模板引擎

> 模板引擎可以让将数据和HTML模板更加友好的结合，省去繁琐的字符串拼接，使代码更加易于维护。

ART-TEMPLATE

高性能 JavaScript 模板引擎，使用`npm install art-template`命令下载。

模板编译

通过调用模板引擎提供的template函数，告知模板引擎将特定模板和特定数据进行拼接，最终返回拼接结果。

```
// 模板引擎导入
const template = require('art-template');
// 模板编译
// 1.模板位置
// 2.向模板中传递要拼接的数据，对象类型，对象属性可以直接在模板中使用。
// result 拼接结果
const result = template('./views/index.html', {msg: 'Hello, art-template'});
```

模板语法

通过模板引擎提供的特殊语法，告知模板引擎数据和模板具体的拼接细节。

插值表达式

即{{}}，用来显式数据，将数据变量放入双大括号之中即可。

原始语法

<% %>

```
标准语法：  {{ 数据 }}
原始语法：<%=数据  %>
```

插值表达式中可以进行运算，最终显式运算的结果。

```
  <!-- 标准语法 -->
<h2>{{value}}</h2>
<h2>{{a ? b : c}}</h2>
<h2>{{a + b}}</h2>

  <!-- 原始语法 -->
<h2><%= value %></h2>
<h2><%= a ? b : c %></h2>
<h2><%= a + b %></h2>
```

原文输出

如果数据中携带HTML标签，默认不会解析标签，会将其转义后输出。使用以下方式可以解析标签。

```
{{@ value }}
```

条件判断

```
<!-- 标准语法 -->
 {{if 条件}} ... {{/if}}
 {{if v1}} ... {{else if v2}} ... {{/if}}
 <!-- 原始语法 -->
 <% if (value) { %> ... <% } %>
 <% if (v1) { %> ... <% } else if (v2) { %> ... <% } %>
```

数据循环

```
 {{each target}}
     {{$index}} {{$value}}
 {{/each}}
  <!-- 原始语法 -->
 <% for(var i = 0; i < target.length; i++){ %>
     <%= i %> <%= target[i] %>
 <% } %>
```

子模板

使用子模板可以将网站公共区块(头部、底部)抽离到单独的文件中。

```
  <!-- 标准语法 -->
 {{include './header.art'}}
  <!-- 原始语法 -->
 <% include('./header.art') %>
```

模板继承

使用模板继承可以将网站HTML骨架抽离到单独的文件中，其他页面模板可以继承骨架文件。

```
{{extend './layout.html'}}
{{block 'head'}} ... {{/block}}
```

学生档案信息管理案例

1. 建立项目文件夹并生成项目描述文件
2. 创建网站服务器实现客户端和服务器端通信
3. 连接数据库并根据需求设计学员信息表
4. 创建路由并实现页面模板呈递
    - 第三方模块 router

功能：实现路由

使用步骤：

1. 获取路由对象
2. 调用路由对象提供的方法创建路由
3. 启用路由，使路由生效

```
const getRouter = require('router')
const router = getRouter();
router.get('/add', (req, res) => {
    res.end('Hello World!')
})
server.on('request', (req, res) => {
    router(req, res)
})
```

5. 实现静态资源访问
    - 第三方模块serve-static
        1. 引入serve-static模块获取创建静态资源服务功能的方法
        2. 调用方法创建静态资源服务并指定静态资源服务目录
        3. 启用静态资源服务功能

```
const serveStatic = require('serve-static')
const serve = serveStatic('public')
server.on('request', () => {
    serve(req, res)
})
server.listen(3000)
```

6. 实现学生信息添加功能
    - 在模板的表单中指定请求地址与请求方式
    - 为每一个表单项添加name属性
    - 添加实现学生信息功能路由
    - 接收客户端传递过来的学生信息
    - 将学生信息添加到数据库中
    - 将页面重定向到学生信息列表页面
7. 实现学生信息展示功能
    1. 从数据库中将所有的学生信息查询出来
    2. 通过模板引擎将学生信息和HTML模板进行拼接
    3. 将拼接好的HTML模板响应给客户端