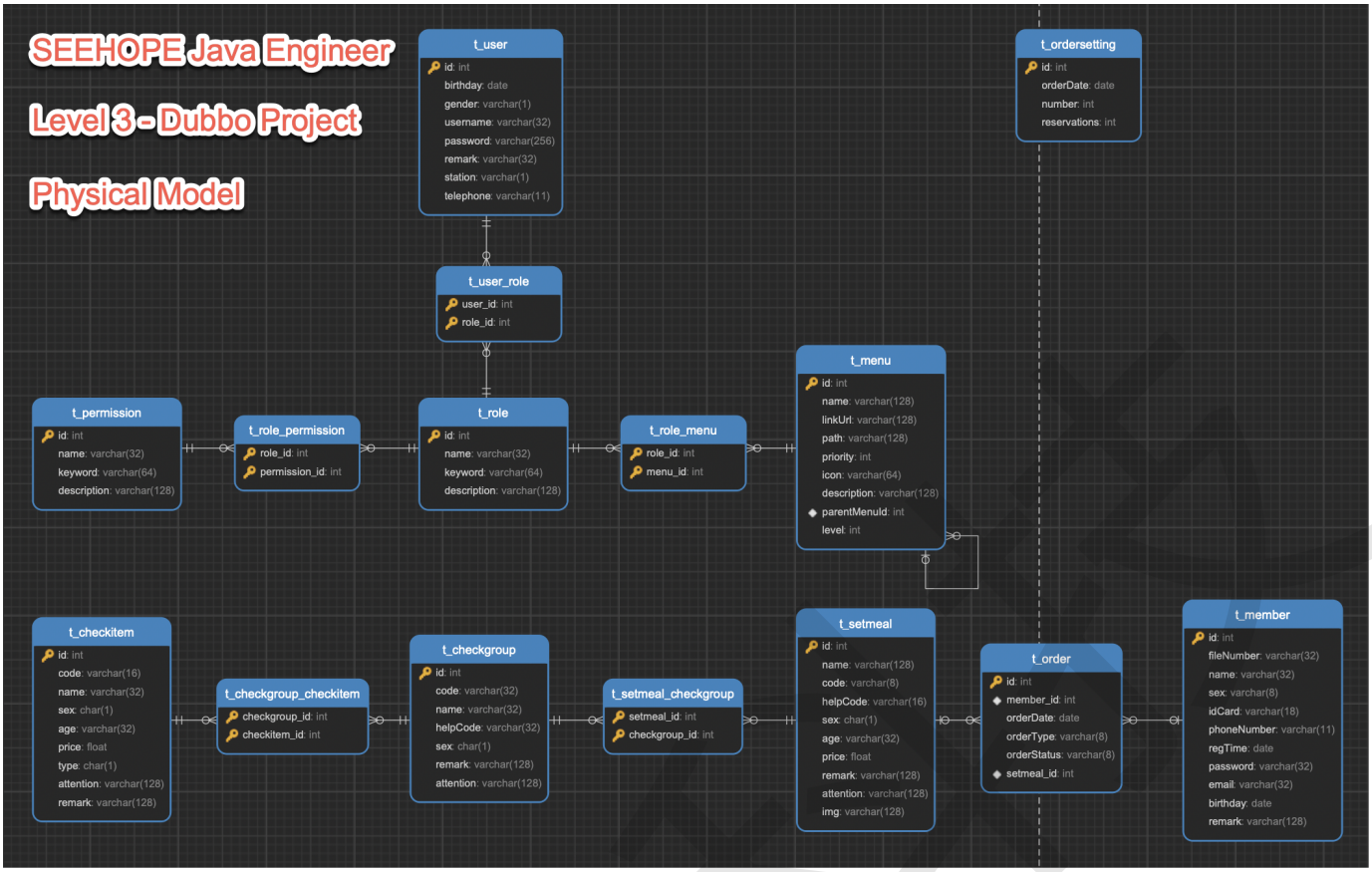
Day17

数据库模型设计与分析



导入数据

导入 sql 脚本有多种方式，可以使用 Navicat 等图形界面工具导入，也可以使用命令行：

```
mysql> use seehope_health
Database changed
mysql> source /Users/szw/level3-code/seehope-health-parent/db/seehope-health.sql
Query OK, 0 rows affected (0.00 sec)
```

完善模型层

项目涉及的实体类统一放到到 seehope-health-common 工程中。

根据数据库模型需要创建如下实体类：

- Order.java
- Menu.java
- Role.java
- User.java
- Permission.java
- CheckItem.java
- OrderSetting.java

- CheckGroup.java
- Setmeal.java
- Member.java

阿里巴巴关于领域模型命名的规约：

1. 数据对象：xxxDO，xxx 即为数据表名。
2. 数据传输对象：xxxDTO，xxx 为业务领域相关的名称。
3. 展示对象：xxxVO，xxx 一般为网页名称。
4. POJO 是 DO/DTO/BO/VO 的统称，禁止命名成 xxxPOJO。

今天任务

- 根据物理模型图自行创建好实体类，统一放到 seehope-health-common 工程中。
- 项目上传到 git，周一 review。

项目所需的公共资源

项目开发过程中一般会提供一些公共资源，供多个模块或者系统来使用。

我们把当前需要的一些公共资源放到 seehope-health-common 工程中。

这样，common 工程除了存放 pojo 的包外，再添加两个包：constant，entity

constant：

- 返回消息常量类 – MessageConstant.java

entity：

- 返回结果类 – Result.java
- 封装查询条件的类 – QueryPageBean.java 和 PageResult.java

后续开发过程中，如果后还有哪些资源可以提取为多个模块使用则也加入到本工程中，成为公共资源。

Vue.js

Vue.js 的介绍

官网：https://cn.vuejs.org/

- Vue是一套构建用户界面的渐进式前端框架。
- 只关注视图层，并且非常容易学习，还可以很方便的与其它库或已有项目整合。
- 通过尽可能简单的 API 来实现响应数据的绑定和组合的视图组件。
- 特点
    - 易用：在有HTMLCSSJavaScript的基础上，快速上手。
    - 灵活：简单小巧的核心，渐进式技术栈，足以应付任何规模的应用。
    - 性能：20kbmin+gzip运行大小、超快虚拟DOM、最省心的优化。

Vue的快速入门

开发步骤

1. 下载和引入vue.js文件。
    a. 引入本地文件
    b. 通过cdn
2. 编写入门程序。
    a. 视图：负责页面渲染，主要由 HTML + CSS 构成。
    b. 脚本：负责业务数据模型（Model）以及数据的处理逻辑。

案例

HTML 复制代码

```html
<!DOCTYPE html>
<html lang="zh-CN">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>快速入门</title>
</head>
<body>
    <!-- 视图 -->
    <div id="div">
        {{msg}}
    </div>
</body>
<script src="js/vue.js"></script>
<script>
    // 脚本
    new Vue({
        el:"#div",
        data:{
            msg:"Hello Vue"
        }
    });
</script>
</html>
```

