小组成员

姓名	学号	负责部分
徐弋童	201831063402	原型调研、页面编写
苟进	201831063119	PPT、文档、全部测试
郝杰	201831063120	建表、接口编写

一、项目简介

1.1 项目背景

某些场景中，我们可能需要一个高度可控的问卷系统，进可数据收集可视化分析，退可胡编乱造凑个数就行。

1.2 产品定位

同终端不同服务端的 H5 问卷系统。

> 同终端不同服务端是指：
>
> - 用户可使用部署在同一处的 H5 终端去访问不同的服务端系统
> - H5 终端是无状态的
> - 服务端调用是透明的

1.3 原型灵感

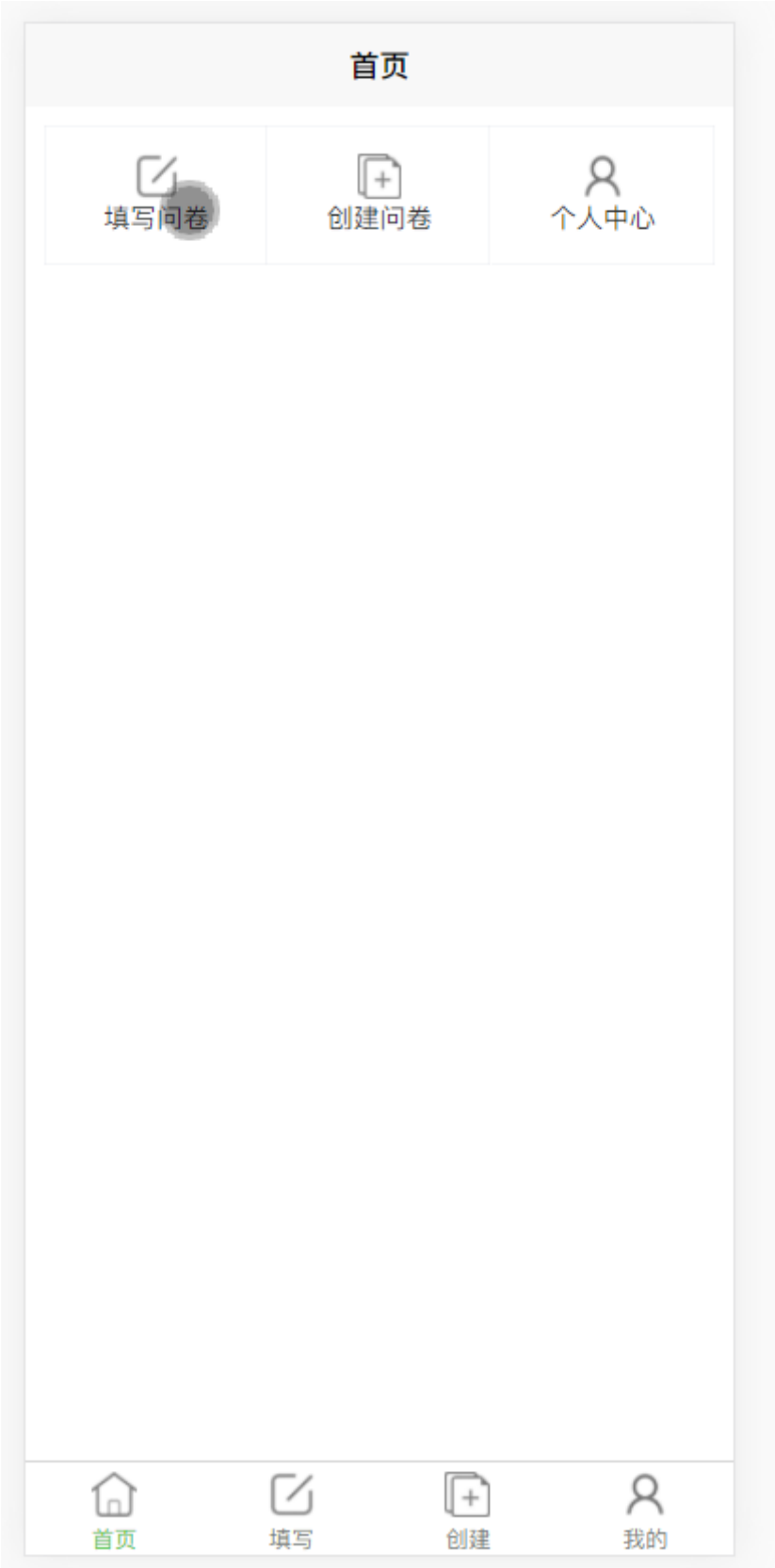
腾讯收集表

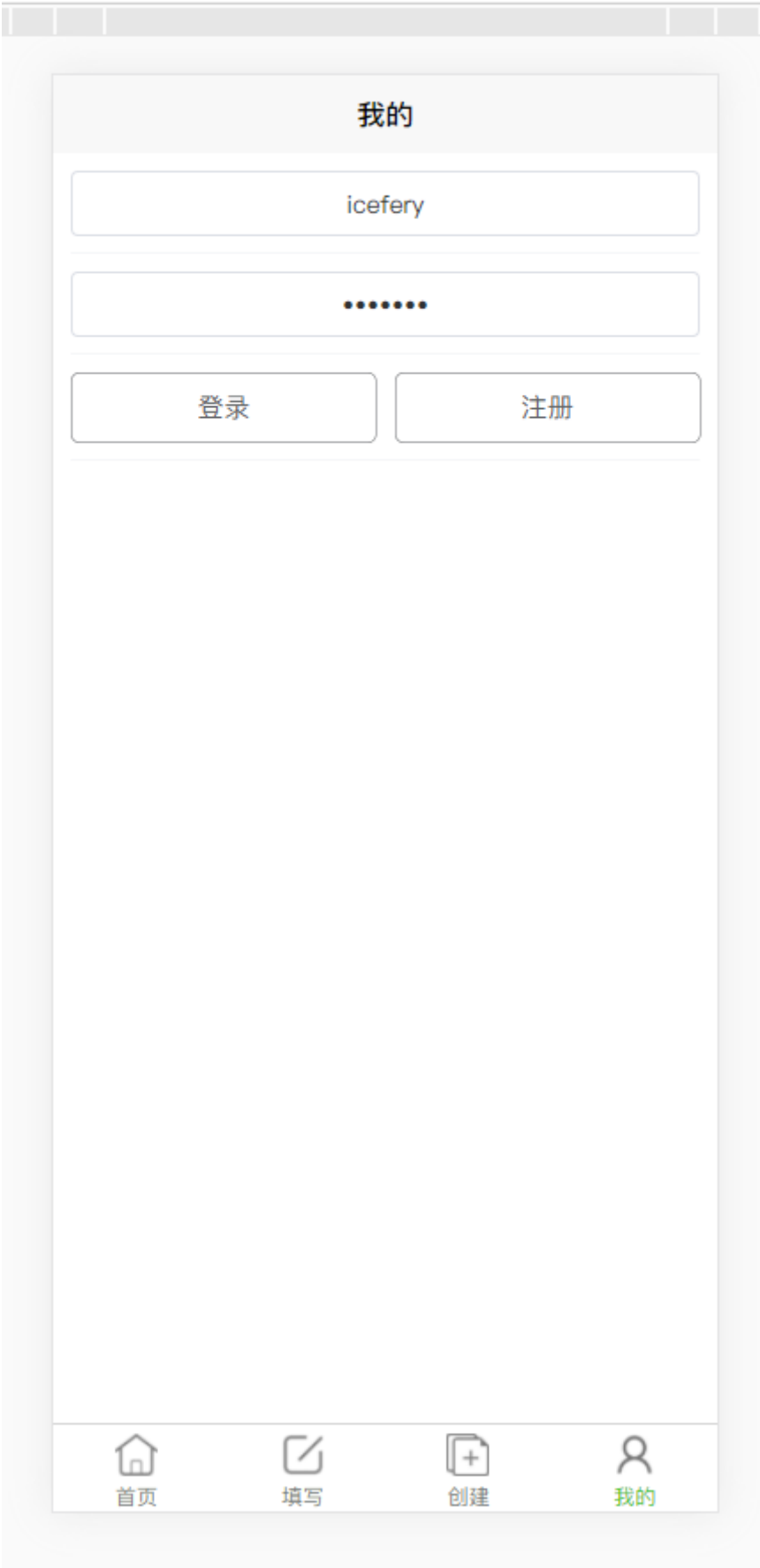
> 腾讯收集表在国内移动用户群体中占据了相当大的比重，我们吸收了腾讯收集表的题型设置并进行了一些扩展。

