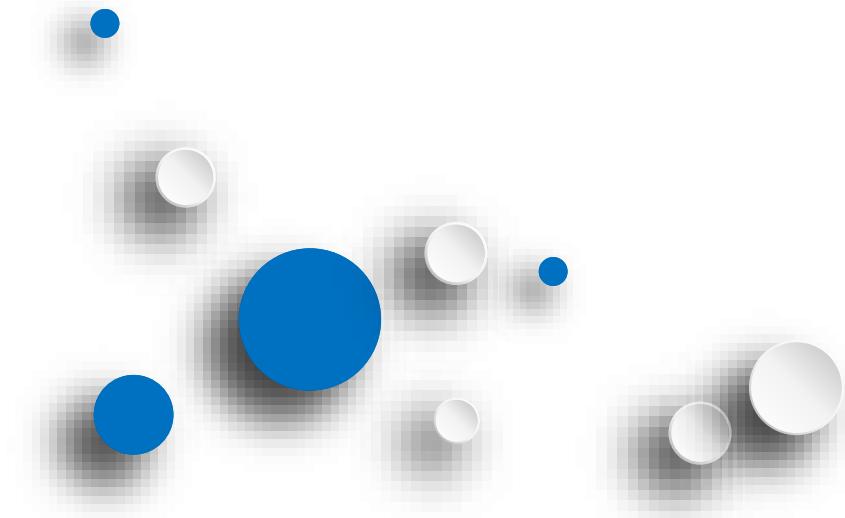


基于RESTful API的 前后端交互

前端页面调用RESTful接口





理解Restful接口

KES Plus采用前后端分离架构实现Web应用。在数据表上“生成代码”时，会自动创建一系列 RESTful 接口，前端页面正是调用这些接口获取和操作数据库信息。

RESTful API 是一种基于 HTTP 协议的接口设计风格，即“**前端发请求、后端给响应**”。

- 前端通过HTTP/HTTPS网络请求说明要访问服务器应用程序后端API的URL地址
- 后端执行API，返回JSON格式数据给前端
- 前端整合到HTML页面中展示。



使用RESTful API可以支持兼容多种终端的开发项目，标准化的JSON数据适配任意前端框架。



KES Plus自动生成的RESTful资源

RESTful API将所有数据视为资源，用URL路径表示。例如，“/courses/subject/list”接口，可以从后端数据库获取学科列表数据。默认创建的RESTful接口有6个：查看、分页、列表、删除、新增、修改。

The screenshot shows the KES Plus interface with a sidebar containing icons for file operations, database development, search, navigation, and a grid with a '7' notification. The main area displays a tree structure under '金仓快速开发与运维平台'. The '数据库开发' node has a 'RESTful' child node, which is expanded to show '课程信息'. A red box highlights a list of generated RESTful methods for '课程信息':

- GET 查看
- GET 分页
- GET 列表
- POST 删除
- POST 新增
- PUT 修改

默认生成的RESTful资源包含常规**增删改查**操作

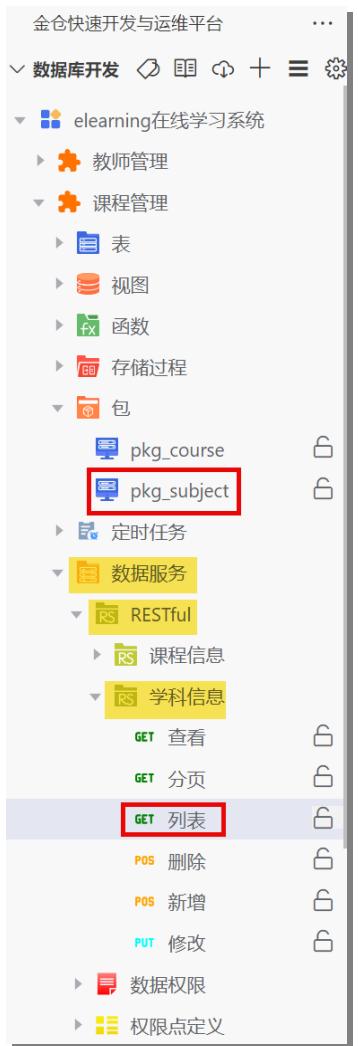
- **GET查看、GET分页、GET列表，都是查询操作。**
 - 查看是按照给定id查询某一个记录
 - 分页和列表都是获取所有记录。（简单理解）列表一次性获取所有记录，分页是按照给定分支索引（页码）获取，一次只获取一部分
 - 分页适合数据量大的情况，适合列表组件的分页展示。
- **POST删除**
- **POST新增**
- **PUT修改**

理解RESTful接口

如，“/courses/subject/list” 接口

pkg_course.kpkg 是一个封装了特定功能模块的包文件，包含了课程管理的核心逻辑。

RESTful接口是这个包对外提供服务的方式之一。



查看接口

```
/* @Description 列表查询 (已发布RESTful:/courses/subject/list)
 * @param jsonData json信息查询条件
 * @Return json
 */
FUNCTION list(jsonData jsonb) RETURNS JSONB AS
DECLARE
    v_condition_sql text;
    v_orderby_sql text;
    v_result_json jsonb;
    v_error_data jsonb;
    v_execute_list_query_sql text;
BEGIN
    --构建查询条件
    v_condition_sql := 'where '|| kesplus_platform.search_model.getConditionSql(jsonData, CONDITIONS);

    --构建排序规则
    v_orderby_sql := "kesplus_platform"."utils".get_order_by_sql(jsonData -> 'orderColumns');

    --查询语句,其中查询条件和排序规则使用占位符
    v_execute_list_query_sql :='
        SELECT t1.*
        from (
            SELECT ROWNUM AS "rn", t1.id as id,t1.SubjectCode as subjectCode,t1.SubjectName as subjectName
            FROM courses.subject t1
    ';

    EXECUTE kesplus_platform.json.array_sql(format(v_execute_list_query_sql,v_condition_sql,nvl(v_orderby_sql,' '))) INTO v_result_json;

    RETURN v_result_json;
EXCEPTION
    WHEN OTHERS THEN
        v_error_data :=kesplus_platform.EXCEPTION.errdata(SQLSTATE);
        kesplus_platform.business_exception(v_error_data->>'code',v_error_data->>'msg');
END list;
```

打开课程管理模块的“包→pkg_subject”查看RESTful API接口及代码



理解RESTful接口

JSON是一种轻量级文本数据格式，以键值对描述结构化数据。

- **JSON对象**: 是用{}包裹的无序键值 ("key": value) 对集合，键和值用冒号分隔，键值对之间用逗号分隔，例如：{"name": "张三", "age": 25, "isStudent": false}。键必须是字符串，值可以是字符串、数字、布尔、对象、数组等。
- **JSON数组**: 是用[]包裹的有序元素集合，元素可以是JSON对象、数组等类型，元素间用逗号分隔；例如：["苹果", 123, {"color": "红色"}, [1, 2, 3]]。

The screenshot shows a database development environment with a sidebar navigation and a main code editor area.