入门详解

- Vue 核心对象：每一个 Vue 程序都是从一个 Vue 核心对象开始的。

JavaScript 复制代码

```javascript
let vm = new Vue({
  选项列表;
});
```

- 选项列表
  - el 选项
    - 用于接收获取到页面中的元素。(根据常用选择器获取)。
  - data 选项
    - 用于保存当前 Vue 对象中的数据。在视图中声明的变量需要在此处赋值。
  - methods 选项
    - 用于定义方法。方法可以直接通过对象名调用，this 代表当前 Vue 对象。
- 数据绑定
  - 在视图部分获取脚本部分的数据。

- {{变量名}}

完善入门案例

HTML 复制代码

```html
<!DOCTYPE html>
<html lang="zh-CN">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>入门案例升级</title>
</head>
<body>
    <!-- 视图 -->
    <div id="div">
        <div>姓名：{{name}}</div>
        <div>年龄：{{age}}</div>
        <button onclick="hi()">打招呼</button>
        <button onclick="update()">修改年龄</button>
    </div>
</body>
<script src="js/vue.js"></script>
<script>
    // 脚本
    let vm = new Vue({
        el:"#div",
        data:{
            name: "张三",
            age: 21
        },
        methods:{
            sayAge(){
                alert(this.name + "现在 " + this.age + " 岁了！");
            }
        }
    });

    //定义打招呼方法
    function hi(){
        vm.sayAge();
    }

    //定义修改班级
    function update(){
        vm.age++;
    }
</script>
</html>
```

入门总结

- Vue是一套构建用户界面的渐进式前端框架。

- Vue的程序包含视图和脚本两个核心部分。
- 脚本部分
  - Vue 核心对象。
  - 选项列表
    - el：接收获取的元素。
    - data：保存数据。
    - methods：定义方法。
- 视图部分
  - 数据绑定语法：{{变量名}}

Vue 常用指令

指令介绍

- 指令：是带有 v- 前缀的特殊属性，不同指令具有不同含义。
  - 例如 v-html，v-if，v-for。
- 使用指令时，通常编写在标签的属性上，值可以使用 JS 的表达式。
- 常用指令

指令	作用
v-html	把文本解析成 HTML 代码
v-bind	为 HTML 标签绑定属性值
v-if	条件渲染某元素，返回 true 时渲染，否则不渲染。
v-else	
v-else-if	
v-show	根据条件展示某元素，切换的是 display 属性的值
v-for	列表渲染，遍历容器的元素或者对象的属性
v-on	为 HTML 元素绑定事件
v-model	在表单元素上创建双向数据绑定

2.2、文本插值

v-html：把文本解析为 HTML 代码。

HTML 复制代码

```html
<!DOCTYPE html>
<html lang="zh-CN">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>文本插值</title>
  </head>
  <body>
    <div id="div">
      <div>{{msg}}</div>
      <div v-html="msg"></div>
    </div>
  </body>
  <script src="js/vue.js"></script>
  <script>
    new Vue({
      el:"#div",
      data:{
        msg:"<b>Hello Vue</b>"
      }
    });
  </script>
</html>
```

2.3、绑定属性

v-bind：为 HTML 标签绑定属性值。

HTML 复制代码

```html
<!DOCTYPE html>
<html lang="zh-CN">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>绑定属性</title>
    <style>
      .my {
        border: 1px solid red;
      }
    </style>
  </head>
  <body>
    <div id="div">
      <a v-bind:href="url">百度一下</a>
      <br>
      <a :href="url">百度一下</a>
      <br>
      <div :class="cls">我是div</div>
    </div>
  </body>
  <script src="js/vue.js"></script>
  <script>
    new Vue({
      el:"#div",
      data:{
        url:"https://www.baidu.com",
        cls:"my"
      }
    });
  </script>
</html>
```

2.4、条件渲染

- v-if：条件性的渲染某元素，判定为真时渲染,否则不渲染。
- v-else：条件性的渲染。
- v-else-if：条件性的渲染。
- v-show：根据条件展示某元素，区别在于切换的是display属性的值。

HTML 复制代码

```html
<!DOCTYPE html>
<html lang="zh-CN">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>条件渲染</title>
</head>
<body>
    <div id="div">
        <!-- 判断num的值，对3取余  余数为0显示div1  余数为1显示div2  余数为2显示div3 -->
        <div v-if="num % 3 == 0">div1</div>
        <div v-else-if="num % 3 == 1">div2</div>
        <div v-else="num % 3 == 2">div3</div>

        <div v-show="flag">div4</div>
    </div>
</body>
<script src="js/vue.js"></script>
<script>
    new Vue({
        el:"#div",
        data:{
            num: 1,
            flag: false
        }
    });
</script>
</html>
```

2.5、列表渲染

v-for：列表渲染，遍历容器的元素或者对象的属性。

HTML 复制代码

```html
<!DOCTYPE html>
<html lang="zh-CN">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>列表渲染</title>
</head>
<body>
    <div id="div">
        <ul>
            <li v-for="name in names">
                {{name}}
            </li>
            <li v-for="value in student">
                {{value}}
            </li>
        </ul>
    </div>
</body>
<script src="js/vue.js"></script>
<script>
    new Vue({
        el:"#div",
        data:{
            names:["张三","李四","王五"],
            student:{
                name:"张三",
                age:23
            }
        }
    });
</script>
</html>
```

2.6、事件绑定

v-on：为 HTML 标签绑定事件。

```HTML
<!DOCTYPE html>
<html lang="zh-CN">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>事件绑定</title>
</head>
<body>
    <div id="div">
        <div>{{name}}</div>
        <button v-on:click="change()">改变div的内容</button>
        <button v-on:dblclick="change()">改变div的内容</button>

        <button @click="change()">改变div的内容</button>
    </div>
</body>
<script src="js/vue.js"></script>
<script>
    new Vue({
        el:"#div",
        data:{
            name:"张三"
        },
        methods:{
            change(){
                this.name = "李四"
            }
        }
    });
</script>
</html>
```

2.7、表单绑定

- **表单绑定**
  - v-model：在表单元素上创建双向数据绑定。
- **双向数据绑定**
  - 更新data数据，页面中的数据也会更新。更新页面数据，data数据也会更新。
- **MVVM模型(ModelViewViewModel)：是MVC模式的改进版**
  - 在前端页面中，JS 对象表示 Model，页面表示 View，两者做到了最大限度的分离。
  - 将 Model 和 View 关联起来的就是 ViewModel，它是桥梁。
  - ViewModel 负责把 Model 的数据同步到 View 显示出来，还负责把 View 修改的数据同步回 Model。

HTML 复制代码

```html
<!DOCTYPE html>
<html lang="zh-CN">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>表单绑定</title>
</head>
<body>
    <div id="div">
        <form autocomplete="off">
            姓名: <input type="text" name="username" v-model="username">
            <br>
            年龄: <input type="number" name="age" v-model="age">
        </form>
    </div>
</body>
<script src="js/vue.js"></script>
<script>
    new Vue({
        el:"#div",
        data:{
            username:"张三",
            age:23
        }
    });
</script>
</html>
```

