

# 数据可视化

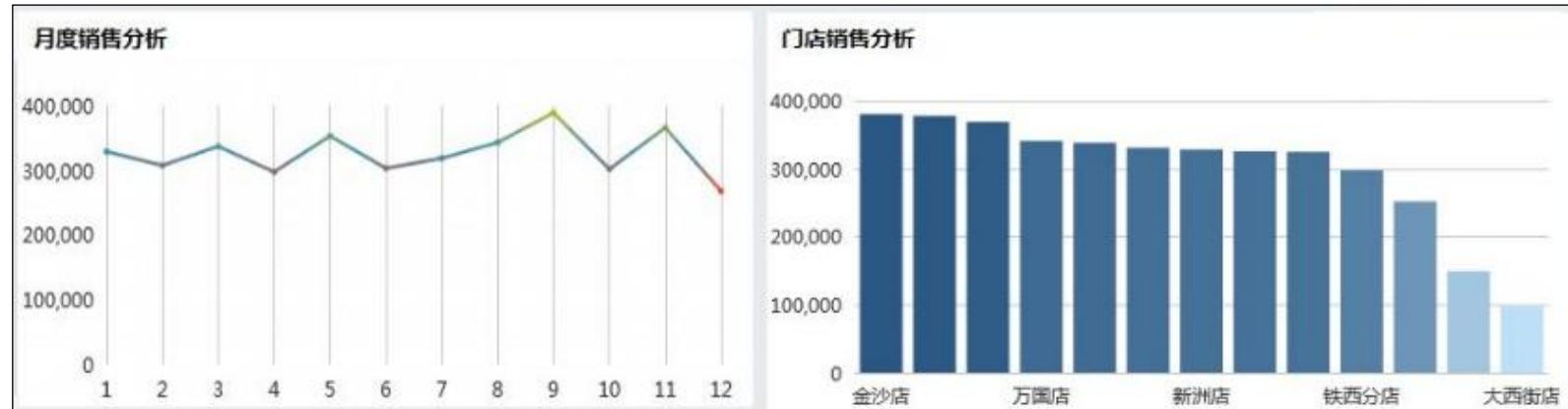




## 认识图表

在Web数据库应用系统中，一些数据需要以图表的形式汇总分析、显示或打印。KES Plus提供了快速开发可视化图表的方法。

图表实现只需两步：选择所需类型的图表，手动配置数据或从后端接口获取数据库的数据。



产品销售数据分析页面

物料面板的图表类型有7种



# 门户页面图表设计——静态图表

依据制作时设定的数据绘制，以后不再变化。通过手动配置图表数据可以快速实现静态数据图表。

## 【例8.1】在门户页面展示学生生源地分布饼图。

- ① 打开“快捷路径”中的门户设计，从图表控件中拖拽一个“饼图”到门户页面下方，呈现默认绘图效果。
- ② 在右侧的组件属性区手动配置属性值完成图表设计。除图表标题外，每组选项值（表头、标题、返回结果、图表类型）对应图表上的1个数据点。
- ③ 本例图表标题设为“学生生源地情况”。添加4个选项，所有选项的表头为“生源地”、图表类型为“pie”；每个选项的标题和返回结果设为地区名称及学生数量：上海5、北京3、黑龙江3、湖北3。
- ④ 点击页面右上方“保存”按钮完成，点“预览”按钮可查看页面效果。

表头相当于数据分组，本例只有一个分组

The screenshot shows the Keshi Plus portal design interface. On the left, the '门户设计' (Portal Design) tab is selected in the sidebar. In the center, there's a preview window titled '学生生源地情况' (Student Source Region Situation) displaying a pie chart with four segments labeled '上海' (Shanghai), '北京' (Beijing), '黑龙江' (Heilongjiang), and '湖北' (Hubei). To the right is the '组件属性' (Component Properties) panel, which is highlighted with a red border. It contains the following configuration:

标题	学生生源地情况
手动配置数据	
选项1	表头 生源地 标题 上海 返回结果 5 图表类型 pie
选项2	表头 生源地

A red arrow points from the '饼图' (Pie Chart) icon in the '图表控件' (Chart Components) section of the main interface towards the pie chart in the preview.