- ① In the sidebar under "RESTful" → "学科信息", the "GET 列表" (Get List) option is highlighted with a red box.
- ② At the top right, there is a button labeled "运行RESTful" (Run RESTful) with a red box around it.
- ③ In the code editor, there is a modal dialog titled "保持默认输入" (Keep Default Input) with a red box around it. It contains the placeholder "按Enter" (Press Enter).
- ④ In the bottom right panel, the output of the API call is shown as a JSON array. One object in the array is highlighted with a red box and labeled "subject表的一条记录" (A record from the subject table). The JSON data is as follows:

```
[{"id": "099298a81d7340049b22854013436bcc", "rn": 6, "subjectCode": "S07", "subjectName": "文学"}, {"id": "69cb1f9ae64d48fe9b9c67be43a79916", "rn": 7, "subjectCode": "S08", "subjectName": "工程"}, {"id": "3c7d26bfa79e433b8fd647e7477604e6", "rn": 8, "subjectCode": "S09", "subjectName": "法学"}]
```

输出结果是一个JSON数组，数组元素是JSON对象。



自定义RESTful接口及调用

如果用户需要更复杂的功能，例如多表联接查询、统计汇总、事务操作等，需要创建自定义函数来实现。

自定义函数执行一系列操作后返回一个结果。

- 函数可以0个或多个参数作为输入值，输入值可以是各种基本数据类型或JSON对象；
- 函数返回值可以是一个标量值，例如数字、字符串、日期等，也可以是一个表，即包含多行多列的结果集。一般建议把**返回类型**设为**JSON**类型，将需返回的所有数据存储在一个JSON对象中。由于**JSONB**支持索引且处理速度更快，更适合于存储和查询，后续示例采用JSONB类型。

为自定义函数创建RESTful接口并配置功能权限，以便在前端页面访问自定义函数。

新建页面和菜单，调用自定义RESTful接口。

【例7.1】创建自定义功能，实现按学号查询加权平均分。

【例7.1】创建自定义功能，实现按学号查询加权平均分。

查询结果是一标量值，在SELECT语句中使用jsonb_build_object函数创建一个键为score、值为加权平均分的JSONB对象并作为函数的返回值。

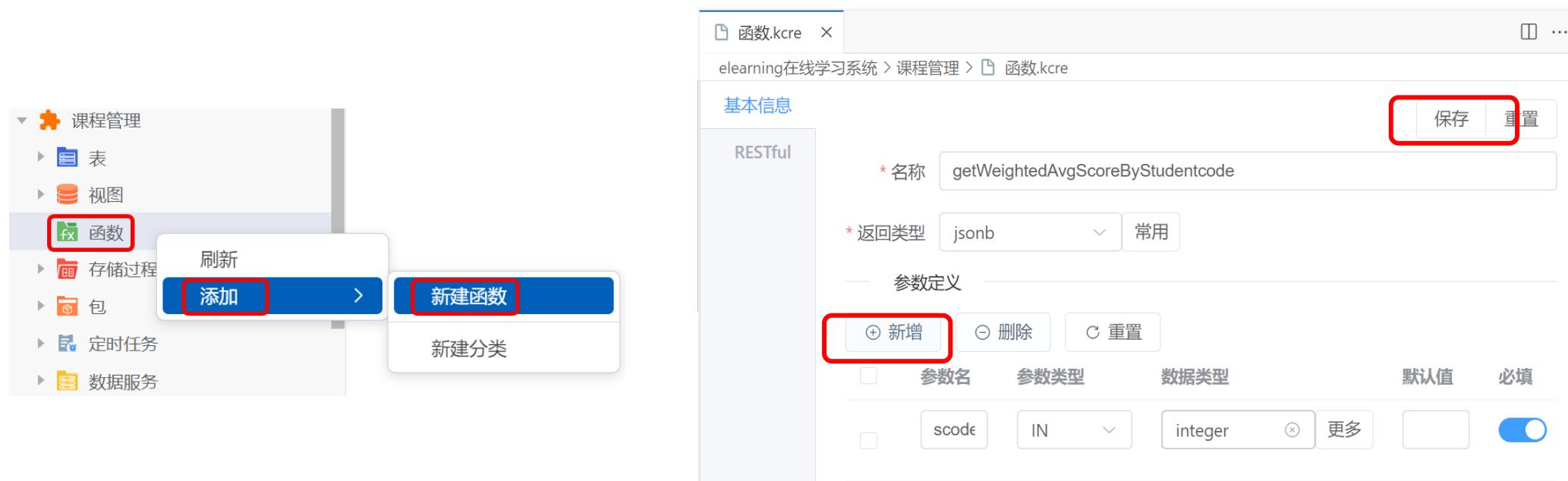


加权平均分：

$$\text{sum}(\text{Credits} * \text{Score}) / \text{sum}(\text{Credits})$$

例7.1 第一步 创建自定义函数→创建RESTful接口并配权限→调用自定义RESTful接口

在课程管理模块右击“函数”选“添加→新建函数”命令；在打开的函数.kcre页中填写名称getWeightedAvgScoreByStudentcode、**返回类型jsonb**。点击参数定义“新增”按钮，填写参数名scode、参数类型IN、数据类型integer、设为必填后保存。



例7.1 第一步 创建自定义函数→创建RESTful接口并配权限→调用自定义RESTful接口

在打开页面中编写函数代码后保存。

```
CREATE FUNCTION getWeightedAvgScoreByStudentcode (scode integer) RETURN jsonb AS
DECLARE
    v_result jsonb;
BEGIN
    SELECT jsonb_build_object(
        'score', round(sum(Credits*Score)/sum(Credits),1) --计算加权平均分并保留一位小数
    ) INTO v_result
    FROM courses.course c1
    JOIN courses.courseenroll c2 ON c1.CourseCode=c2.CourseCode
    JOIN students.student s ON c2.StudentCode=s.StudentCode
    WHERE s.StudentCode=scode;
    RETURN v_result;
END
```

例7.1 第一步 创建自定义函数→创建RESTful接口并配权限→调用自定义RESTful接口

选页面右肩的按钮“”运行测试函数功能。在弹出窗口填写1001后按回车键，在输出窗口查看该生的加权平均成绩65.4。说明函数可正常调用。



The screenshot shows a database development environment with a sidebar navigation menu and a central code editor.