Vue.js 基础总结

- **指令：**
  - **是带有v-前缀的特殊属性，不同指令具有不同含义。**
- **文本插值**
  - v-html：把文本解析为HTML代码。
- **绑定属性**
  - v-bind：为HTML标签绑定属性值。
- **条件渲染**
  - v-if：条件性的渲染某元素，判定为真时渲染,否则不渲染。v-else：条件性的渲染。v-else-if：条件性的渲染。
  - v-show：根据条件展示某元素，区别在于切换的是display属性的值。
- **列表渲染**
  - v-for：列表渲染，遍历容器的元素或者对象的属性。

- **事件绑定**
  - v-on：为HTML标签绑定事件。
- **表单绑定**
  - v-model：在表单元素上创建双向数据绑定。

Element UI

简介

- Element：网站快速成型工具。是饿了么公司前端开发团队提供的一套基于Vue的网站组件库。
- 使用Element前提必须要有Vue。
- 组件：组成网页的部件，例如超链接、按钮、图片、表格等等
- 官网：https://element.eleme.cn/#/zh-CN

Element 入门案例

开发步骤

1. 下载 Element 核心库。
2. 引入 Element 样式文件。
3. 引入 Vue 核心 js 文件。
4. 引入 Element 核心 js 文件。
5. 编写按钮标签。
6. 通过 Vue 核心对象加载元素。

案例代码

HTML 复制代码

```html
<!DOCTYPE html>
<html lang="en">

  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>入门案例</title>
    <!-- 引入 Vue.js -->
    <script src="https://cdn.jsdelivr.net/npm/vue@2/dist/vue.js"></script>
    <!-- 引入样式 -->
    <link rel="stylesheet" href="https://unpkg.com/element-ui/lib/theme-chalk/index.css">
    <!-- 引入组件库 -->
    <script src="https://unpkg.com/element-ui/lib/index.js"></script>
  </head>

  <body>
    <button>我是按钮</button>
    <br><br>
    <div id="div">
      <el-row>
        <el-button>默认按钮</el-button>
        <el-button type="primary">主要按钮</el-button>
        <el-button type="success">成功按钮</el-button>
        <el-button type="info">信息按钮</el-button>
        <el-button type="warning">警告按钮</el-button>
        <el-button type="danger">危险按钮</el-button>
      </el-row>
      <br>
      <el-row>
        <el-button plain>朴素按钮</el-button>
        <el-button type="primary" plain>主要按钮</el-button>
        <el-button type="success" plain>成功按钮</el-button>
        <el-button type="info" plain>信息按钮</el-button>
        <el-button type="warning" plain>警告按钮</el-button>
        <el-button type="danger" plain>危险按钮</el-button>
      </el-row>
      <br>
      <el-row>
        <el-button round>圆角按钮</el-button>
        <el-button type="primary" round>主要按钮</el-button>
        <el-button type="success" round>成功按钮</el-button>
        <el-button type="info" round>信息按钮</el-button>
        <el-button type="warning" round>警告按钮</el-button>
        <el-button type="danger" round>危险按钮</el-button>
      </el-row>
      <br>
      <el-row>
```

```
        <el-button icon="el-icon-search" circle></el-button>
        <el-button type="primary" icon="el-icon-edit" circle></el-button>
        <el-button type="success" icon="el-icon-check" circle></el-button>
        <el-button type="info" icon="el-icon-message" circle></el-button>
        <el-button type="warning" icon="el-icon-star-off" circle></el-button>
        <el-button type="danger" icon="el-icon-delete" circle></el-button>
      </el-row>
    </div>
  </body>
  <script>
    new Vue({
      el: "#div"
    });
  </script>

</html>
```

Element 基础布局

将页面分成最多 24 个部分，自由切分。

代码例子：

HTML | 复制代码

```html
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>基础布局</title>
  <!-- 引入 Vue.js -->
  <script src="https://cdn.jsdelivr.net/npm/vue@2/dist/vue.js"></script>
  <!-- 引入样式 -->
  <link rel="stylesheet" href="https://unpkg.com/element-ui/lib/theme-chalk/index.css">
  <!-- 引入组件库 -->
  <script src="https://unpkg.com/element-ui/lib/index.js"></script>

  <style>
    .el-row {
      /* 行距为20px */
      margin-bottom: 20px;
    }

    .bg-purple-dark {
      background: red;
    }

    .bg-purple {
      background: blue;
    }

    .bg-purple-light {
      background: green;
    }

    .grid-content {
      /* 边框圆润度 */
      border-radius: 4px;
      /* 行高为36px */
      min-height: 36px;
    }
  </style>
</head>

<body>
  <div id="div">
    <el-row>
      <el-col :span="24">
        <div class="grid-content bg-purple-dark"></div>
      </el-col>
```

```
    </el-row>
    <el-row>
      <el-col :span="12">
        <div class="grid-content bg-purple"></div>
      </el-col>
      <el-col :span="12">
        <div class="grid-content bg-purple-light"></div>
      </el-col>
    </el-row>
    <el-row>
      <el-col :span="8">
        <div class="grid-content bg-purple"></div>
      </el-col>
      <el-col :span="8">
        <div class="grid-content bg-purple-light"></div>
      </el-col>
      <el-col :span="8">
        <div class="grid-content bg-purple"></div>
      </el-col>
    </el-row>
    <el-row>
      <el-col :span="6">
        <div class="grid-content bg-purple"></div>
      </el-col>
      <el-col :span="6">
        <div class="grid-content bg-purple-light"></div>
      </el-col>
      <el-col :span="6">
        <div class="grid-content bg-purple"></div>
      </el-col>
      <el-col :span="6">
        <div class="grid-content bg-purple-light"></div>
      </el-col>
    </el-row>
    <el-row>
      <el-col :span="4">
        <div class="grid-content bg-purple"></div>
      </el-col>
      <el-col :span="4">
        <div class="grid-content bg-purple-light"></div>
      </el-col>
      <el-col :span="4">
        <div class="grid-content bg-purple"></div>
      </el-col>
      <el-col :span="4">
        <div class="grid-content bg-purple-light"></div>
      </el-col>
      <el-col :span="4">
        <div class="grid-content bg-purple"></div>
      </el-col>
```