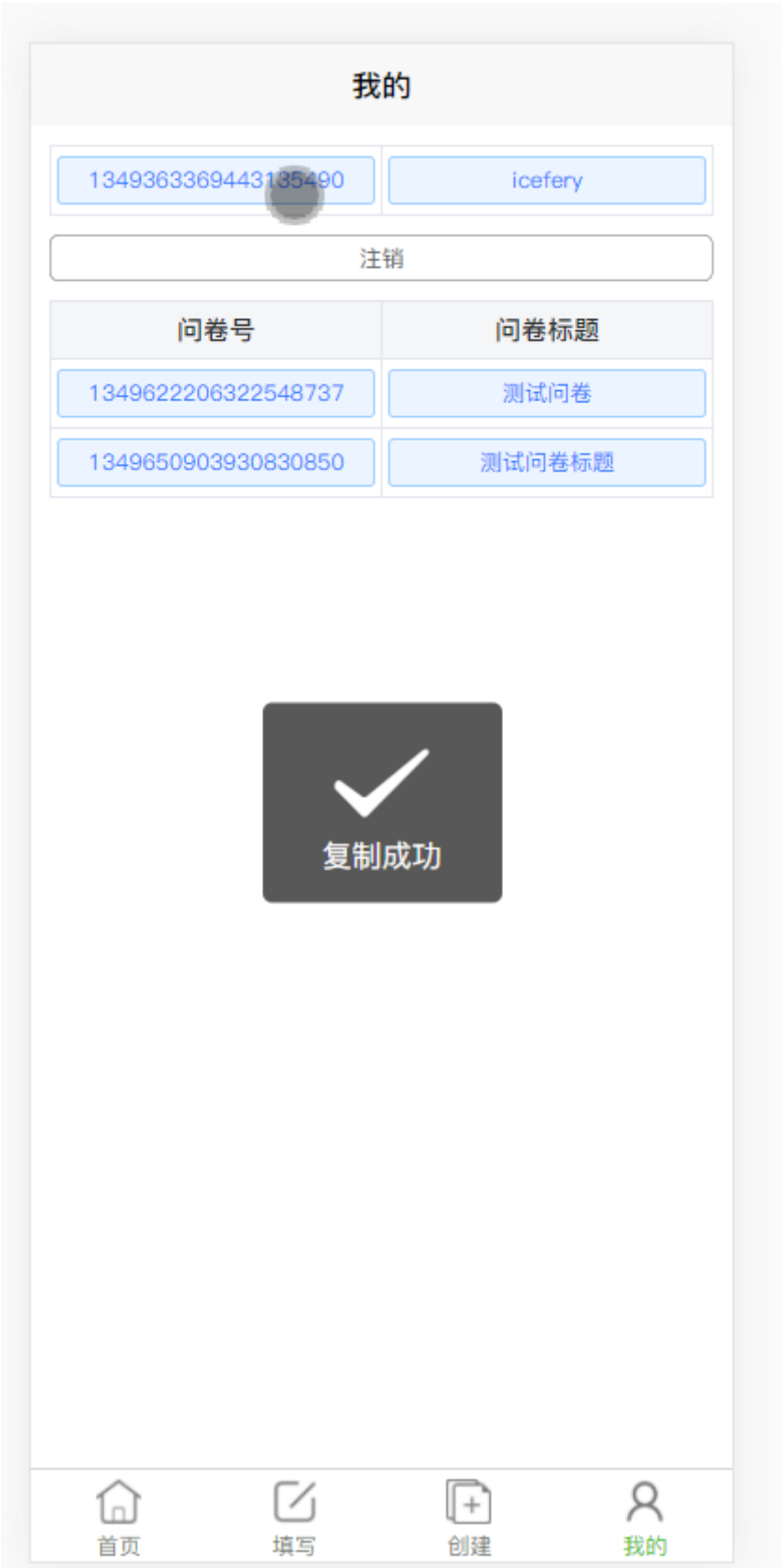
`Object.assign(target, ...sources)` 函数

> 借助 `Object.assign(target, ...sources)` 函数可以简单便捷的进行对象的浅拷贝和合并操作，同时它也是我们问卷系统的核心设计思想之一。