- ① 点击运行按钮**: A red circle highlights the "Run" button (play icon) in the top right corner of the interface.
- ② 输入学号**: A red circle highlights the input field where "1001" has been entered.
- ③ 按Enter键，运行函数**: A red circle highlights the F12 key, indicating to press Enter to run the function.
- ④ 查看运行结果**: A red circle highlights the output window showing the result: { "score": 65.4 }.

The code in the editor is:

```
1 create or replace function "courses"."getWeighedAvgScoreByStudentcode"(scode integer) return jsonb as
2 /**
3  * @Description
4  * @Param scode
5  * @Return jsonb
6  */
7 declare
8     v_result jsonb;
9 begin
10    SELECT json_build_object(
11        'score', round(sum(Credits*Score)/sum(Credits),1)
12    ) INTO v_result
13    FROM courses.course c1
14    JOIN courses.courseenroll c2 ON c1.CourseCode=c2.CourseCode
15    JOIN students.student s ON c2.StudentCode=s.StudentCode
16    WHERE s.StudentCode=scode;
17    return v_result;
18 end;
```

The bottom status bar shows: master*+ Launchpad kesplus http://localhost:54801 0 0 0 UTF-8 LF KSQL 由您于2025-08-04 23:08:29锁定 Prettier EN ZH-CN

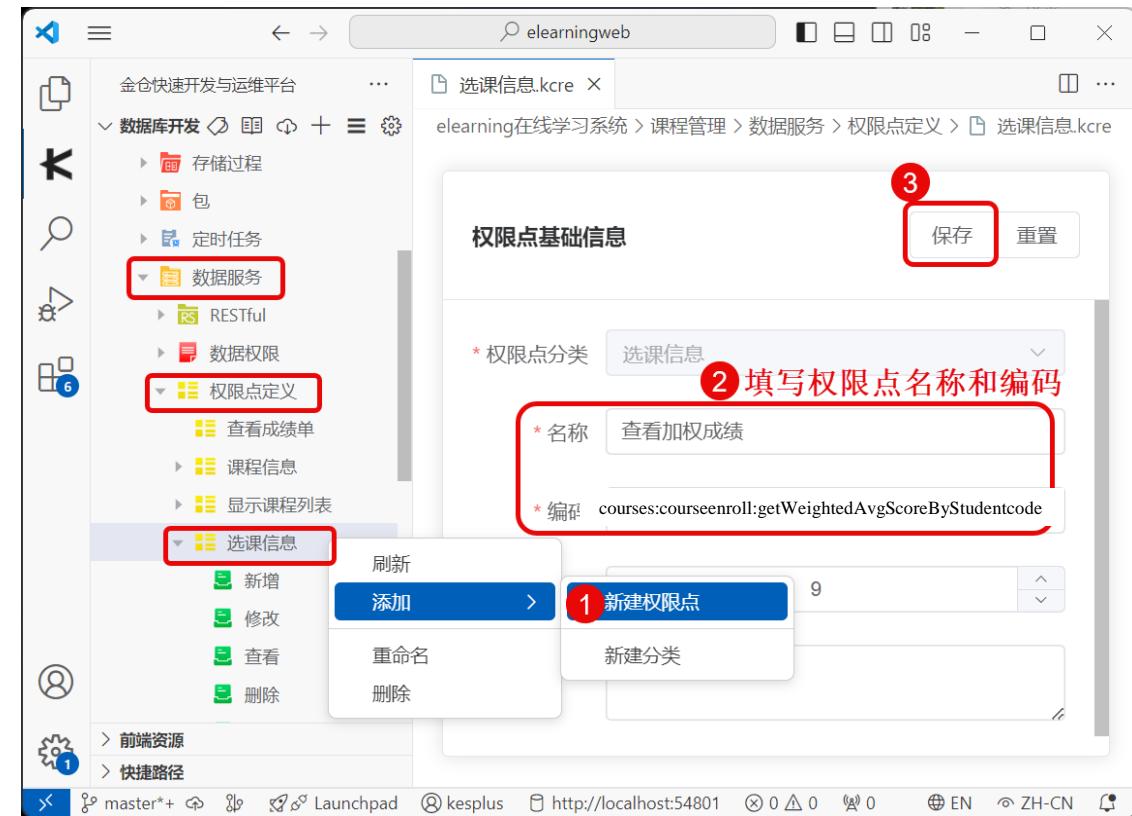
例7.1 创建自定义函数→ 第二步 创建RESTful接口并配权限→ 调用自定义RESTful接口

为在前端页面访问自定义函数，需为自定义函数创建RESTful接口并配置功能权限。具体操作：

- 首先创建一个功能权限点
- 然后创建一个与自定义函数和功能权限点关联的RESTful接口
- 最后将该功能权限点授权给特定角色。

① 在课程管理模块，展开“数据服务→权限点定义→**选课信息**”，右击选“添加→新建权限点”命令，填写名称“查看加权成绩”、编码“courses:courseenroll:getWeightedAvgScoreByStudentcode”，保存。

也可设为“根节点”



例7.1 创建自定义函数→第二步 创建RESTful接口并配权限→调用自定义RESTful接口

② 找到getWeighedAvgScoreByStudentcode函数，右击选“查看”命令，在打开页面的左侧选RESTful标签页，填写/选择如下设置，然后保存：