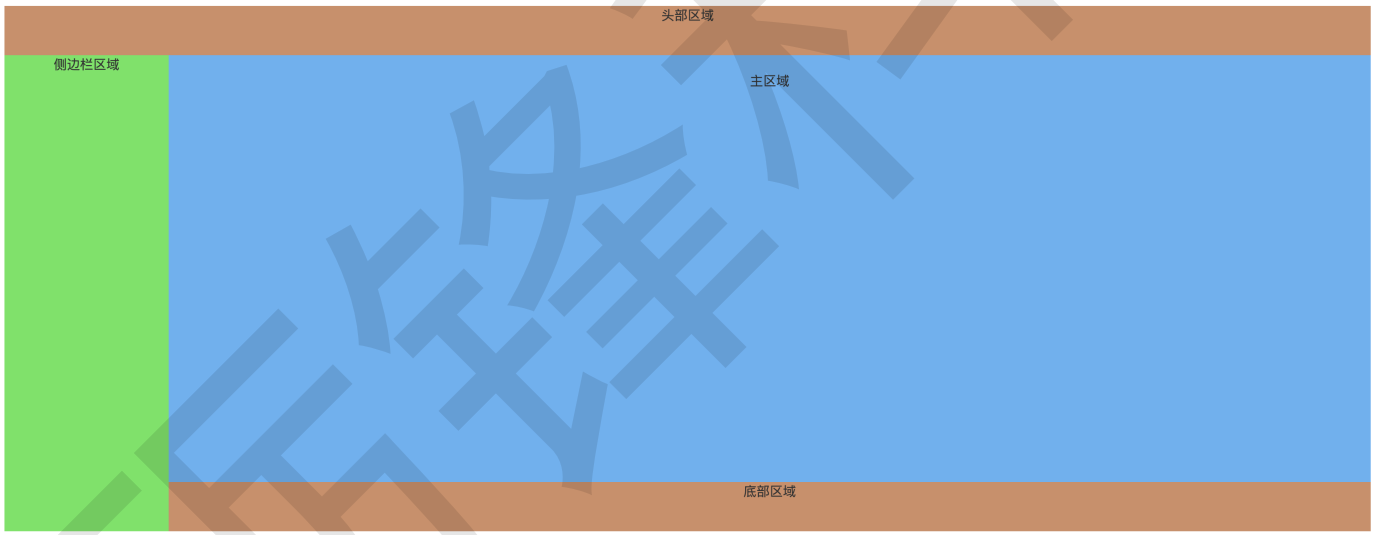
```
      <el-col :span="4">
        <div class="grid-content bg-purple-light"></div>
      </el-col>
    </el-row>
  </div>
</body>
<script>
  new Vue({
    el: "#div"
  });
</script>

</html>
```

Element 容器布局

将页面分成头部区域、侧边栏区域、主区域、底部区域。

代码例子：



HTML 复制代码

```html
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>基础布局</title>
    <!-- 引入 Vue.js -->
    <script src="https://cdn.jsdelivr.net/npm/vue@2/dist/vue.js"></script>
    <!-- 引入样式 -->
    <link rel="stylesheet" href="https://unpkg.com/element-ui/lib/theme-chalk/index.css">
    <!-- 引入组件库 -->
    <script src="https://unpkg.com/element-ui/lib/index.js"></script>

    <style>
        .el-header,
        .el-footer {
            background-color: #B3C0D1;
            color: #333;
            text-align: center;
            line-height: 60px;
        }

        .el-aside {
            background-color: #D3DCE6;
            color: #333;
            text-align: center;
            line-height: 200px;
        }

        .el-main {
            background-color: #E9EEF3;
            color: #333;
            text-align: center;
            line-height: 160px;
        }

        body>.el-container {
            margin-bottom: 40px;
        }

        .el-container:nth-child(5) .el-aside,
        .el-container:nth-child(6) .el-aside {
            line-height: 260px;
        }

        .el-container:nth-child(7) .el-aside {
```

```
            line-height: 320px;
        }
    </style>
</head>

<body>
    <div id="div">
        <el-header>Header</el-header>
        <el-container>
            <el-aside width="200px">Aside</el-aside>
            <el-container>
                <el-main>Main</el-main>
                <el-footer>Footer</el-footer>
            </el-container>
        </el-container>
    </div>
</body>
<script>
    new Vue({
        el: "#div"
    });
</script>

</html>
```

Element 表单组件

由输入框、下拉列表、单选框、多选框等控件组成，用以收集、校验、提交数据。

代码例子：

HTML 复制代码

```html
<!DOCTYPE html>
<html lang="en">

  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>表单组件</title>
    <!-- 引入 Vue.js -->
    <script src="https://cdn.jsdelivr.net/npm/vue@2/dist/vue.js"></script>
    <!-- 引入样式 -->
    <link rel="stylesheet" href="https://unpkg.com/element-ui/lib/theme-chalk/index.css">
    <!-- 引入组件库 -->
    <script src="https://unpkg.com/element-ui/lib/index.js"></script>
  </head>

  <body>
    <div id="div">
      <el-form :model="ruleForm" :rules="rules" ref="ruleForm" label-width="100px" class="demo-ruleForm">
        <el-form-item label="活动名称" prop="name">
          <el-input v-model="ruleForm.name"></el-input>
        </el-form-item>
        <el-form-item label="活动区域" prop="region">
          <el-select v-model="ruleForm.region" placeholder="请选择活动区域">
            <el-option label="区域一" value="shanghai"></el-option>
            <el-option label="区域二" value="beijing"></el-option>
          </el-select>
        </el-form-item>
        <el-form-item label="活动时间" required>
          <el-col :span="11">
            <el-form-item prop="date1">
              <el-date-picker type="date" placeholder="选择日期" v-model="ruleForm.date1" style="width: 100%;">
              </el-date-picker>
            </el-form-item>
          </el-col>
          <el-col class="line" :span="2">-</el-col>
          <el-col :span="11">
            <el-form-item prop="date2">
              <el-time-picker placeholder="选择时间" v-model="ruleForm.date2" style="width: 100%;"></el-time-picker>
            </el-form-item>
          </el-col>
        </el-form-item>
        <el-form-item label="即时配送" prop="delivery">
          <el-switch v-model="ruleForm.delivery"></el-switch>
        </el-form-item>
```

```
        <el-form-item label="活动性质" prop="type">
          <el-checkbox-group v-model="ruleForm.type">
            <el-checkbox label="美食/餐厅线上活动" name="type"></el-checkbox>
            <el-checkbox label="地推活动" name="type"></el-checkbox>
            <el-checkbox label="线下主题活动" name="type"></el-checkbox>
            <el-checkbox label="单纯品牌曝光" name="type"></el-checkbox>
          </el-checkbox-group>
        </el-form-item>
        <el-form-item label="特殊资源" prop="resource">
          <el-radio-group v-model="ruleForm.resource">
            <el-radio label="线上品牌商赞助"></el-radio>
            <el-radio label="线下场地免费"></el-radio>
          </el-radio-group>
        </el-form-item>
        <el-form-item label="活动形式" prop="desc">
          <el-input type="textarea" v-model="ruleForm.desc"></el-input>
        </el-form-item>
        <el-form-item>
          <el-button type="primary" @click="submitForm('ruleForm')">立即创建
</el-button>
          <el-button @click="resetForm('ruleForm')">重置</el-button>
        </el-form-item>
      </el-form>
    </div>
  </body>
  <script>
    new Vue({
      el: "#div",
      data: {
        ruleForm: {
          name: '',
          region: '',
          date1: '',
          date2: '',
          delivery: false,
          type: [],
          resource: '',
          desc: ''
        },
        rules: {
          name: [
            { required: true, message: '请输入活动名称', trigger: 'blur' },
            { min: 3, max: 5, message: '长度在 3 到 5 个字符', trigger: 'blur' }
          ],
          region: [
            { required: true, message: '请选择活动区域', trigger: 'change' }
          ],
          date1: [
            { type: 'date', required: true, message: '请选择日期', trigger:
'change' }
```

```
          ],
          date2: [
            { type: 'date', required: true, message: '请选择时间', trigger: 'change' }
          ],
          type: [
            { type: 'array', required: true, message: '请至少选择一个活动性质', trigger: 'change' }
          ],
          resource: [
            { required: true, message: '请选择活动资源', trigger: 'change' }
          ],
          desc: [
            { required: true, message: '请填写活动形式', trigger: 'blur' }
          ]
        }
      },
      methods: {
        submitForm(formName) {
          this.$refs[formName].validate((valid) => {
            if (valid) {
              alert('submit!');
            } else {
              console.log('error submit!!');
              return false;
            }
          });
        },
        resetForm(formName) {
          this.$refs[formName].resetFields();
        }
      }
    });
  </script>

</html>
```