1.4 截图预览


首页
填写问卷
创建问卷
个人中心
首页
填写
创建
我的


我的
icefery
登录
注册
首页
填写
创建
我的


我的
1349363369443135490
icefery
注销
问卷号
问卷标题
1349622206322548737
测试问卷
1349650903930830850
测试问卷标题
复制成功
首页
填写
创建
我的


创建问卷
基本信息
*标题
测试问卷标题
*描述
测试问卷描述
1
类型
问答
单选
多选
位置
时间
题目
测试位置
内容
四川省 成都市 新都区
必选
位置
省
市
区
2
取消
确认
广西壮族自治区
海南省
重庆市
四川省
贵州省
云南省
西藏自治区
成都市
自贡市
攀枝花市
泸州市
成华区
龙泉驿区
青白江区
新都区
温江区
双流区
郫都区

二、需求分析

2.1 可行性分析

- 存在一些允许我们自定义问卷调查并进行统计分析的场景
- `Object.assign(target, ...sources)`函数
- MyBatisPlus 多租户模式支持

2.2 功能分析

- [x] 登录注册
- [x] 创建问卷
- [x] 填写问卷
- [ ] 服务端切换
- [ ] 填写时扩展
- [ ] 数据可视化
- [ ] 多格式导入导出
- [ ] 统计分析
- [ ] 自动化批量胡编乱造

三、总体设计

3.1 概要

我们问卷系统的核心需求之一就是填写时扩展。填写时扩展是指：用户创建一份问卷也即创建一条问卷记录，问卷中既包括题目类型信息还包括题目默认数据。填写一份问卷也即创建一条基于该问卷模板的问卷项记录并修改其默认数据，同时按需扩展新题目。

3.2 数据逻辑结构

3.2.1 数据类型

```ts
// 问卷
interface Questionnaire {
  // 问卷 ID（服务端生成）
  id: string
  // 租户 ID
  tenantId: string
  // 问卷标题
  title: string
  // 问卷描述
  description: string
  // 问卷项题目模板
  questions: Question[]
}

// 问卷项
interface QuestionnaireItem {
  // 问卷项 ID（服务端生成）
  id: string
  // 租户 ID
  tenantId: string
  // 问卷 ID
  questionnaireId: string
  // 问卷项题目
  questions: Question[]
}

// 题目
interface Question {
  // 题目类型（问答 | 单选 | 多选 | 位置 | 时间 | 图片）
  type: 'qa' | 'radio' | 'checkbox' | 'location' | 'datetime' | 'image'
  // 题目标题
  title: string
  // 题目回答
  value: string | RadioValue | CheckboxValue | LocationValue | DatetimeValue | ImageValue
  // 题目是否必填
  required: boolean
  // 题目配置
  config: QuestionConfig
}
```