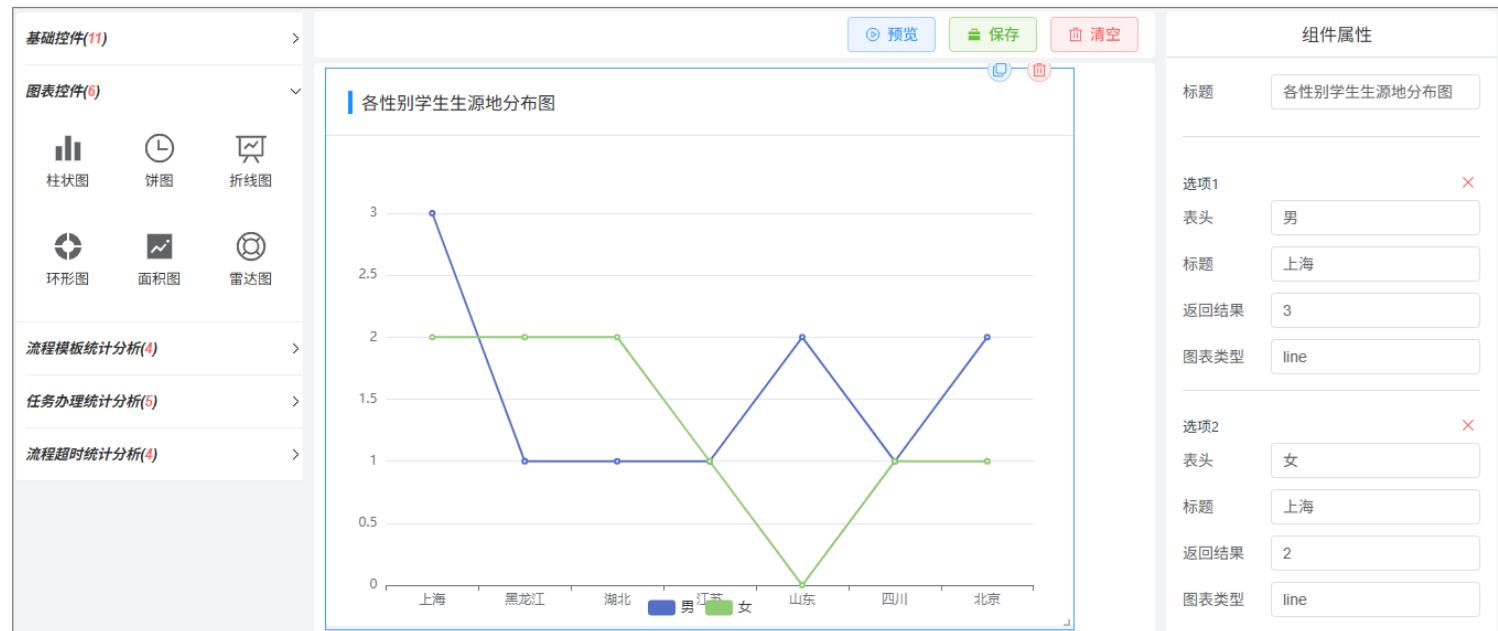
# 静态图表

## 【例8.2】在门户页面绘制折线图展示各生源地男生和女生人数分布。

本例需在一个图表上显示多组值，用表头区分组。

拖拽一个“折线图”，在组件属性区添加14个选项，图表类型“line”；  
**7个表头为“男”，7个为“女”，**标题和返回结果设为某地区相应性别  
的学生数。折线图上两条线，分别对应表头是男、女的两类数据。

本例有2个表头，对应2个分组，  
绘制2条折线





# 门户页面图表设计——动态图表

业务系统经常需要用图表展示从数据库查询动态获取的数据。

页面上的图表可以接收后端接口获取的数据，格式为JSONB数组，每个数据项包括4个属性，即表头（legend）、标题（label）、返回结果（value）、图表类型（type）。例8.2前2个数据项为：

```
[ { legend: '男', label: '上海', value: 3, type: 'line' },  
  { legend: '女', label: '上海', value: 2, type: 'line' }]
```

**【例8.3】通过后端接口获取数据库数据在门户页面展示学生生源地分布饼图。**

The screenshot shows a software interface for designing portal pages. On the left, there's a sidebar with various project and resource management options. In the center, under the '图表控件' (Chart Components) section, a pie chart icon is highlighted with a red box and a large red arrow pointing to its preview window on the right. The preview window displays a dynamic pie chart titled '学生生源地分布图 (动态统计)' (Dynamic Statistics of Student Origin Distribution). The chart shows proportions for different regions: Shanghai (blue), Beijing (green), Shandong (light green), Jiangsu (yellow), Zhejiang (orange), Anhui (pink), Hubei (red), Sichuan (purple), and others like Henan, Shaanxi, Gansu, and Yunnan.

# 动态图表

## ① 创建自定义函数从后端数据库动态查询数据。

在学生管理模块的函数右击选“新建→添加函数”命令，创建自定义函数`getLocationStat`，实现按生源地查询统计学生人数，并按图表数据格式要求返回JSONB数组。运行查看返回的查询结果。

```
CREATE OR REPLACE FUNCTION "students"."getLocationStat"() RETURN JSONB AS
DECLARE
v_result JSONB;
BEGIN
    SELECT jsonb_agg(
        jsonb_build_object (
            'label', s.location,
            'value', COUNT(s.studentcode),
            'legend', '学生生源地情况',
            'type', 'pie'
        )
    )
    INTO v_result
    FROM students.student AS s
    WHERE s.location IS NOT NULL -- 排除 location 为空的记录
    GROUP BY s.location
    ORDER BY COUNT(s.studentcode) DESC; -- 按学生数量降序排序
    RETURN COALESCE(v_result, '[]' :: JSONB); -- 处理没有数据的情况，返回一个空数组[]
END;
```



输出结果是JSONB数组  
一个数据点

```
[{"type": "pie", "label": "上海", "value": 5, "legend": "学生生源地情况"}, {"type": "pie", "label": "北京", "value": 3, "legend": "学生生源地情况"}]
```