Element 表格组件

用于展示多条结构类似的数据，可对数据进行编辑、删除或其他自定义操作。

代码例子

HTML 复制代码

```html
<!DOCTYPE html>
<html lang="en">

  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>表格组件</title>
    <!-- 引入 Vue.js -->
    <script src="https://cdn.jsdelivr.net/npm/vue@2/dist/vue.js"></script>
    <!-- 引入样式 -->
    <link rel="stylesheet" href="https://unpkg.com/element-ui/lib/theme-chalk/index.css">
    <!-- 引入组件库 -->
    <script src="https://unpkg.com/element-ui/lib/index.js"></script>
  </head>

  <body>
    <div id="div">
      <template>
        <el-table :data="tableData" style="width: 100%">
          <el-table-column prop="date" label="日期" width="180">
          </el-table-column>
          <el-table-column prop="name" label="姓名" width="180">
          </el-table-column>
          <el-table-column prop="address" label="地址">
          </el-table-column>

          <el-table-column label="操作" width="180">
            <el-button type="warning">编辑</el-button>
            <el-button type="danger">删除</el-button>
          </el-table-column>
        </el-table>
      </template>
    </div>
  </body>
  <script>
    new Vue({
      el: "#div",
      data: {
        tableData: [{
          date: '2021-05-02',
          name: '王一',
          address: '广州市天河区科韵路 1111 号'
        }, {
          date: '2021-05-04',
          name: '王二',
          address: '广州市天河区科韵路 2222 号'
        }, {
```

```
            date: '2021-05-01',
            name: '王三',
            address: '广州市天河区科韵路 2222 号'
          }, {
            date: '2021-05-03',
            name: '王四',
            address: '广州市天河区科韵路 2222 号'
          }]
        }
      });
    </script>

</html>
```

Element 顶部导航栏组件

代码例子：

HTML 复制代码

```html
<!DOCTYPE html>
<html lang="zh-CN">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>顶部导航栏</title>
  <!-- 引入 Vue.js -->
  <script src="https://cdn.jsdelivr.net/npm/vue@2/dist/vue.js"></script>
  <!-- 引入样式 -->
  <link rel="stylesheet" href="https://unpkg.com/element-ui/lib/theme-chalk/index.css">
  <!-- 引入组件库 -->
  <script src="https://unpkg.com/element-ui/lib/index.js"></script>
</head>

<body>
  <div id="div">
    <el-menu :default-active="activeIndex2" class="el-menu-demo" mode="horizontal" @select="handleSelect"
      background-color="#545c64" text-color="#fff" active-text-color="#ffd04b">
      <el-menu-item index="1">首页</el-menu-item>
      <el-submenu index="2">
        <template slot="title">我的工作台</template>
        <el-menu-item index="2-1">选项1</el-menu-item>
        <el-menu-item index="2-2">选项2</el-menu-item>
        <el-menu-item index="2-3">选项3</el-menu-item>
        <el-submenu index="2-4">
          <template slot="title">选项4</template>
          <el-menu-item index="2-4-1">选项1</el-menu-item>
          <el-menu-item index="2-4-2">选项2</el-menu-item>
          <el-menu-item index="2-4-3">选项3</el-menu-item>
        </el-submenu>
      </el-submenu>
      <el-menu-item index="3" disabled>消息中心</el-menu-item>
      <el-menu-item index="4"><a href="https://www.ele.me" target="_blank">订单管理</a></el-menu-item>
    </el-menu>
  </div>
</body>
<script>
  new Vue({
    el: "#div"
  });
</script>

</html>
```

Element 侧边导航栏组件

HTML 复制代码

```html
<!DOCTYPE html>
<html lang="zh-CN">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>侧边导航栏</title>
  <!-- 引入 Vue.js -->
  <script src="https://cdn.jsdelivr.net/npm/vue@2/dist/vue.js"></script>
  <!-- 引入样式 -->
  <link rel="stylesheet" href="https://unpkg.com/element-ui/lib/theme-chalk/index.css">
  <!-- 引入组件库 -->
  <script src="https://unpkg.com/element-ui/lib/index.js"></script>
</head>

<body>
  <div id="div">
    <el-col :span="6">
      <el-menu default-active="2" class="el-menu-vertical-demo" @open="handleOpen" @close="handleClose"
        background-color="#545c64" text-color="#fff" active-text-color="#ffd04b">
        <el-submenu index="1">
          <template slot="title">
            <i class="el-icon-location"></i>
            <span>导航一</span>
          </template>
          <el-menu-item-group>
            <template slot="title">分组一</template>
            <el-menu-item index="1-1">选项1</el-menu-item>
            <el-menu-item index="1-2">选项2</el-menu-item>
          </el-menu-item-group>
          <el-menu-item-group title="分组2">
            <el-menu-item index="1-3">选项3</el-menu-item>
          </el-menu-item-group>
          <el-submenu index="1-4">
            <template slot="title">选项4</template>
            <el-menu-item index="1-4-1">选项1</el-menu-item>
          </el-submenu>
        </el-submenu>
        <el-menu-item index="2">
          <i class="el-icon-menu"></i>
          <span slot="title">导航二</span>
        </el-menu-item>
        <el-menu-item index="3" disabled>
          <i class="el-icon-document"></i>
          <span slot="title">导航三</span>
```

```
          </el-menu-item>
          <el-menu-item index="4">
            <i class="el-icon-setting"></i>
            <span slot="title">导航四</span>
          </el-menu-item>
        </el-menu>
      </el-col>
    </div>
  </body>
  <script>
    new Vue({
      el: "#div"
    });
  </script>

  </html>
```