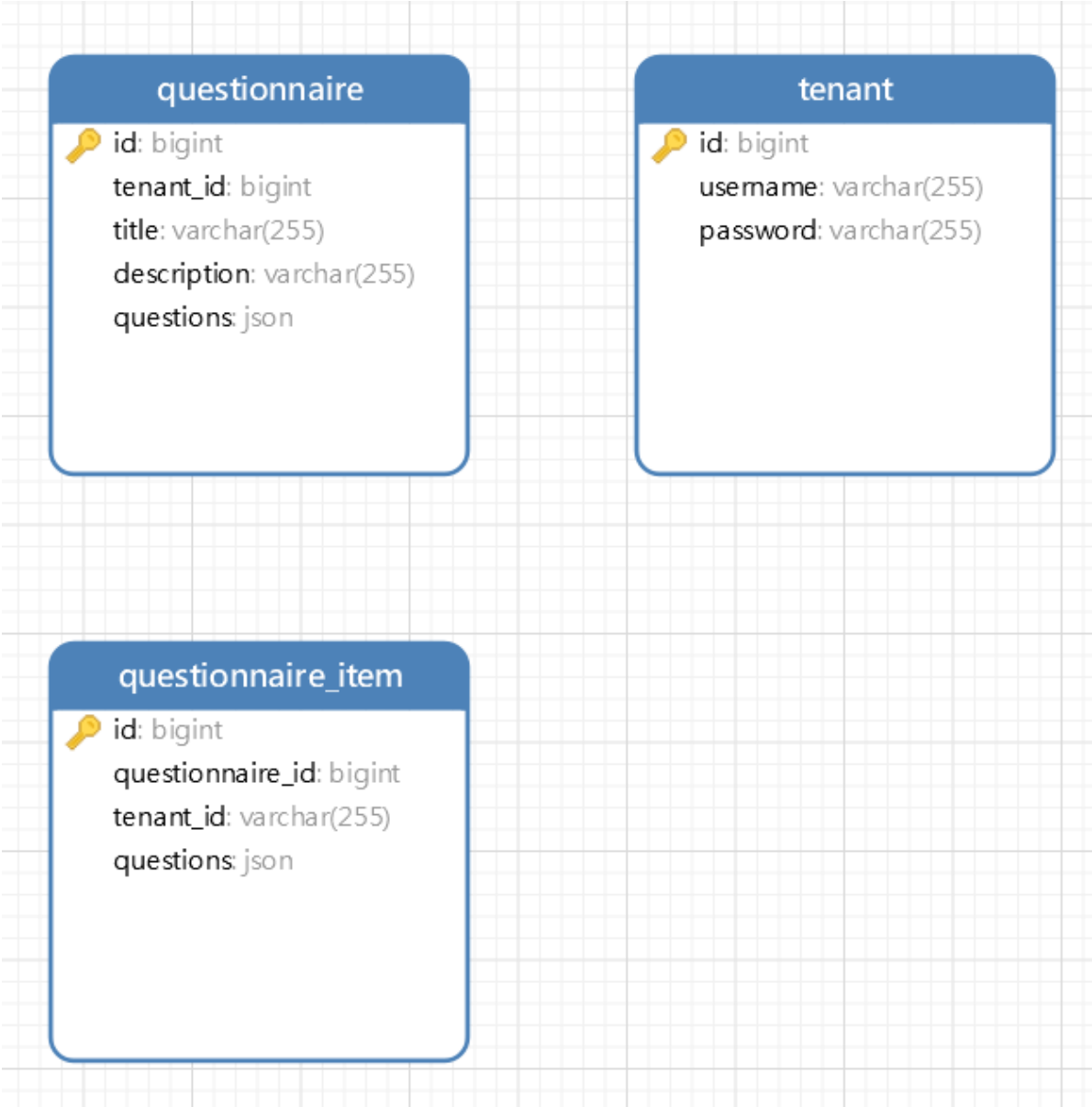
######

3.2.3 测试数据

```ts
// 测试问卷
const testQuestionnaire: Questionnaire = {
  id: '1349622206322548737',
  tenantId: '1349363369443135490',
  title: '测试问卷',
  description: '',
  questions: [
    {
      type: 'datetime',
      title: '当前日期',
      value: { year: '2000', month: '01', day: '01' },
      required: true,
      config: {
        datetimeParams: [
          { name: 'year', checked: true },
          { name: 'month', checked: true },
          { name: 'day', checked: true },
          { name: 'hour', checked: false },
          { name: 'minute', checked: false },
          { name: 'second', checked: false }
        ]
      }
    }
  ]
}

// 测试问卷项
const testQuestionnaireItem: QuestionnaireItem = {
  id: '1349685921243742209',
  tenantId: '1349363369443135490',
  questionnaireId: '1349622206322548737',
  questions: [
    {
      type: 'datetime',
      title: '当前日期',
      value: { year: '2021', month: '01', day: '14' },
      required: true,
      config: {
        datetimeParams: [
          { name: 'year', checked: true },
          { name: 'month', checked: true },
          { name: 'day', checked: true },
          { name: 'hour', checked: false },
          { name: 'minute', checked: false },
          { name: 'second', checked: false }
        ]
      }
    }
  ]
}
```