# 动态图表

注意操作时可能有延时，可刷新等待权限点、接口等出现

## ② 给自定义函数创建相应的RESTful接口，以及配置相关权限点。

选择学生管理模块下的“数据服务→权限点定义→学生信息”，右击学生信息选“添加→添加权限点”，设置名称“**查看生源地分布**”、编码**students:student:getLocationStat**，保存。

找到getLocationStat函数，右击选“查看”命令，在打开页面选RESTful标签页，填写/选择如下：发布RESful “是”， RESTful名称“**getLocationStat**”，分类目录“**选课信息**”，请求类型“**GET**”，接口URL“**/students/getLocationStat**”，权限点“**查看生源地分布**”，保存。

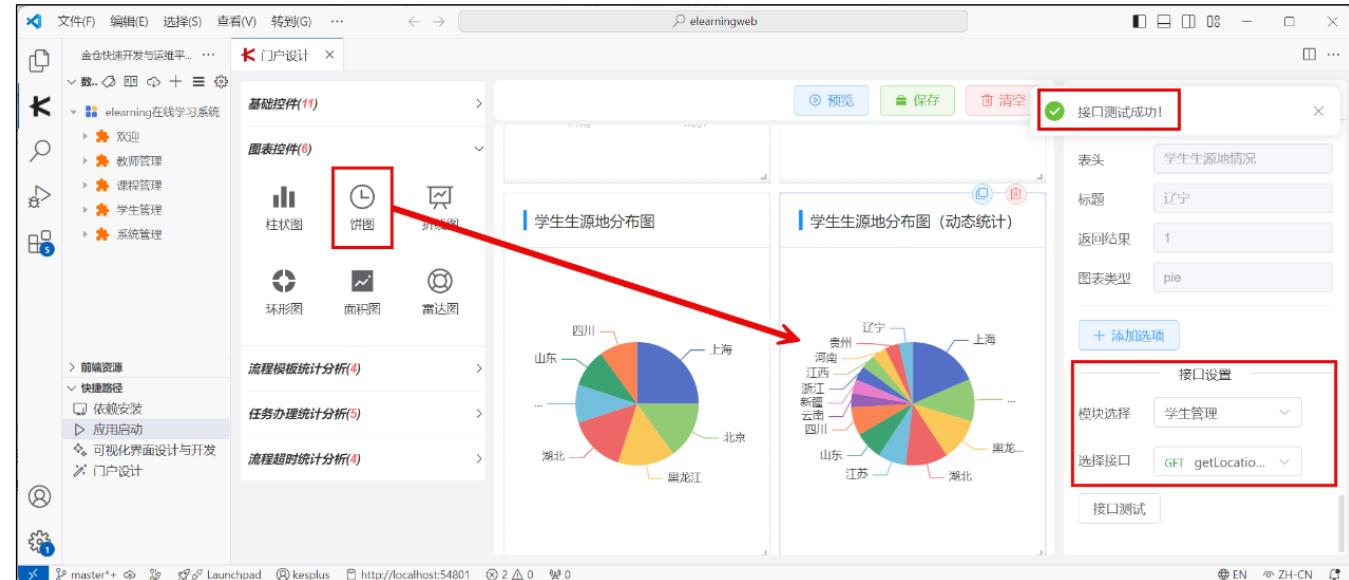
在“学生管理\数据服务\RESTful\学生信息”下，运行该接口，可看到与函数运行相同的结果。

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“查看生源地分布”，之后即可在前端页面中访问该接口。

## ③ 门户页面访问RESTful接口获取数据并展示在图表中。

在门户设计窗口拖拽一个“饼图”，在组件属性最下方的“接口设置”中，选学生管理模块，找到自定义函数getLocationStat对应的接口，随后看到消息提示“接口测试成功”，保存。