发布RESTful：是，

RESTful名称：getWeighedAvgScoreByStudentcode

分类目录：选课信息

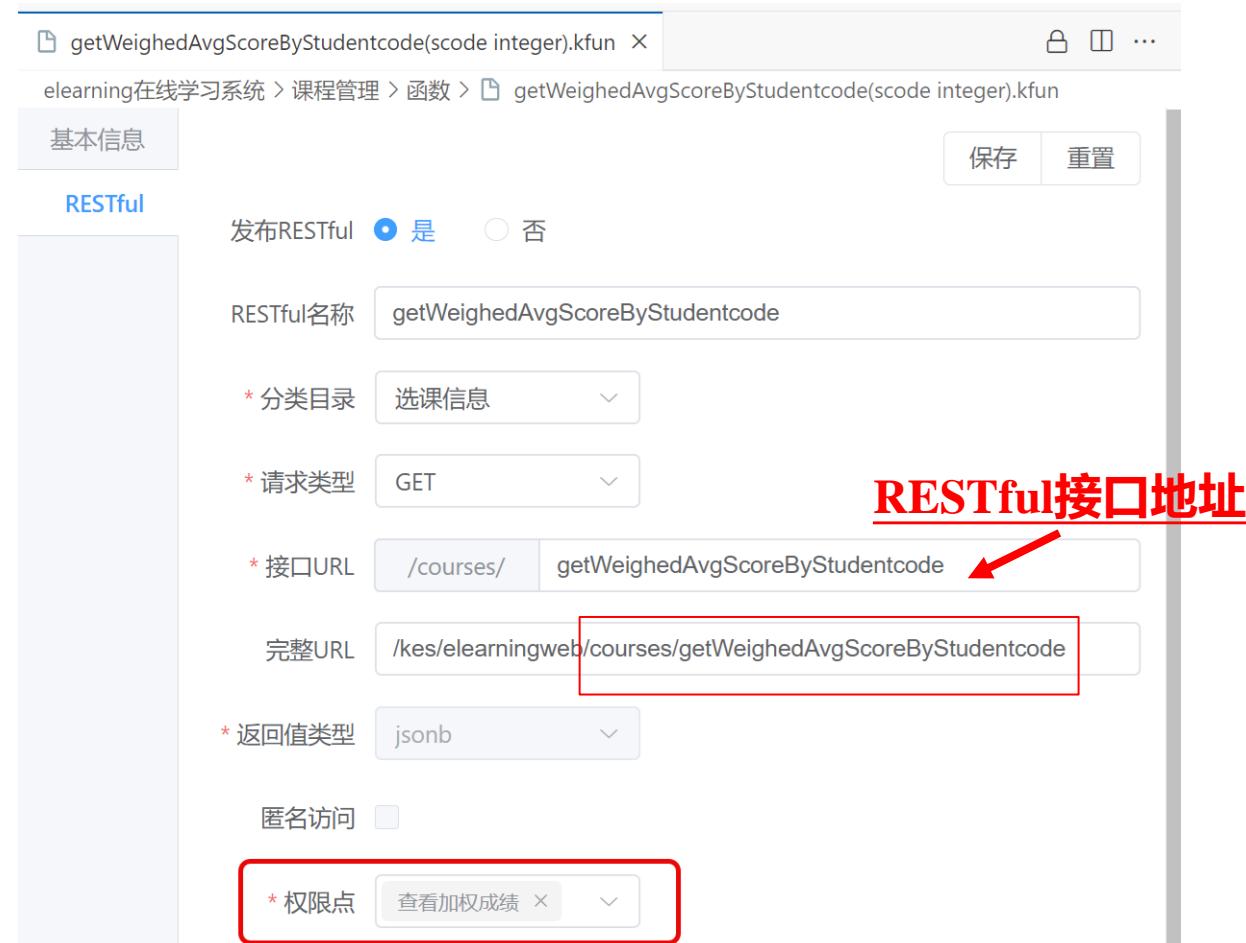
请求类型：GET

接口URL：/courses/getWeighedAvgScoreByStudentcode

权限点：查看加权成绩

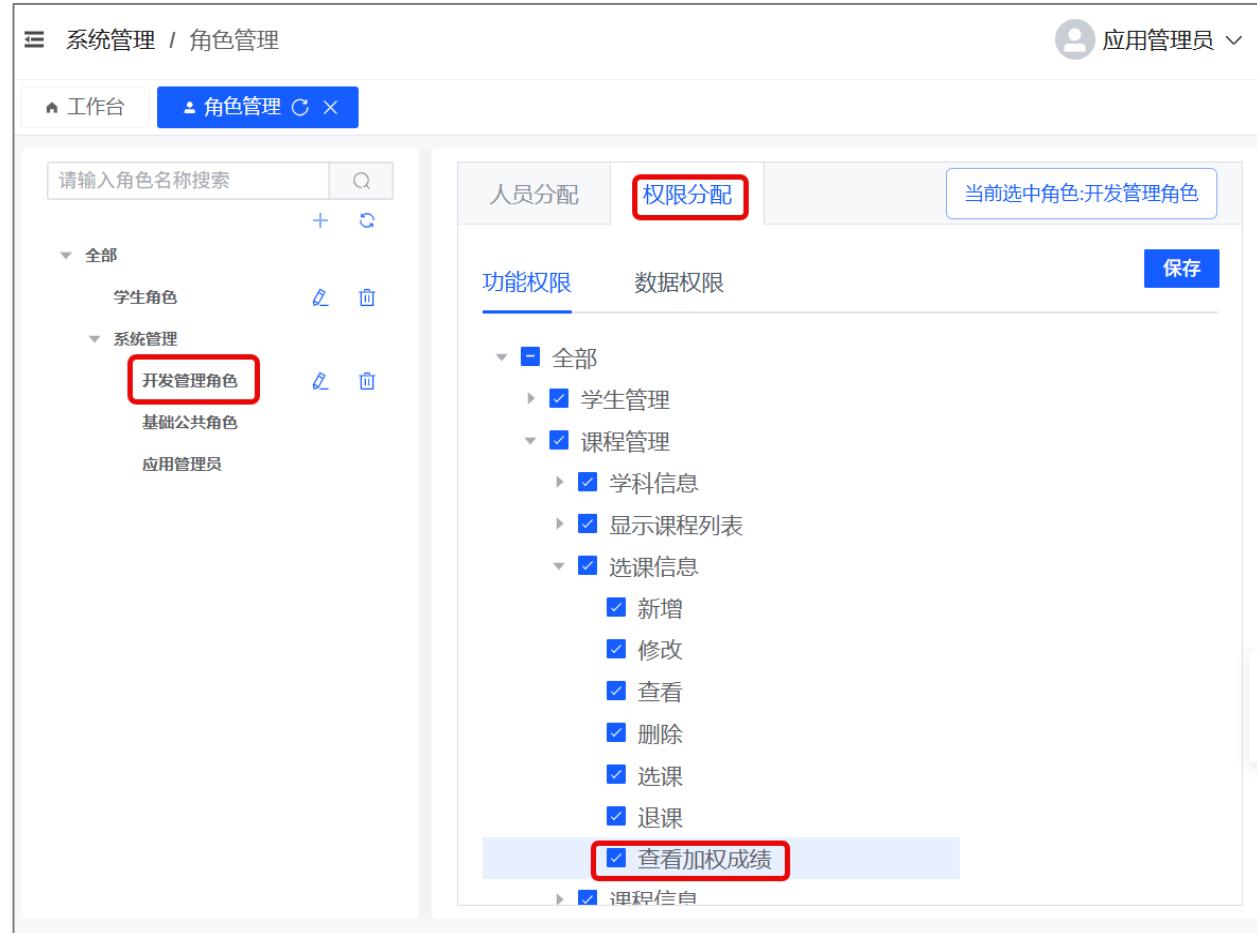
之后可在课程管理模块中的“数据服务→RESTful”的选课信息目录下，查看或修改该RESTful接口信息。