ElementUI 总结

- Element：网站快速成型工具。是一套为开发者、设计师、产品经理准备的基于Vue的桌面端组件库。
- 使用Element前提必须要有Vue。
- 使用步骤
  a. 下载Element核心库。
  b. 引入Element样式文件。
  c. 引入Vue核心js文件。
  d. 引入Element核心js文件。
  e. 借助常用组件编写网页。
- 常用组件
  - 网页基本组成部分，布局、按钮、表格、表单等等
  - 常用组件不需要记住，只需要在 Element 官网中复制使用即可。

ElementUI 综合案例

HTML 复制代码

```
<!DOCTYPE html>
<html lang="zh-CN">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>学生列表</title>
  <!-- 引入 Vue.js -->
  <script src="https://cdn.jsdelivr.net/npm/vue@2/dist/vue.js"></script>
  <!-- 引入样式 -->
  <link rel="stylesheet" href="https://unpkg.com/element-ui/lib/theme-chalk/index.css">
  <!-- 引入组件库 -->
  <script src="https://unpkg.com/element-ui/lib/index.js"></script>
  <style>
    .el-header {
      background-color: #545c64;
    }

    .header-img {
      width: 100px;
      margin-top: 20px;
    }
  </style>
</head>

<body>
  <div id="div">
    <el-container>
      <!-- 头部 -->
      <el-header class="el-header">
        <el-container>
          <div>
            <el-image src="img/export.png" class="header-img"></el-image>
          </div>
          <el-menu :default-active="activeIndex2" mode="horizontal" @select="handleSelect" background-color="#545c64"
            text-color="white" active-text-color="#ffd04b" style="margin-left: auto;">
            <el-menu-item index="1">处理中心</el-menu-item>
            <el-submenu index="2">
              <template slot="title">我的工作台</template>
              <el-menu-item index="2-1">选项1</el-menu-item>
              <el-menu-item index="2-2">选项2</el-menu-item>
              <el-menu-item index="2-3">选项3</el-menu-item>
            </el-submenu>
            <el-menu-item index="3"><a href="学生列表.html" target="_self">首页</a></el-menu-item>
```

```
          </el-menu>
        </el-container>
      </el-header>

      <!-- 侧边栏区域 -->
      <el-container style="height: 580px; border: 1px solid #eee">
        <el-aside width="200px" style="background-color: rgb(238, 241, 246)">
          <el-menu :default-openeds="['1']">
            <el-submenu index="1">
              <template slot="title"><i class="el-icon-menu"></i>学工部
</template>
              <el-menu-item-group>
                <el-menu-item index="1-1"><i class="el-icon-help"></i>在校学生
管理</el-menu-item>
                <el-menu-item index="1-2"><i class="el-icon-help"></i>学生升级/
留级</el-menu-item>
                <el-menu-item index="1-3"><i class="el-icon-help"></i>学生就业
情况</el-menu-item>
              </el-menu-item-group>
            </el-submenu>
            <el-submenu index="2">
              <template slot="title"><i class="el-icon-menu"></i>咨询部
</template>
              <el-menu-item-group>
                <el-menu-item index="2-1"><i class="el-icon-help"></i>意向学生
管理</el-menu-item>
                <el-menu-item index="2-2"><i class="el-icon-help"></i>未报名学
生管理</el-menu-item>
                <el-menu-item index="2-3"><i class="el-icon-help"></i>已报名学
生管理</el-menu-item>
              </el-menu-item-group>
            </el-submenu>
            <el-submenu index="3">
              <template slot="title"><i class="el-icon-menu"></i>教研部
</template>
              <el-menu-item-group>
                <el-menu-item index="3-1"><i class="el-icon-help"></i>已有课程
管理</el-menu-item>
                <el-menu-item index="3-2"><i class="el-icon-help"></i>正在研发
课程管理</el-menu-item>
                <el-menu-item index="3-3"><i class="el-icon-help"></i>新技术课
程管理</el-menu-item>
              </el-menu-item-group>
            </el-submenu>
          </el-menu>
        </el-aside>

        <!-- 主区域 -->
        <el-container>
          <el-main>
```

```
            <b style="color: red;font-size: 20px;">学生列表</b>
            <div style="float:right">
              <el-button type="primary">添加学生</el-button>
            </div>
            <el-table :data="tableData">
              <el-table-column prop="date" label="日期" width="140">
              </el-table-column>
              <el-table-column prop="name" label="姓名" width="120">
              </el-table-column>
              <el-table-column prop="address" label="地址">
              </el-table-column>
              <el-table-column label="操作" width="180">
                <el-button type="warning">编辑</el-button>
                <el-button type="danger">删除</el-button>
              </el-table-column>
            </el-table>
          </el-main>
        </el-container>
      </el-container>
    </el-container>
  </div>
</body>
<script>
  new Vue({
    el: "#div",
    data: {
      tableData: [
        {
          date: "2021-08-02",
          name: "张三",
          address: "广州市天河区"
        }, {
          date: "2021-08-02",
          name: "李四",
          address: "广州市天河区"
        }, {
          date: "2021-08-02",
          name: "王五",
          address: "广州市天河区"
        },
      ]
    }
  });
</script>

</html>
```

课堂作业

- 完成vue.js 和 elementui 的所有案例

- 代码上传到 git

Vue.js 高级使用

自定义组件

- 学完了 Element 组件后，我们会发现组件其实就是自定义的标签。
- 本质上，组件是带有一个名字且可复用的 Vue 实例，我们是可以自己定义的。
- 定义格式

JavaScript 复制代码

```
Vue.component(组件名称, {
 props:组件的属性,
 data: 组件的数据函数,
 template: 组件解析的标签模板
})
```