**注意：**门户页面配置的接口数据只能来自GET类型的无参数自定义函数。

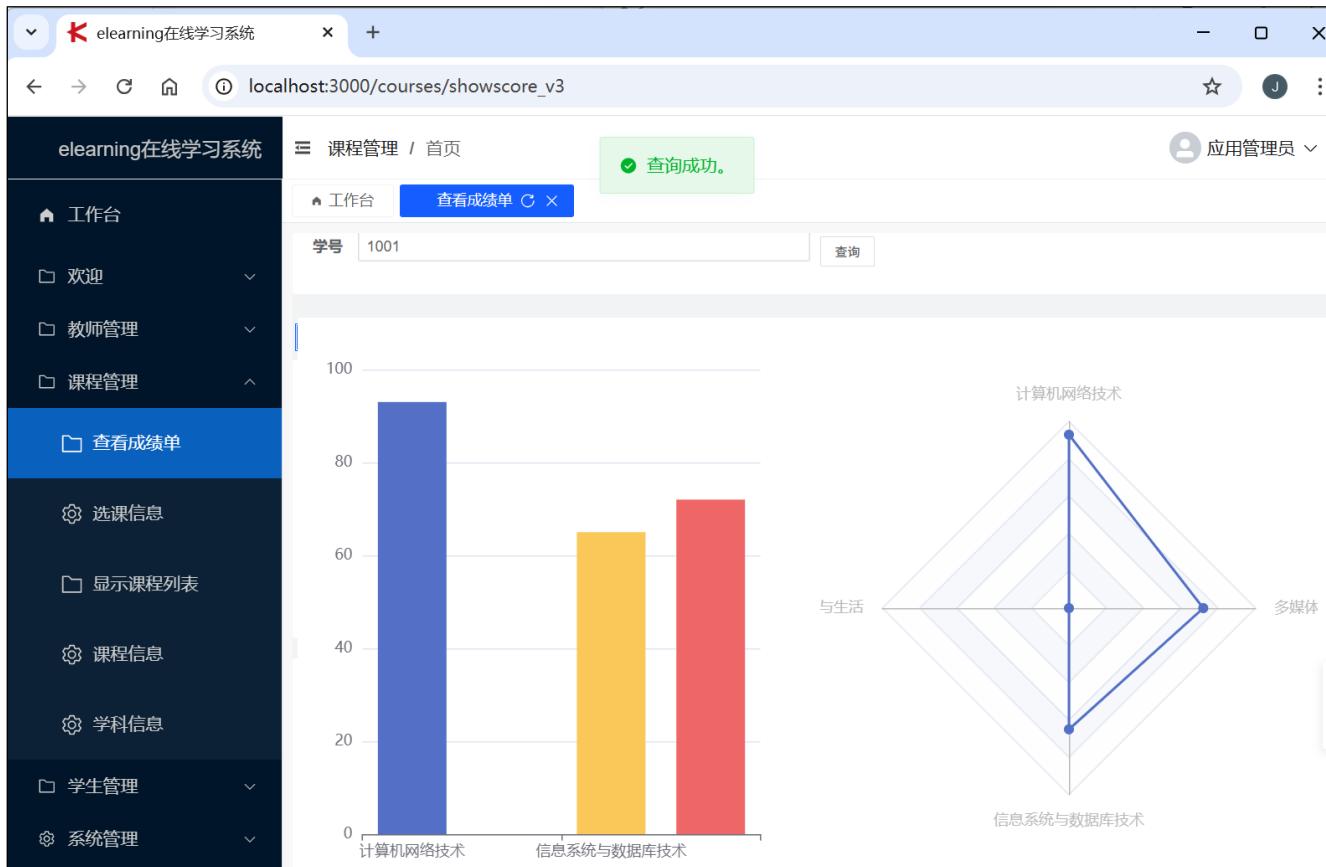




# 自定义页面图表设计

除了在门户页面添加图表，还可以在自定义页面添加图表，实现方法与门户页面设计基本类似，但是功能更强大，例如根据用户输入动态获取数据、根据触发条件渲染图表等。

**【例8.4】在查看成绩单页面按学号查询成绩分布，以柱状图和雷达图展示。**



# 自定义页面图表设计

## ① 创建自定义函数getChartDataByStudentcode及RESTful接口。设置RESTful名称

“getChartDataByStudentcode”、 URL路径“/courses/getChartDataByStudentcode”、 请求类型“GET”。

```
CREATE OR REPLACE FUNCTION "courses"."getChartDataByStudentcode" (stuinfo JSONB) RETURN JSONB AS
declare
v_input_student_code INTEGER; -- 用于存储从输入 JSON 中提取的学号
v_student_exists BOOLEAN := FALSE; -- 标志,用于判断学生是否存在
v_type TEXT; -- 用于存储图表类型, line/bar/pie/radar
vChartData jsonb; -- 用于存储绘制图表的数据
begin
v_input_student_code := (stuinfo->>'studentCode')::INTEGER; -- 从输入 JSON 中提取学生学号
v_type :=(stuinfo ->> 'type'); -- 从输入JSON中提取图表类型
-- 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;
-- 2.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
    RETURN jsonb_build_object(
        'studentCode', v_input_student_code,
        'status', 'Failure',
        'message', '未找到学号为 ' || v_input_student_code || ' 的学生。' -- ||是字符串拼接操作
    );
END IF;
```