在“课程管理\数据服务\RESTful\选课信息”下，选中该接口，点击“”运行该接口，可看到与函数运行相同的结果。



例7.1 创建自定义函数→第二步 创建RESTful接口并配权限→调用自定义RESTful接口

③ 启动应用，以应用管理员admin身份登录，展开“系统设置→角色管理”，为开发管理角色勾选功能权限中的“查看加权成绩”，之后即可在前端页面中访问该接口。

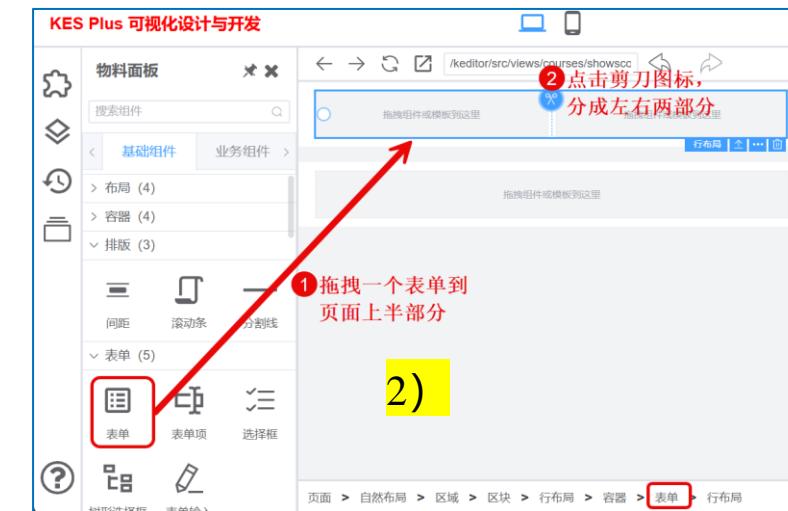


例7.1 创建自定义函数→创建RESTful接口并配权限→第三步 调用自定义RESTful接口

为调用自定义RESTful接口，需新增页面和菜单并授权，具体步骤：

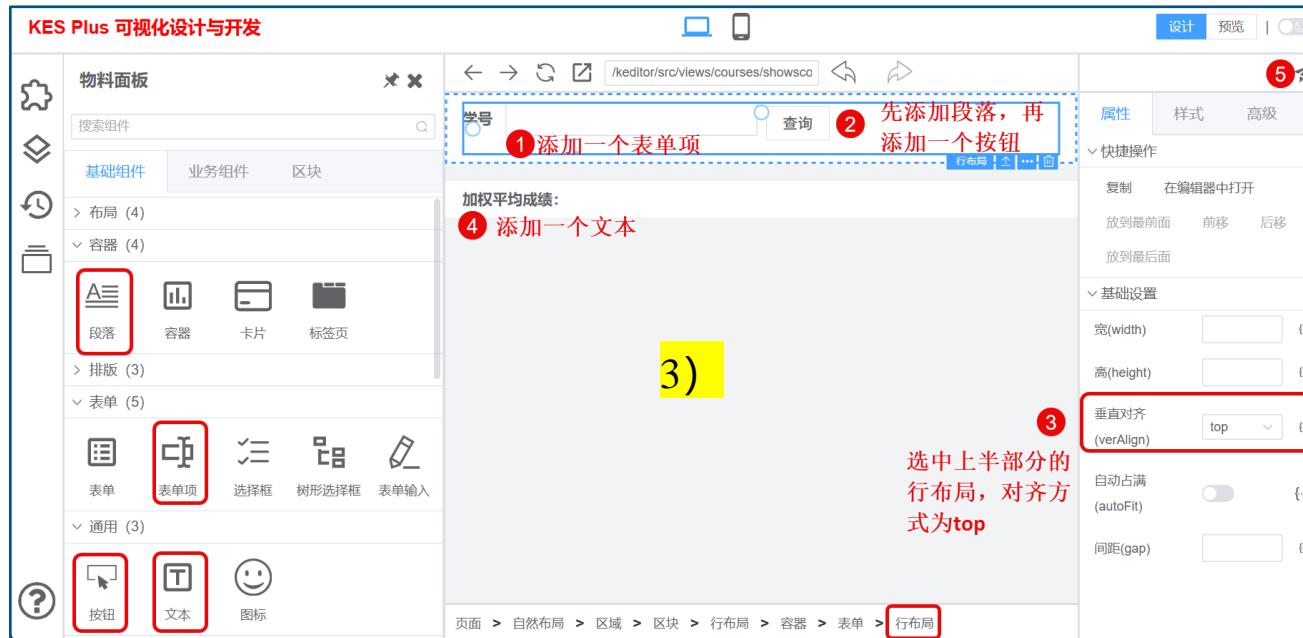
- 新增页面并编辑页面代码，调用RESTful接口；
- 新建菜单并链接到该页面
- 创建一个与菜单对应的权限点；
- 启动应用后，将刚创建的菜单和权限点授权给开发管理角色。

① 新增一个前端页面查看成绩单。1) 打开KES Plus可视化设计与开发页面，新增一个页面，选择或填写上级目录courses、文件名称showscore_v1.vue、模板“上下.vue”。2) 在页面上部添加一个表单后，剪成左右两列。



例7.1 创建自定义函数→创建RESTful接口并配权限→第三步 调用自定义RESTful接口

3) 上部分别添加一个表单项用于接收用户输入的学号、一个查询按钮；下部添加一个文本组件，显示“加权平均成绩：”。4) 为了获取用户输入的学号，将上部学号处的输入框和变量studentCode双向绑定。



例7.1 创建自定义函数→创建RESTful接口并配权限→第三步 调用自定义RESTful接口

② 修改前端页面showscore_v1.vue的代码

i. 修改script部分代码

在编辑器中打开该前端页面showscore_v1.vue的代码，在script部分中添加查询按钮的单击事件处理函数getScoreByCode，调用RESTful接口，获取按学号查询的加权平均成绩。

```
<script setup>
import { reactive } from 'vue'
import { ElMessage } from 'element-plus' // 导入消息提示组件
const form = reactive({})
const studentCode = ref(null) // 存储用户输入的学号，ref将基本数据类型转换为响应式数据
const scoreData = ref(null) // 存储查询到的加权平均成绩
// 异步函数，根据学号查询成绩，即查询按钮的事件处理程序
const getScoreByCode = async () => {
  if (!studentCode.value) { // 1. 确保学号不为空，否则给出提示并终止
    ElMessage.warning('请输入学号！')
    return;
  }
  scoreData.value = null // 2. 清空旧数据和错误信息
  try {
```