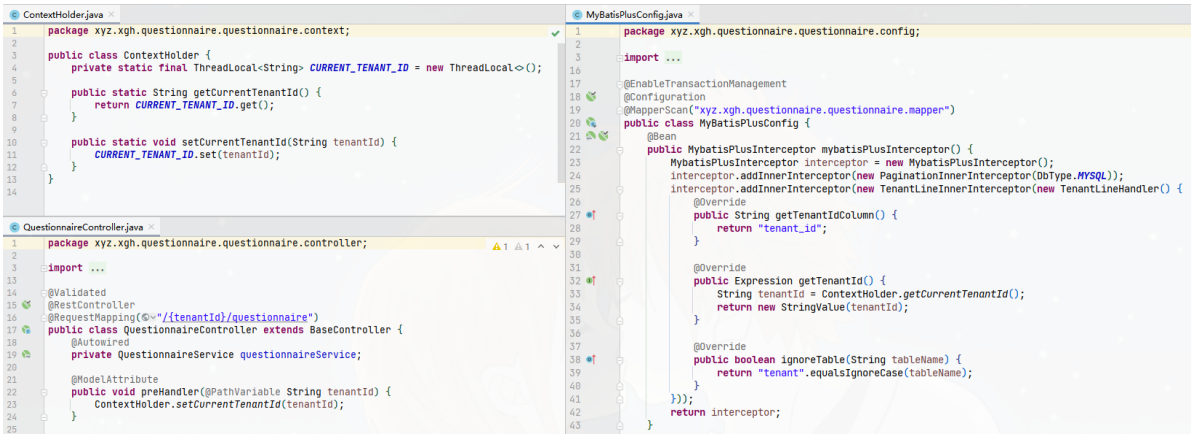
3.3 数据存储结构



四、详细设计与实现

4.1 共享数据表的租户过滤

进入 Controller 时将租户信息注入租户上下文中，进入 DAO 层后从租户上下文取出租户信息并注入到 SQL 中：



ContextHolder.java

```java
package xyz.xgh.questionnaire.questionnaire.context;

public class ContextHolder {
    private static final ThreadLocal<String> CURRENT_TENANT_ID = new ThreadLocal<>();

    public static String getCurrentTenantId() {
        return CURRENT_TENANT_ID.get();
    }

    public static void setCurrentTenantId(String tenantId) {
        CURRENT_TENANT_ID.set(tenantId);
    }
}
```

QuestionnaireController.java

```java
package xyz.xgh.questionnaire.questionnaire.controller;

import ...

@Validated
@RestController
@RequestMapping("/{tenantId}/questionnaire")
public class QuestionnaireController extends BaseController {
    @Autowired
    private QuestionnaireService questionnaireService;

    @ModelAttribute
    public void preHandler(@PathVariable String tenantId) {
        ContextHolder.setCurrentTenantId(tenantId);
    }
```

MyBatisPlusConfig.java

```java
package xyz.xgh.questionnaire.questionnaire.config;

import ...

@EnableTransactionManagement
@Configuration
@MapperScan("xyz.xgh.questionnaire.questionnaire.mapper")
public class MyBatisPlusConfig {
    @Bean
    public MybatisPlusInterceptor mybatisPlusInterceptor() {
        MybatisPlusInterceptor interceptor = new MybatisPlusInterceptor();
        interceptor.addInnerInterceptor(new PaginationInnerInterceptor(DbType.MYSQL));
        interceptor.addInnerInterceptor(new TenantLineInnerInterceptor(new TenantLineHandler() {
            @Override
            public String getTenantIdColumn() {
                return "tenant_id";
            }

            @Override
            public Expression getTenantId() {
                String tenantId = ContextHolder.getCurrentTenantId();
                return new StringValue(tenantId);
            }

            @Override
            public boolean ignoreTable(String tableName) {
                return "tenant".equalsIgnoreCase(tableName);
            }
        }));
        return interceptor;
    }
```

4.2 创建问卷

4.2.1 题目类型切换

```vue
<!-- 题型 -->
<u-form-item label="类型">
  <u-subsection
    style="width: 100%;"
    :animation="true"
    :list="types"
    :current="types.findIndex(item => item.value === question.type)"
    @change="i => onTypeChange(types[i].value, question)"
  />
</u-form-item>
```

```
  },
  // 题目类型
  types: [
    { name: '问答', value: 'qa' },
    { name: '单选', value: 'radio' },
    { name: '多选', value: 'checkbox' },
    { name: '位置', value: 'location' },
    { name: '时间', value: 'datetime' }
    // TODO { name: '图片', value: 'image' }
  ],
```

```
// 更改类型
onTypeChange(type, question) {
  question.type = type
  question.control = { swipeShow: false, pickerShow: false, uploadRef: null }
  switch (type) {
    // 问答
    case 'qa':
      question.value = ''
      break
    // 单选
    case 'radio':
      question.value = { options: ['选项 1', '选项 2'], checked: '选项 1' }
      question.control.cachedOptionNo = 2
      break
    // 多选
    case 'checkbox':
      question.value = [{ name: '选项 1', checked: false }, { name: '选项 2', checked: false }]
      question.control.cachedOptionNo = 2
      break
    // 位置
    case 'location':
      question.config = {
        locationParams: [
          { name: '省', value: 'province', checked: true },
          { name: '市', value: 'city', checked: true },
          { name: '区', value: 'area', checked: true }
        ]
      }
      question.value = { province: '四川省', city: '成都市', area: '新都区' }
      break
    // 时间
    case 'datetime':
      question.config = {
        datetimeParams: [
          { name: '年', value: 'year', checked: true },
          { name: '月', value: 'month', checked: true },
          { name: '日', value: 'day', checked: true },
          { name: '时', value: 'hour', checked: true },
          { name: '分', value: 'minute', checked: true },
          { name: '秒', value: 'second', checked: true }
        ]
      }
      const date = new Date()
      question.value = {
        year: date.getFullYear(),
        month: date.getMonth() + 1,
        day: date.getDate(),
        hour: date.getHours(),
        minute: date.getMinutes(),
        second: date.getSeconds()
      }
      break
    // 图片
    case 'image':
      question.value = []
      break
  }
},
```