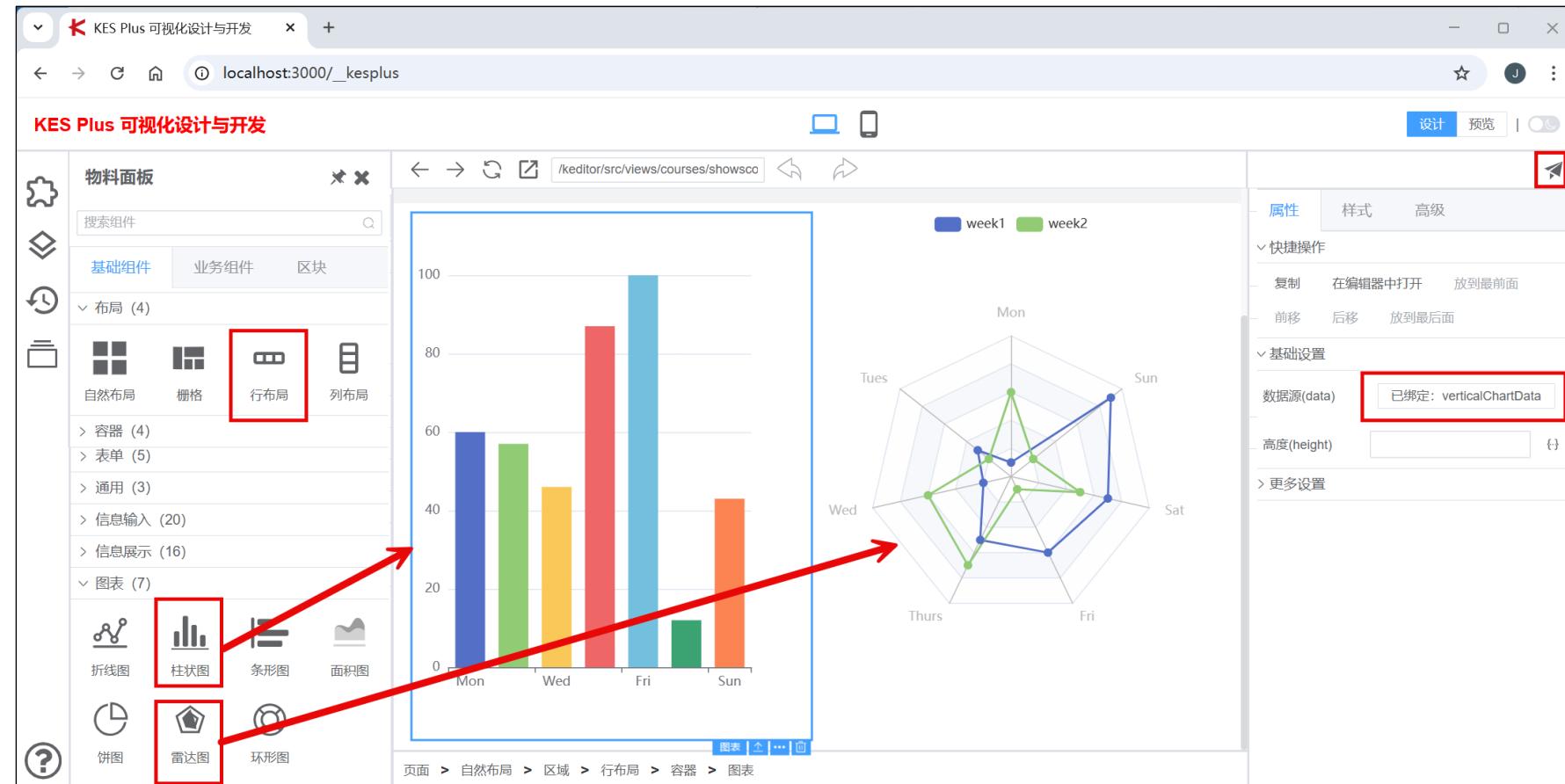
--3. 如果学生存在, 获取所有选修课程的成绩数据, 按指定的图表类型存储到vChartData

```
SELECT jsonb_agg(
    jsonb_build_object(
        'label', c2.coursename,
        'value', c1.score,
        'legend', v_input_student_code,
        'type', v_type
    )
)
INTO vChartData
FROM courses.courseenroll c1
JOIN courses.course c2 ON c1.CourseCode = c2.CourseCode
WHERE c1.StudentCode = v_input_student_code;
RETURN jsonb_build_object(
    'studentCode', v_input_student_code,
    'chartData',vChartData, -- 返回绘图所需数据
    'status', 'Success'
);
end;
```

# 自定义页面图表设计

② 创建一个新的上下布局的页面showscore\_v3.vue。打开“可视化界面设计与开发”界面，为页面添加组件。首先，按照showscore\_v1.vue类似的方式添加学号输入框和查询按钮。接着，拖拽一个“行布局”到页面下半部分，只保留两个容器，然后拖拽柱状图和雷达图组件，点击右上角‘'执行修改。最后，配置菜单项链接到该新页面。

在柱状图属性窗口可看到数据源(data)自动绑定了一个变量名为verticalChartData的示例数据，而雷达图自动绑定了一个变量名为radarChartData的示例数据以及相应的属性配置变量radarChartOptions。在后面的步骤中根据需要将示例数据替换。



# 自定义页面图表设计

---

③ 修改**showscore\_v3.vue**代码，为查询按钮的单击事件添加事件处理函数，即设置前端页面点击“查询”按钮后访问RESTful接口获取数据并展示在图表中。主要修改内容包括（代码附后）：

从后端获取图表数据的RESTful接口是“`/courses/getChartdataByStudentcode`”。**添加从后端接口获取数据绘制柱状图和雷达图的代码**，柱状图数据直接从后端获取，之后把柱状图数据verticalChartData的type更改为radar后作为雷达图数据。

为了在点击查询后获取绘图数据并渲染图表，代码中**增加一个布尔值showFigures**作为渲染条件，初始化为false，用户点击查询按钮后将值设为true，允许渲染图表。给页面上的柱状图组件和雷达图组件**添加v-if属性**，指明渲染条件。`showscore_v3.vue`的完整代码见后面页面。