// 3. 发送 GET 请求
const response = await request.get('/courses/getWeightedAvgScoreByStudentcode', {
 params: {
 scode: studentCode.value
 }
});
// 4. 处理响应
if (response && response.score !== undefined && response.score !== null) {
 scoreData.value = response.score
 ElMessage.success('查询成功。')
} else {
 ElMessage.warning('未找到该学号的成绩。')
}
} catch (error) {
 // 5. 捕获并处理请求失败
 ElMessage.error('查询失败')
};
</script>

RESTful接口地址

接口参数和值地址

例7.1 创建自定义函数→创建RESTful接口并配权限→第三步 调用自定义RESTful接口

ii. **更新查询按钮处的代码**，给按钮的单击事件绑定事件处理函数getScoreByCode，表示单击该按钮时调用script中的getScoreByCode函数，修改后的代码如下：

```
<el-button @click="getScoreByCode"><el-text>查询</el-text></el-button>
```

更新显示加权平均成绩处的代码，{{ scoreData }}是插值表达式，表示在页面上显示scoreData变量的当前值，修改后的代码如下：

```
<KText :strong="true">加权平均成绩：{{ scoreData }}</KText>
```

例7.1 创建自定义函数→创建RESTful接口并配权限→第三步 调用自定义RESTful接口

③ 新建菜单并链接到页面

为页面showscore_v1.vue创建菜单。

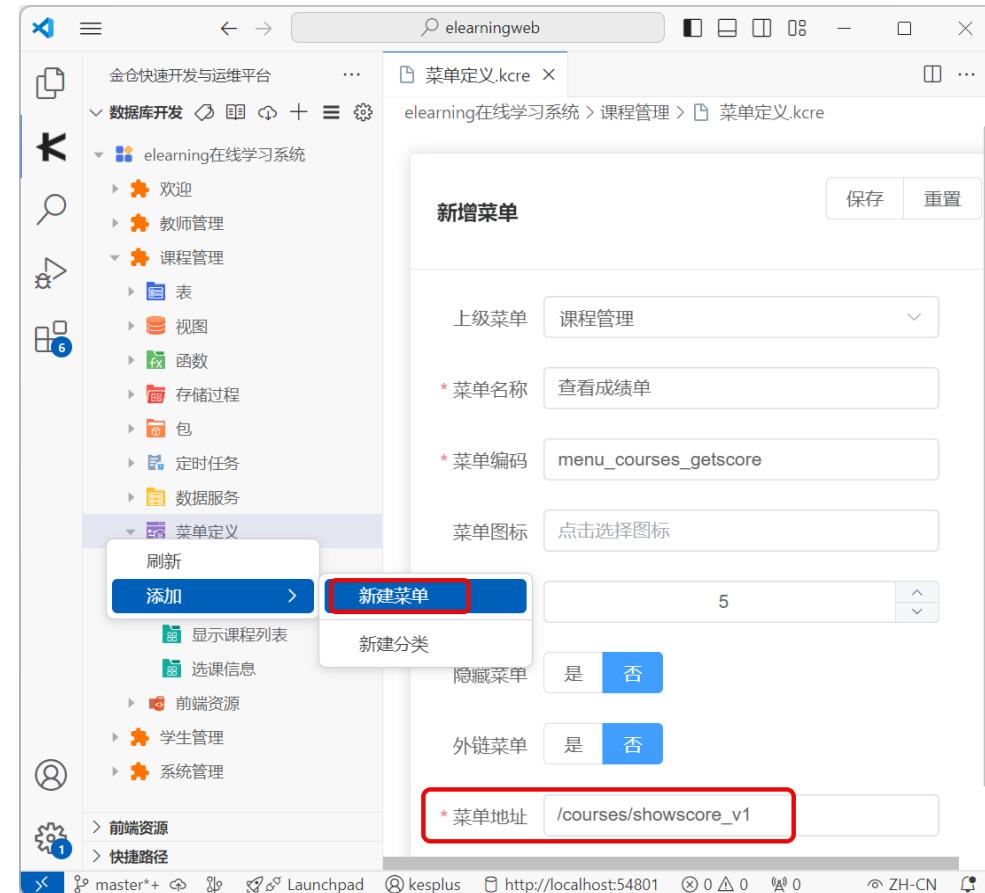
在课程管理的菜单定义处，右击选“新建菜单”，填写：

菜单名称：查看成绩单

菜单编码：menu_courses_getscore

菜单地址：/courses/showscore_v1

然后，为开发管理角色添加该页面访问权限

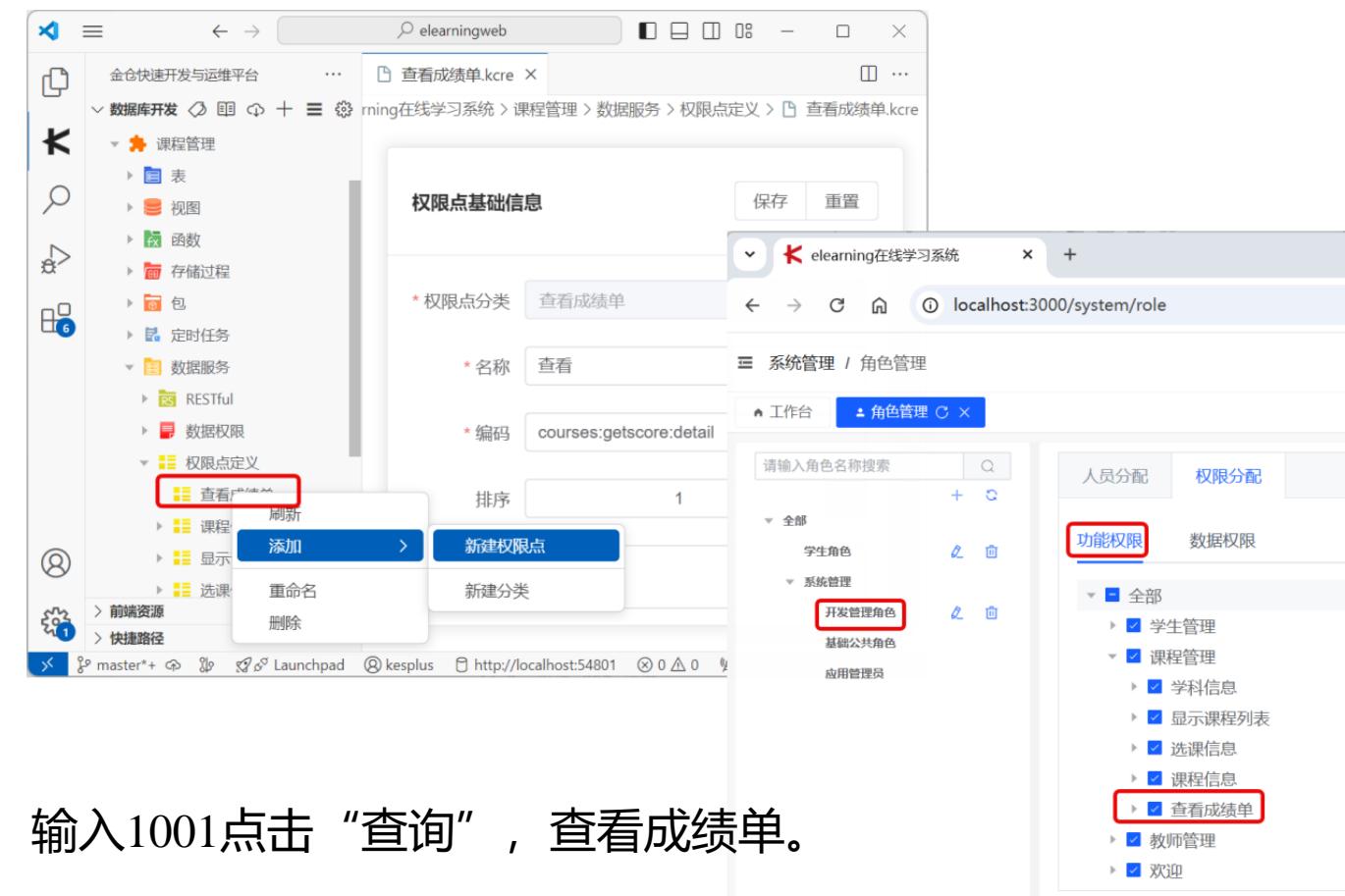


例7.1 创建自定义函数→创建RESTful接口并配权限→第三步 调用自定义RESTful接口

④ 创建一个与“查看成绩单”菜单对应的权限点。

找到课程管理模块下的“数据服务→权限点定义→查看成绩单”，该权限点分类创建菜单后自动生成，右击该分类从快捷菜单中选“添加→新建权限点”命令，填写名称“查看”，编码“courses:getscore:detail”。

应用启动后，将刚创建的菜单和权限点授权给开发管理角色。



⑤ 测试：应用启动，打开查看成绩单页面，输入1001点击“查询”，查看成绩单。



调用自定义RESTful接口

【例7.2】实现按学号查询加权平均分和选课列表。

本例调用后端Restful接口获得汇总结果。

The screenshot shows a web browser window for the "elearning 在线学习系统". The URL is `localhost:3000/courses/showscore_v2`. The page title is "课程管理 / 首页". On the left, there is a sidebar with navigation items: "工作台", "欢迎", "教师管理", "课程管理" (which is expanded), "查看成绩单" (selected and highlighted in blue), "选课信息", and "显示课程列表". The main content area has a green success message "查询成功.". It includes a search bar with "学号" input set to "1001" and a "查询" button. Below this is a table titled "我的成绩单" (My Grade List) with columns: "课程号" (Course ID), "课程名" (Course Name), "学分" (Credits), "学时" (Hours), and "成绩" (Grade). The table data is:

课程号	课程名	学分	学时	成绩
C003	计算机网络技术	3	48	93
C005	心理学与生活	1.5	30	
C002	信息系统与数据库技术	4	64	65
C001	多媒体技术及应用	2.5	40	72

At the bottom, there are summary statistics: "加权平均成绩: 65.4", "选修课程数: 4", and "总学时: 182".

例7.2 第一步 创建自定义函数→创建RESTful接口并配权限→调用自定义RESTful接口

① 创建函数getStatByStudentcode，实现根据学号查询所有选修课程的成绩。在课程模块中

查询结果是一个包含多行多列的数据表，代码中jsonb_build_object函数将每行存储在一个JSONB对象中，jsonb_agg函数将多个JSONB对象组合成一个JSONB数组存储到v_result，并将它作为函数的返回值。