4.2.1 添加单选、多选选项

```
// 添加单选
addRadioOption(question) {
  const no = question.control.cachedOptionNo + 1
  question.value.options.push(`选项 ${no}`)
  question.control.cachedOptionNo = no
},
// 移除单选
removeRadioOption(question, index) {
  const isDeletingChecked = question.value.options[index] === question.value.checked
  question.value.options.splice(index, 1)
  if (isDeletingChecked || question.value.options.length <= 1) {
    question.value.checked = question.value.options[0]
  }
},
// 添加多选
addCheckboxOption(question) {
  const no = question.control.cachedOptionNo + 1
  question.value.push({ name: `选项 ${no}`, checke: false })
  question.control.cachedOptionNo = no
},
// 移除多选
removeCheckboxOption(question, index) {
  question.value.splice(index, 1)
},
```

4.2.3 数据结构与组件参数适配

```
<!-- 位置 -->
<template v-else-if="question.type === 'location'">
  <u-button style="width: 100%;" @click="() => (question.control.pickerShow = true)">
    {{ computedLocationDefaultRegion(question).join(' ') }}
  </u-button>
  <u-picker
    v-model="question.control.pickerShow"
    mode="region"
    :default-region="computedLocationDefaultRegion(question)"
    @confirm="param => onLocationConfirm(param, question)"
    :params="computedPickerParam(question.config.locationParams)"
  />
</template>
<!-- 时间 -->
<template v-else-if="question.type === 'datetime'">
  <u-button style="width: 100%;" @click="() => (question.control.pickerShow = true)">
    {{ computedDatetimeDisplayValue(question) }}
  </u-button>
  <u-picker
    v-model="question.control.pickerShow"
    mode="time"
    :default-time="computedDatetimeDefaultTime(question)"
    @confirm="param => onDatetimeConfirm(param, question)"
    :params="computedPickerParam(question.config.datetimeParams)"
  />
</template>
```

```
// 计算属性 | 适配位置到 Picker 默认值
computedLocationDefaultRegion() {
  return question => {
    const { province, city, area } = question.value
    const arr = []
    if (province) {
      arr.push(province)
    }
    if (city) {
      arr.push(city)
    }
    if (area) {
      arr.push(area)
    }
    return arr
  }
},
// 计算属性 | 适配时间到 Picker 默认值
computedDatetimeDefaultTime() {
  return question => {
    const { year, month, day, hour, minute, second } = question.value
    const date = new Date()
    return [
      [year || date.getFullYear(), month || date.getMonth() + 1, day || date.getDate()].join('-'),
      [hour || date.getHours(), minute || date.getMinutes(), second || date.getSeconds()].join(':')
    ].join(' ')
  }
},
// 计算属性 | 适配位置与时间配置到 Picker 配置
computedPickerParam() {
  return params => {
    const obj = {}
    params.filter(item => item.checked === true).forEach(item => (obj[item.value] = true))
    return obj
  }
}
}
```