- **代码实现**

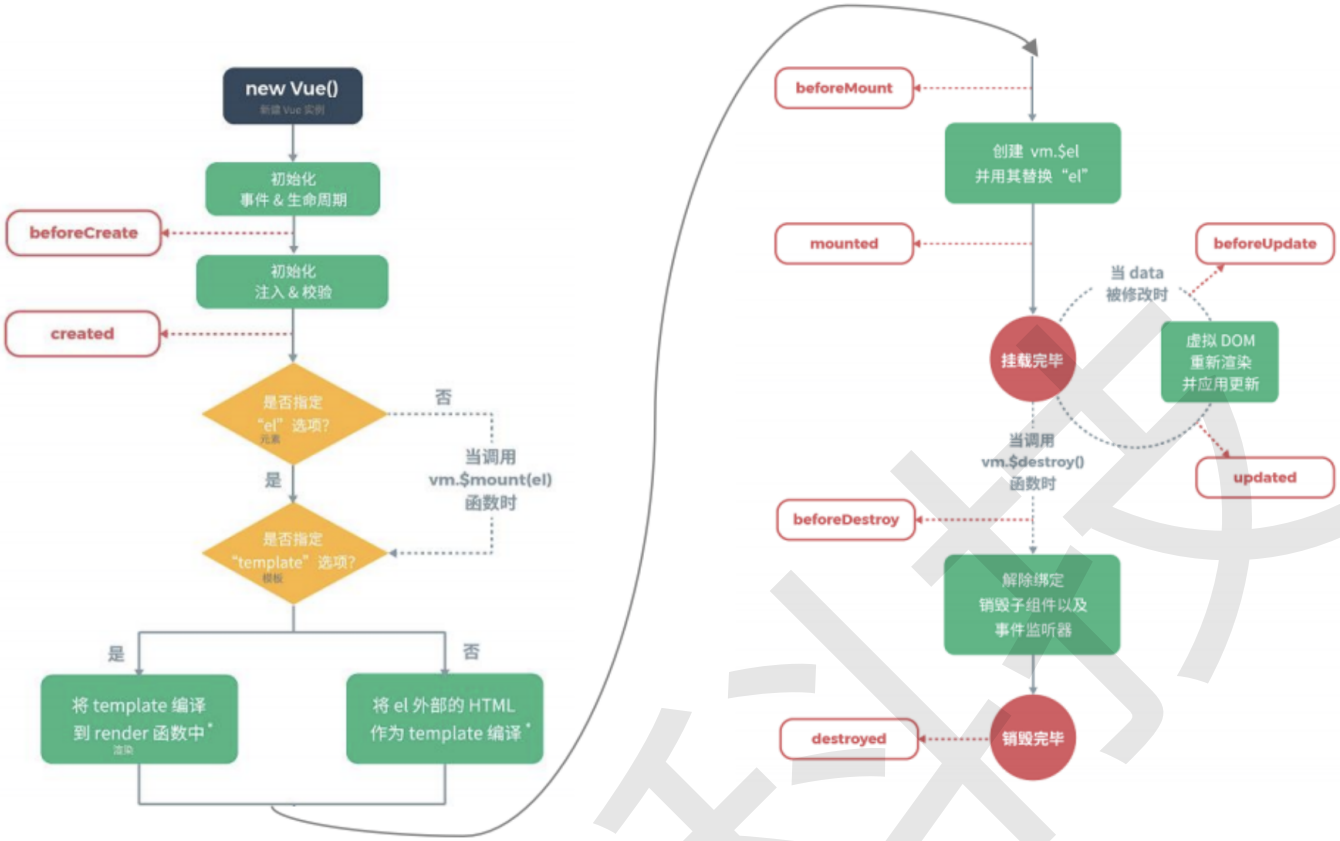
HTML 复制代码

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>自定义组件</title>
    <script src="vue/vue.js"></script>
  </head>
  <body>
    <div id="div">
      <my-button>我的按钮</my-button>
    </div>
  </body>
  <script>
    Vue.component("my-button",{
      // 属性
      props:["style"],
      // 数据函数
      data: function(){
        return{
          msg:"我的按钮"
        }
      },
      //解析标签模板
      template:"<button style='color:red'>{{msg}}</button>"
    });

    new Vue({
      el:"#div"
    });
  </script>
</html>
```

Vue.js 生命周期

- **Vue.js 的生命周期有八个阶段**


new Vue()
初始化
事件 & 生命周期
beforeCreate
初始化
注入 & 校验
created
是否指定
"el" 选项?
否
当调用
vm.$mount(el)
函数时
是
是否指定
"template" 选项?
是
否
将 template 编译
到 render 函数中
将 el 外部的 HTML
作为 template 编译
beforeMount
创建 vm.$el
并用其替换 "el"
mounted
beforeUpdate
当 data
被修改时
挂载完毕
虚拟 DOM
重新渲染
并应用更新
当调用
vm.$destroy()
函数时
updated
beforeDestroy
解除绑定
销毁子组件以及
事件监听器
destroyed
销毁完毕

- 代码实现

HTML 复制代码

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>生命周期</title>
    <script src="vue/vue.js"></script>
  </head>
  <body>
    <div id="app">
      {{message}}
    </div>
  </body>
  <script>
    let vm = new Vue({
      el: '#app',
      data: {
        message: 'Vue的生命周期'
      },
      beforeCreate: function() {
        console.group('------beforeCreate创建前状态------');
        console.log("%c%s", "color:red", "el     : " + this.$el); //undefined
        console.log("%c%s", "color:red", "data   : " + this.$data);
//undefined
        console.log("%c%s", "color:red", "message: " +
this.message);//undefined
      },
      created: function() {
        console.group('------created创建完毕状态------');
        console.log("%c%s", "color:red", "el     : " + this.$el); //undefined
        console.log("%c%s", "color:red", "data   : " + this.$data); //已被初始
化
        console.log("%c%s", "color:red", "message: " + this.message); //已被初
始化
      },
      beforeMount: function() {
        console.group('------beforeMount挂载前状态------');
        console.log("%c%s", "color:red", "el     : " + (this.$el)); //已被初始
化
        console.log(this.$el);
        console.log("%c%s", "color:red", "data   : " + this.$data); //已被初始
化
        console.log("%c%s", "color:red", "message: " + this.message); //已被初
始化
      },
      mounted: function() {
        console.group('------mounted 挂载结束状态------');
        console.log("%c%s", "color:red", "el     : " + this.$el); //已被初始化
```

```
        console.log(this.$el);
        console.log("%c%s", "color:red", "data   : " + this.$data); //已被初始化
        console.log("%c%s", "color:red", "message: " + this.message); //已被初始化
      },
      beforeUpdate: function() {
        console.group('beforeUpdate 更新前状态===============》');
        let dom = document.getElementById("app").innerHTML;
        console.log(dom);
        console.log("%c%s", "color:red", "el     : " + this.$el);
        console.log(this.$el);
        console.log("%c%s", "color:red", "data   : " + this.$data);
        console.log("%c%s", "color:red", "message: " + this.message);
      },
      updated: function() {
        console.group('updated 更新完成状态===============》');
        let dom = document.getElementById("app").innerHTML;
        console.log(dom);
        console.log("%c%s", "color:red", "el     : " + this.$el);
        console.log(this.$el);
        console.log("%c%s", "color:red", "data   : " + this.$data);
        console.log("%c%s", "color:red", "message: " + this.message);
      },
      beforeDestroy: function() {
        console.group('beforeDestroy 销毁前状态===============》');
        console.log("%c%s", "color:red", "el     : " + this.$el);
        console.log(this.$el);
        console.log("%c%s", "color:red", "data   : " + this.$data);
        console.log("%c%s", "color:red", "message: " + this.message);
      },
      destroyed: function() {
        console.group('destroyed 销毁完成状态===============》');
        console.log("%c%s", "color:red", "el     : " + this.$el);
        console.log(this.$el);
        console.log("%c%s", "color:red", "data   : " + this.$data);
        console.log("%c%s", "color:red", "message: " + this.message);
      }
    });


    // 销毁Vue对象
    //vm.$destroy();
    //vm.message = "hehe";  // 销毁后 Vue 实例会解绑所有内容

    // 设置data中message数据值
    vm.message = "good...";
  </script>
</html>
```

