1}
1: {fieldName: "CREATEDATE", fieldValue: "2022-06-20 00:00:00", fieldType: 3}
2: {fieldName: "NICKNAME", fieldValue: "aaa", fieldType: 1}
3: {fieldName: "USERNAME", fieldValue: "tm_a1", fieldType: 1}
fieldName: "EMAIL"
fieldType: 1
fieldValue: "aaa@tiamaes.com"
1: [{fieldName: "PASSWORD", fieldValue: 222, fieldType: 1},…]
0: {fieldName: "PASSWORD", fieldValue: 222, fieldType: 1}
1: {fieldName: "CREATEDATE", fieldValue: "2022-06-20 00:00:00", fieldType: 3}
2: {fieldName: "NICKNAME", fieldValue: "bbb", fieldType: 1}
3: {fieldName: "USERNAME", fieldValue: "tm_a2", fieldType: 1}
fieldName: "EMAIL"
fieldType: 1
fieldValue: "bbb@tiamaes.com"
id: "63bf995210c04bb5afca46ff1bc2c578"


1快速开发平台
数据导入成功
菜单搜索
方案名称
下载模板
test.xlsx
校验
只看错误
密码
创建日期
姓名
工号
111
2022-06-17 00:00:00
aabb
111
222
2022-06-17 00:00:00
ccdd
222

4. 更新日志