④ 新建菜单并链接到页面，创建一个与菜单对应的权限点。

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 启动应用，在查看成绩单页面的输入框输入1001，点击查询按钮后看到图表。

```
<template>
<KPage>
<KSection>
<KBlock>
<KRow>
<KCell>
  <el-form ref="formRef" label-width="auto" :model="form">
    <KRow ver-align="top">
      <KCell ver-align="top">
        <el-form-item :rules="[{ required: false, message: '必填', }]" label="学号">
          <el-input v-model="studentCode" />
        </el-form-item>
      </KCell>
      <KCell>
        <KP ver-align="top"><el-button @click="getScoreByCode"><el-text>查询</el-text></el-button></KP>
      </KCell>
    </KRow>
  </el-form>
</KCell>
</KRow>
</KBlock>
<KBlock>
<KRow>
<KCell>
  <k-chart v-if="showFigures" :data="verticalChartData" />
</KCell>
<KCell>
  <k-chart v-if="showFigures" :data="radarChartData" :option="radarChartOptions" />
</KCell>
</KRow>
</KBlock>
</KSection>
</KPage>
</template>
```

showscore\_v3.vue的完整代码

添加红字内容

```
<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 verticalChartData = ref([]); // 存储柱状图数据的变量
const radarChartData = ref([]); // 存储雷达图数据的变量
const radarChartOptions = ref({ //雷达图的配置选项，初始为空
    radar: {
        indicator: [],
    },
});
const showFigures = ref(false); // 渲染条件变量，初始为false，即不渲染
// 异步函数，用于根据学号查询成绩
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;
    }
    showFigures.value = false //2. 清空渲染条件变量
    try {
        const stuinfoData = { studentCode: studentCode.value, type: 'bar' }; //构建一个包含学号和图表类型的JSON对象
        const stuinfoJsonString = JSON.stringify(stuinfoData);
        // 3. 发送 GET 请求
        const response = await request.get('/courses/getChartdataByStudentcode', {
            params: {
                stuinfo: stuinfoJsonString
            }
        });
    }
}
```

修改

showscore\_v3.vue的完整代码

添加红字内容

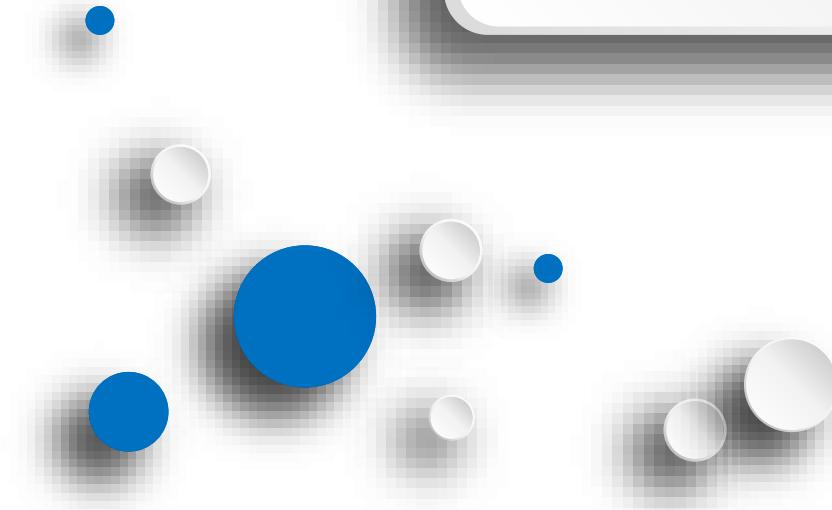
```
// 4. 处理响应
if (response && response.status == 'Success') {
    ElMessage.success('查询成功。')
    //获取绘制柱状图的数据
    verticalChartData.value = response.chartData
    //绘制雷达图的数据和柱状图的数值一样， type更改为radar
    radarChartData.value = response.chartData.map(point => {
        return {
            ...point, // 使用扩展运算符 (...) 复制原始对象的所有属性
            type: 'radar' // 覆盖type属性的值
        }
    })
    // 根据图形数据动态生成雷达图的配置
    const uniqueLabels = [...new Set(radarChartData.value.map(item => item.label))]; //从数据中提取所有唯一的label (去除重复值)
    const newIndicators = uniqueLabels.map(item => ({
        name: item, // 使用label作为每个轴的名称
        max: 100, // 假设最高分为 100
    }));
    radarChartOptions.value.radar.indicator = newIndicators
    showFigures.value = true //允许渲染图表
} 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>
```