CROS

CORS是一个W3C标准，全称是"跨域资源共享"（Cross-origin resource sharing）。

它允许浏览器向跨源服务器，发出XMLHttpRequest请求，从而克服了AJAX只能同源使用的限制。

CORS需要浏览器和服务器同时支持。目前，所有浏览器都支持该功能，IE浏览器不能低于IE10。

整个CORS通信过程，都是浏览器自动完成，不需要用户参与。对于开发者来说，CORS通信与同源的AJAX通信没有差别，代码完全一样。浏览器一旦发现AJAX请求跨源，就会自动添加一些附加的头信息，有时还会多出一次附加的请求，但用户不会有感觉。

因此，实现CORS通信的关键是服务器。只要服务器实现了CORS接口，就可以跨源通信。

Java 复制代码

```java
package com.rushuni.api.filter;

import javax.servlet.*;
import javax.servlet.annotation.WebFilter;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

/**
 * @author rushuni
 * @date 2021/08/02
 */
@WebFilter("/*")
public class CorsFilter implements Filter {

    @Override
    public void init(FilterConfig filterConfig) throws ServletException {

    }

    @Override
    public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain) throws IOException, ServletException {
        System.out.println("Filter CROS 过滤器！！");
        HttpServletResponse httpResponse = (HttpServletResponse) response;

        // 允许所有的请求域名访问我们的跨域资源，可以固定单个或者多个内容
        httpResponse.setHeader("Access-Control-Allow-Origin", "*");
        // 允许所有的请求域名访问我们的跨域资源，可以固定单个或者多个内容
        // httpResponse.setHeader("Access-Control-Allow-Origin", "http://localhost:5500");
        // 允许何种请求方法访问该跨域资源服务器
        httpResponse.setHeader("Access-Control-Allow-Methods", "POST, GET, OPTIONS, DELETE,PUT");
        // 预检请求的有效期，单位为秒。有效期内，不会重复发送预检请求
        httpResponse.setHeader("Access-Control-Max-Age", "3600");
        httpResponse.addHeader("Access-Control-Allow-Headers",
                "Accept,Origin, No-Cache, X-Requested-With, If-Modified-Since, Pragma, Last-Modified, Cache-Control, Expires, Content-Type, X-E4M-With");// 允许所有的请求header访问，可以自定义设置任意请求头信息 后期加上 authorization
        // 是否允许用户发送、处理cookie
        httpResponse.setHeader("Access-Control-Allow-Credentials", "true");
        // 如果额外设置自己的头需要在这定义
        httpResponse.setHeader("Access-Control-Expose-Headers", "Access-Token");
        chain.doFilter(request, httpResponse);
    }

    @Override
```

```
43      public void destroy() {
44
45      }
46  }
```

Vue异步操作

- **在Vue中发送异步请求，本质上还是AJAX。**
- **在 Vue 中，不会使用 jQuery，而是用 axios 这个插件来简化操作！**
- **axios 官网：**https://axios-http.com/

使用步骤

1. 引入axios核心js文件。
2. 调用axios对象的方法来发起异步请求。
3. 调用axios对象的方法来处理响应的数据。

axios常用方法

- **代码实现**

HTML 复制代码

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>异步操作</title>

    <!-- 引入 Vue.js -->
    <script src="https://cdn.jsdelivr.net/npm/vue@2/dist/vue.js"></script>
    <script src="https://unpkg.com/axios/dist/axios.min.js"></script>
</head>
<body>
    <div id="div">
        {{name}}
        <button @click="send()">发起请求</button>
    </div>
</body>
<script>
    new Vue({
        el:"#div",
        data:{
            name:"张三"
        },
        methods:{
            send(){
                // GET方式请求
                axios.get("testServlet?name=" + this.name)
                    .then(resp => {
                        alert(resp.data);
                    })
                    .catch(error => {
                        alert(error);
                    })

                // POST方式请求
                // axios.post("testServlet","name="+this.name)
                //     .then(resp => {
                //         alert(resp.data);
                //     })
                //     .catch(error => {
                //         alert(error);
                //     })
            }
        }
    });
</script>
</html>
```

Vue.js 高级总结

- 自定义组件：本质上，组件是带有一个名字且可复用的 Vue 实例，我们可以自己来定义。

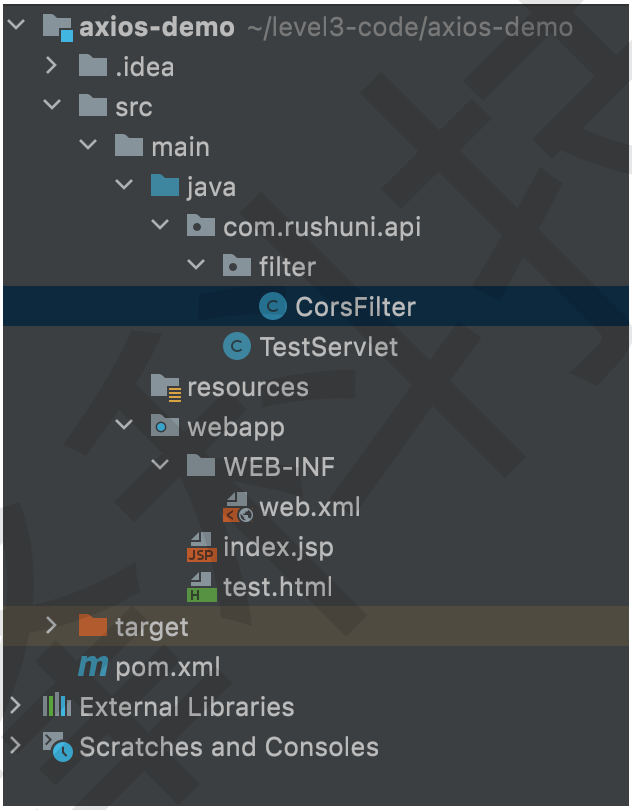
```JavaScript
Vue.component(组件名称, {
 props:组件的属性,
 data: 组件的数据函数,
 template: 组件解析的标签模板
})
```

- 生命周期：核心八个阶段
  - beforeCreate：创建前
  - created：创建后
  - beforeMount：载入前
  - mounted：载入后
  - beforeUpdate：更新前
  - updated：更新后
  - beforeDestroy：销毁前
  - destroyed：销毁后
- 异步操作：通过 axios 插件来实现。

今晚作业

- 完成今天所有案例代码
- 代码上传到 git


01-入门案例
js
01-快速入门.html
02-入门案例升级.html
02-常用指令
01-文本插值.html
02-绑定属性.html
03-条件渲染.html
04-列表渲染.html
05-事件绑定.html
06-表单绑定.html
03-ElementUI 案例
01-入门案例.html
02-基础布局.html
03-容器布局.html
04-表单组件.html
05-表格组件.html
06-顶部导航栏.html
07-侧边导航栏.html
04-综合案例
img
学生列表.html
05-高级
01-自定义组件.html
02-生命周期.html
03-axios.html


axios-demo ~/level3-code/axios-demo
.idea
src
main
java
com.rushuni.api
filter
CorsFilter
TestServlet
resources
webapp
WEB-INF
web.xml
index.jsp
test.html
target
pom.xml
External Libraries
Scratches and Consoles