```
CREATE FUNCTION getStatByStudentcode(stuinfo jsonb) RETURNS jsonb AS
DECLARE
    v_input_student_code INTEGER; -- 用于存储从输入的JSONB对象中提取的学号
    v_weighted_score NUMERIC; -- 用于存储计算出的加权成绩
    v_course_count BIGINT; -- 用于存储选修课程的数量
    v_result jsonb; -- 用于存储所有选修课程的信息
    v_student_exists BOOLEAN := FALSE; -- 标志,用于判断学生是否存在
    v_status TEXT; -- 用于存储查询状态(成功/失败)
    v_message TEXT; -- 用于存储状态消息
BEGIN
    v_input_student_code := (stuinfo->>'studentcode')::INTEGER; -- 从输入JSONB中提取学号
    -- 1. 检查学生是否存在于 students 表中
    SELECT EXISTS (
        SELECT 1 -- 只需要判断是否存在, 不需要返回具体数据
        FROM students.student
        WHERE StudentCode = v_input_student_code
    ) INTO v_student_exists;
    -- 如果学生不存在,则设置失败状态并返回结果
    IF NOT v_student_exists THEN
        v_status := 'Failure';
        v_message := '未找到学号为 ' || v_input_student_code || ' 的学生。';
        RETURN jsonb_build_object(
            'studentcode', v_input_student_code,
            'status', v_status,
            'message', v_message
        );
    END IF;
    -- 2. 如果学生存在,则计算其加权成绩和选修课程数量, NULLIF 避免除以零
    SELECT ROUND(SUM(c.credits * ce.score) * 1.0 / NULLIF(SUM(c.credits), 0), 1),
           COUNT(DISTINCT c.coursecode) -- 选修课程数量
    INTO v_weighted_score, v_course_count
    FROM courses.course AS c
    JOIN courses.courseenroll AS ce ON c.coursecode = ce.coursecode
    WHERE ce.studentcode = v_input_student_code; -- 根据学号过滤选课记录
```

```
--3. 查询所有选修课程的信息
SELECT jsonb_agg(
    jsonb_build_object(
        'coursecode', c1.coursecode,
        'coursetname', coursetname,
        'credits', credits,
        'hours', hours,
        'score', c2.score
    )
) INTO v_result
FROM courses.course c1 JOIN courses.courseenroll c2 ON c1.coursecode=c2.coursecode
JOIN students.student s ON c2.studentcode=s.studentcode
WHERE s.studentcode=v_input_student_code;
-- 4. 根据课程记录数量判断最终状态和消息
IF v_course_count IS NULL OR v_course_count = 0 THEN
    v_status := 'Failure';
    v_message := '学号为' || v_input_student_code || '的学生没有有效的课程记录来计算成绩。';
    v_weighted_score := NULL; -- 确保加权成绩为 NULL
    v_course_count := 0; -- 确保课程数量为 0
ELSE
    v_status := 'Success';
    v_message := '查询成功。';
END IF;
-- 5. 构建并返回最终的 JSON 对象
RETURN jsonb_build_object(
    'studentcode', v_input_student_code,
    'score', v_weighted_score,
    'coursecount', v_course_count,
    'courses', v_result,
    'status', v_status,
    'message', v_message
);
END
```