showscore\_v3.vue的完整代码

添加红字内容

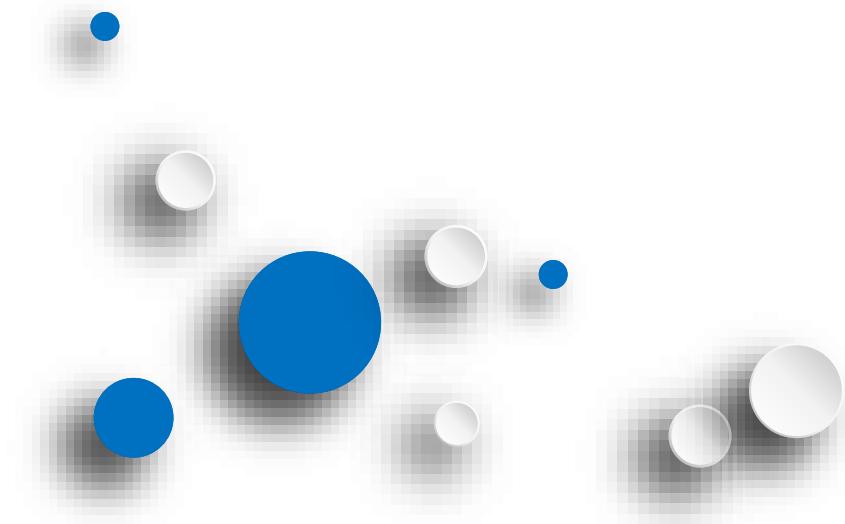
**THANK YOU!**



# 系统部署



# 系统部署





## 构建发布包

KES Plus应用构建是将一个应用项目相关的源代码、资源、数据等转换为可部署的发布包，是应用开发过程中的重要一环。

实现步骤：

①在KES Plus中右击应用项目，从快捷菜单选“查看”，展示应用管理窗口，选“构建管理”菜单，点击“新增”按钮，在构建对话框输入构建信息，点击“保存”，系统自动运行pnpm run build命令，使用Vite构建应用发布包。若构建成功，Terminal窗口显示“构建完成”，并在构建包列表新增一条记录，点击可查看构建信息。

②点击“”按钮下载发布包。它是一个zip文件，解压后可看到目录结构。

The screenshot shows the KES Plus application management interface. On the left, there is a sidebar with tabs: 基本信息, 成员信息, BUG管理, 构建管理 (Build Management), 部署管理, and 拦截器管理. The '构建管理' tab is selected and highlighted with a red circle. On the right, there is a table titled 'elearning 在线学习系统.kp' with columns: 详细 (Details), 版本号 (Version), 打包时间 (Packaging Time), 打包人 (Packer), and 操作 (Operations). A new row is being added, indicated by a green '新增' button. The first column has a '大于' (Greater Than) operator. The second column contains '1.0'. The third column contains '2025-06-27 04:06:59'. The fourth column contains 'kesplus'. The fifth column has a download icon, which is also circled in red. Below the table, a modal dialog box titled '构建' (Build) is open. It contains fields for '版本号' (Version) set to '1.0', '打包名称' (Packaging Name) set to 'elearning 在线学习系统', and '发版日志' (Release Log) set to '部署版本1'. At the bottom of the dialog are '重置' (Reset) and '保存' (Save) buttons, with '保存' also circled in red.

# 发布包结构

---

下载的发布包是一个zip文件，解压后可看到目录结构，包括前端和后端两部分。

发布包中：

- .kesplus是后端数据库相关文件目录
- dist是前端文件目录
- 还有一个“应用编码.ksp”文件。

📁 .kesplus	# 后端数据库相关文件目录
db-data	# 数据目录
app-meta	# 应用元信息
📁 dist	# 前端构建文件目录
favicon.	# 网页图标
index.html	# 索引页
logo.svg	# logo 图标
server.json	# 服务器信息
wine-outline.svg	# 大纲图片
static	# 静态资源目录



# 应用部署

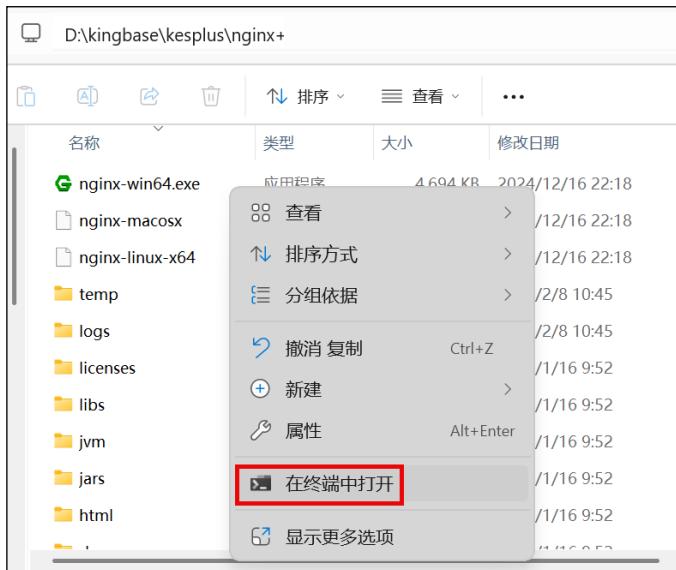
## 1. 准备部署环境

KES Plus服务器安装时会附带安装一个Nginx，用于部署前端代码。Nginx 是一个高性能的HTTP和反向代理Web服务器，同时也提供了IMAP/POP3/SMTP服务。

在部署应用前应先验证Nginx是否已正确安装。在KES Plus安装目录下找到nginx+文件夹（例如“D:\kingbase\kesplus\nginx+”），在文件夹空白处右击鼠标选“在终端中打开”，在终端中输入以下命令：

`.\nginx-win64.exe -t`

如果执行上述命令后，系统显示如右图所示的信息，说明KES Plus服务中的Nginx 已正确安装。



```
PS D:\kingbase\kesplus\nginx+> .\nginx-win64.exe -t
nginx: the configuration file D:\kingbase\kesplus\nginx+/conf/nginx.conf syntax is ok
nginx: configuration file D:\kingbase\kesplus\nginx+/conf/nginx.conf test is successful
PS D:\kingbase\kesplus\nginx+> |
```