4.3 登录注册

```
doRegister() {
  uni.request({
    url: "http://localhost:9000/tenant/register",
    method: "POST",
    data: JSON.stringify(this.tenant),
    header: { "Content-Tye": "application/json" },
    success: (response) => {
      const { code, message, data } = response.data;
      if (response.statusCode === 200 && code === 0) {
        uni.showToast({ title: "注册成功" });
        this.refresh();
      } else {
        uni.showToast({ title: message || "无响应" });
      }
    },
  });
},
doLogin() {
  uni.request({
    url: " http://localhost:9000/tenant/login",
    method: "POST",
    data: this.tenant,
    header: { "Content-Type": "application/x-www-form-urlencoded" },
    success: (response) => {
      const { code, message, data } = response.data;
      if (response.statusCode === 200 && code === 0) {
        uni.setStorage({
          key: "login",
          data: { jwt: data.jwt, tenant: data.tenant },
        });
        uni.showToast({ title: "登录成功" });
        this.refresh();
      } else {
        uni.showToast({ title: message || "无响应" });
      }
    },
  });
},
doLogout() {
  uni.removeStorage({
    key: "login",
    success: () => {
      uni.showToast({ title: "注销成功" });
      this.refresh();
    },
  });
},
```

4.4 填写问卷

4.4.1 拉取模板

```
pullQuestionnaire() {
  const { jwt, tenant } = uni.getStorageSync("login");
  if (
    this.questionnaireItem.tenantId === "" ||
    this.questionnaireItem.questionnaireId === ""
  ) {
    return;
  }
  uni.request({
    url: `http://localhost:9000/${this.questionnaireItem.tenantId}/questionnaire/find/id/${this.questionnaireItem.questionnaireId}`,
    method: "GET",
    success: (response) => {
      const { code, message, data } = response.data;
      if (response.statusCode === 200 && code === 0 && data) {
        this.questionnaireItem.questions = JSON.parse(data.questions);
        uni.showToast({ title: "拉取成功" });
      } else {
        uni.showToast({ title: "拉取失败" });
        this.questionnaireItem.questions = [];
      }
    },
  });
},
```

五、总结

5.1 踩坑与解决方案

5.1.1 Jackson 序列化含转义字符的字符串失败

> 对字段进行自定义序列化：

```
@Data
@Accessors(chain = true)
@TableName("questionnaire")
public class Questionnaire {
    @JsonProperty(access = JsonProperty.Access.READ_ONLY)
    @TableId(value = "id", type = IdType.ASSIGN_ID)
    private String id;

    @JsonProperty(access = JsonProperty.Access.READ_ONLY)
    @TableField("tenant_id")
    private String TenantId;

    @TableField("title")
    private String title;

    @TableField("description")
    private String description;

    @JsonDeserialize(using = QuestionsDeserializer.class)
    @TableField("questions")
    private String questions;
}
```

QuestionsDeserializer.java

```
package xyz.xgh.questionnaire.questionnaire.util;

import ...

public class QuestionsDeserializer extends JsonDeserializer<String> {
    @Override
    public String deserialize(JsonParser p, DeserializationContext ctxt) throws IOException {
        return p.getText();
    }
}
```

5.1.2 Uniapp 请求获取不到自定义响应头 `Authorization`

> 不得已只能改变接口的返回值：

```java
public AuthenticationSuccessHandler successHandler() {
    return (request, response, authentication) -> {
        String username = request.getParameter( name: "username");
        String password = request.getParameter( name: "password");
        Tenant find = tenantService.findTenantByUsername(username);
        String jwt = JwtUtil.createJws(username, password);
        // TODO dto width success response
        // response.setHeader(JwtUtil.HEADER, jwt);
        Map<String, Object> map = new HashMap<>();
        map.put("jwt", jwt);
        map.put("tenant", find);
        R<?> r = R.success(map);
        HttpServletUtil.responseJson(response, r);
    };
}
```

5.2 问题分析与经验总结

由于前期没有及时确定具体的实践题目，在折腾 Vert.x + Vant + JSX 上耽误了很多时间，其中也包括了尝试做其它题目的耽误，因此最终实践赶的比较着急并且还拖了很久。尽可能地确定题目并指定计划尤为重要。

由于 uniapp H5 端还不支持 Vue3，并且没找到 IDEA 系列对 `rpx` 等单位友好支持的解决方案，没有 TS 的类型支持加之我们对 Options-API 的运用不当，我们在编写页面时频繁的遇到了很多难以定位的小问题。之后可能会优先考虑易用度其次再是美观度。

我们并没有很好的利用移动端的优势，反而保留了很多 PC 习惯，我们的界面稍显奇怪并且交互并不是特别友好。我们可能需要去了解一些关于移动端地最佳实践。