例7.2 创建自定义函数→ 第二步 创建RESTful接口并配权限→ 调用自定义RESTful接口

② 参照例7.1步骤为自定义函数`getStatByStudentcode`创建RESTful接口, URL地址“/courses/ getStatByStudentcode”。

步骤如下:

- 首先, 在课程管理模块下, **创建一个功能权限点**。填写名称“查看加权成绩2”、编码“courses:courseenroll:getStatByStudentcode”, 保存。
- 然后, **创建一个与本例自定义函数和功能权限点关联的RESTful接口**
查看 `getStatByStudentcode`函数, 打开页面的左侧选RESTful标签页, 填写/选择如下设置, 然后保存:
发布RESTful: **是**,
RESTful名称: **getStatByStudentcode**
分类目录: **选课信息**
请求类型: **GET**
接口URL: /courses/**getStatByStudentcode**
权限点: **查看加权成绩2**
- 最后将该功能权限点**授权**给特定角色。
启动应用, 以应用管理员admin身份登录, 展开“系统设置→角色管理”, 为开发管理角色勾选功能权限中的“查看加权成绩2”, 之后即可在前端页面中访问该接口。

例7.2 创建自定义函数→创建RESTful接口并配权限→第三步 调用自定义RESTful接口

③ 创建页面并调用自定义接口

- i. 复制showscore_v1.vue页面并另存为showscore_v2.vue。在KES Plus平台侧边栏选择资源管理器，选中showscore_v1.vue后鼠标右击选“复制”，将该备份文件重命名为showscore_v2.vue。

打开showscore_v2.vue，在页面下部（即第二个KBlock处）增加一个el-table组件，表格列包含课程号、课程名、学分、学时、成绩五列。注意为el-table的每一列指定数据来源时，属性名称(prop)必须与从后端接口获取的数据列名一致。表格数据来自courseList变量，courseList是一个JSONB数组，每一个元素是一个JSONB对象，包含键coursecode、coursename、credits、hours和score。在加权平均成绩之后增加2个KCell，分别用于显示选修课程数和总学时。

```
<!-- 在页面下半区域增加一个表格 -->
<KBlock title="我的成绩单">
  <KCell>
    <el-table :data="courseList" style="width: 100%" border>
      <el-table-column prop="coursecode" label="课程号"></el-table-column>
      <el-table-column prop="coursename" label="课程名"></el-table-column>
      <el-table-column prop="credits" label="学分"></el-table-column>
      <el-table-column prop="hours" label="学时"></el-table-column>
      <el-table-column prop="score" label="成绩"></el-table-column>
    </el-table>
  </KCell>
  <KCell>
    <KText :strong="true">加权平均成绩: {{ scoreData }}</KText>
  </KCell>
  <KCell>
    <KText :strong="true">选修课程数: {{ counts }}</KText>
  </KCell>
  <KCell>
    <KText :strong="true">总学时: {{ totalHours }}</KText>
  </KCell>
</KBlock>
```

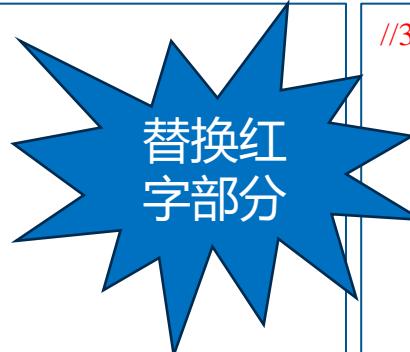


例7.2 创建自定义函数→创建RESTful接口并配权限→第三步 调用自定义RESTful接口

ii. 修改 `showscore_v2.vue` 代码的 `script` 部分，将 Get 请求使用的 RESTful 接口更改为 `getStatByStudentcode` 对应的接口，获取该接口返回的 `response`，从中读取三个键 `score`、`coursecount`、`courses` 对应的值，分别用于显示加权平均成绩、课程数和表格中的课程列表。`courses` 是一个 JSONB 数组，通过累加 `courses` 各个元素的 `hour` 得到总学时。

```
<script setup>
import { reactive } from 'vue'
import { ElMessage } from 'element-plus'
const form = reactive({})
const studentCode = ref(null) //存储用户输入的学号
const scoreData = ref(null) //存储查询到的加权平均成绩
const counts = ref(null) //存储查询到的课程数量
const totalHours = ref(null); //存储计算出来的总学时
const courseList = ref([])

//异步函数，用于根据学号查询成绩
const getCodeByScore = async () => {
  if (!studentCode.value) { //1. 确保学号不为空，否则给出提示并终止
    ElMessage.warning('请输入学号！')
    return;
  }
  scoreData.value = null //2. 清空旧数据和错误信息
  try {
    const stuinfoData = { studentCode: studentCode.value }; //构建包含学号的JSON对象
    const stuinfoJsonString = JSON.stringify(stuinfoData); //将JSON对象转为字符串
  
```



```
//3. 发送 GET 请求
const response = await request.get('/courses/getStatByStudentcode', {
  params: {
    stuinfo: stuinfoJsonString
  }
});
//4. 处理响应
if (response && response.status == 'Success') {
  ElMessage.success('查询成功。')
  scoreData.value = response.score
  counts.value = response.coursecount
  courseList.value = response.courses
  //计算总学时
  totalHours.value = response.courses.reduce((sum, course) => sum +
  course.hours, 0)
} else {
  ElMessage.warning('未找到该学号的成绩。')
}
} catch (error) {
//5. 捕获并处理请求失败
console.error('获取数据失败:', error)
ElMessage.error('查询失败')
}
};

</script>
```

例7.2 创建自定义函数→创建RESTful接口并配权限→第三步 调用自定义RESTful接口

iii. 为页面showscore_v2.vue**创建菜单**。在课程管理的菜单定义处右击选“新建菜单”，填写：

菜单名称：查看成绩单2

菜单编码：menu_courses_getscore2

菜单地址：/courses/showscore_v2

iv. **创建一个与“查看成绩单2”菜单对应的权限点。**

找到课程管理模块下的“数据服务→权限点定义→查看成绩单2”，该权限点分类创建菜单后自动生成，右击该分类从快捷菜单中选“添加→新建权限点”命令，填写：

名称：查看

编码：courses:getscore2:detail

应用启动后，将刚创建的菜单和权限点授权给开发管理角色。

v. **应用启动后**，打开“查看成绩单2”页面，输入1001点击“查询”，查看成绩单。

THANK YOU!