# 应用部署

## 2. 设置KES Plus服务器

应用部署的宿主是KES Plus服务器，部署前先设置目标KES Plus服务器地址。在资源浏览框点击“”平台设置按钮，在服务器设置页面修改服务器名称和地址，点击“保存”（图9.24）。默认服务器地址为本机“localhost:54801”。可根据实际服务器地址修改。



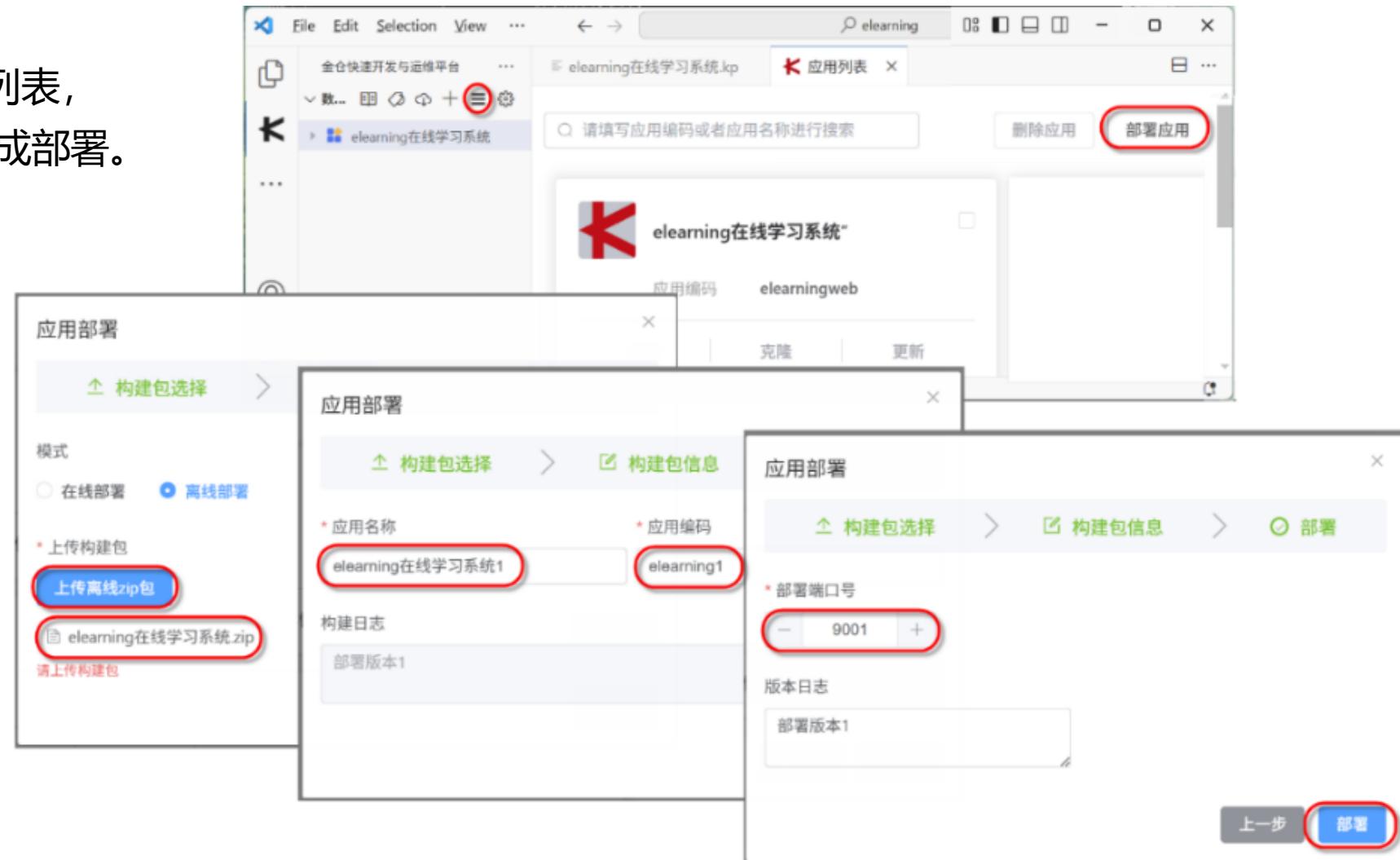
# 应用部署

## 3. 部署应用

在资源导航树点击“”应用列表，  
点击“部署应用”按钮按图步骤完成部署。

- ① 在“构建包选择”步骤点击“上传离线Zip包”按钮上传发布包elearning在线学习系统.zip，选离线部署模式；
- ② 下一步在“构建包信息”对话框，修改部署的应用程序名称、应用程序编码。名称和编码应不同于已创建的应用；
- ③ 下一步在“部署”对话框，修改端口号（应该是未被使用的）。点击“部署”，应用将被部署到KES Plus服务器上。

可通过网址localhost:9001访问即可启动系统，进入登录页面。



若要删除某应用，首先选中它，然后点击页面右上角“删除应用”按钮删除

**THANK YOU!**

