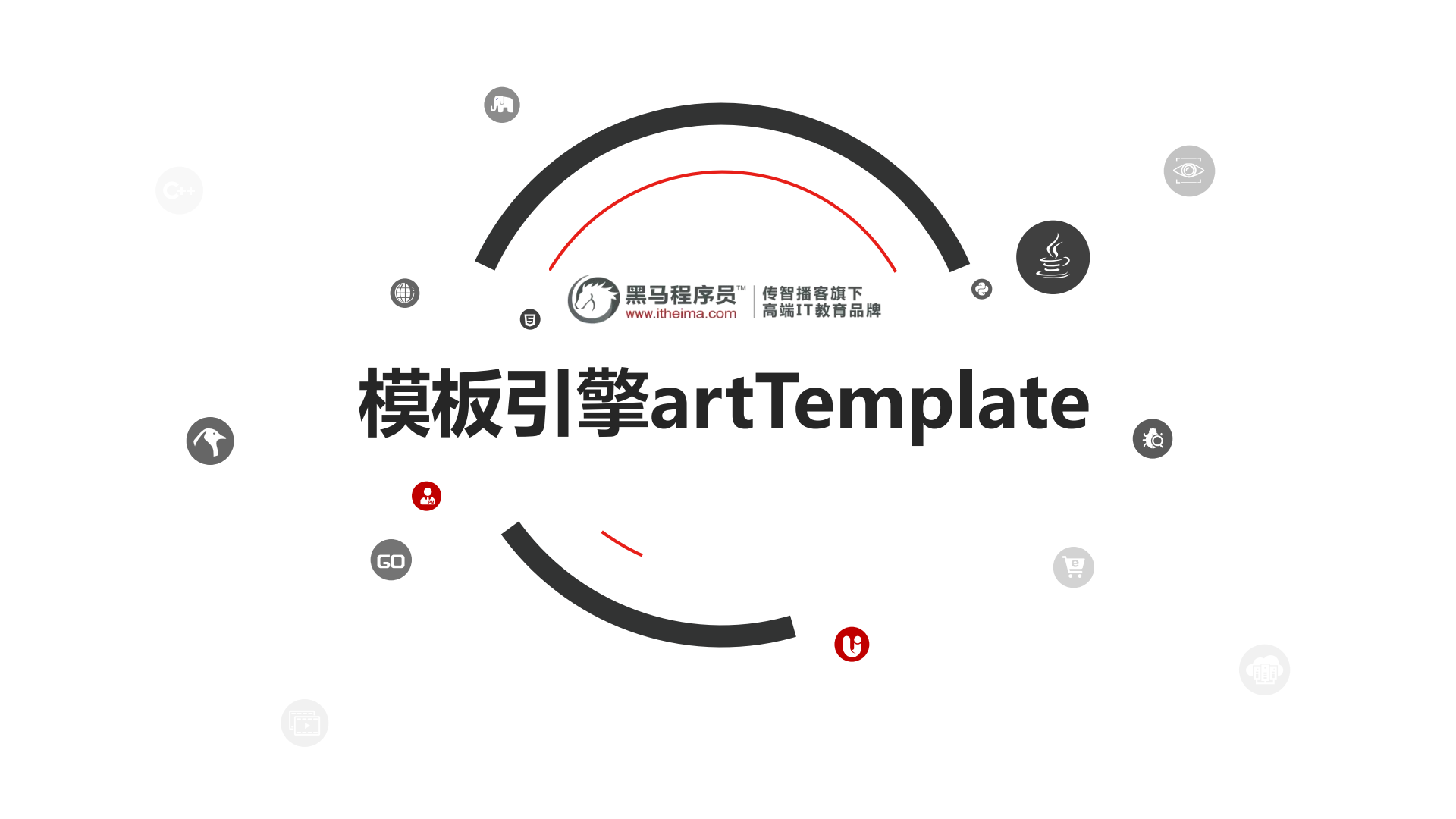


模板引擎artTemplate

目录 Contents

1. 模板引擎的基础概念

1.1 模板引擎

模板引擎是第三方模块。

让开发者以更加友好的方式拼接字符串，使项目代码更加清晰、更加易于维护。

```
// 未使用模板引擎的写法
var ary = [{ name: '张三', age: 20 }];
var str = '<ul>';
for (var i = 0; i < ary.length; i++) {
    str += '<li>\
        <span>'+ ary[i].name +'</span>\
        <span>'+ ary[i].age +'</span>\
    </li>';
}
str += '</ul>';
```

```
<!-- 使用模板引擎的写法 -->
<ul>
    {{each ary}}
        <li>{{$value.name}}</li>
        <li>{{$value.age}}</li>
    {{/each}}
</ul>
```

1. 模板引擎的基础概念

1.2 art-template模板引擎

1. 在命令行工具中使用 **npm install art-template** 命令进行下载
2. 使用`const template = require('art-template')`引入模板引擎
3. 告诉模板引擎要拼接的数据和模板在哪 const html = template(‘模板路径’ , 数据);
4. 使用模板语法告诉模板引擎，模板与数据应该如何进行拼接

1. 模板引擎的基础概念

1.3 art-template代码示例

```
// 导入模板引擎模块
const template = require('art-template');
// 将特定模板与特定数据进行拼接
const html = template('./views/index.art',{
    data: {
        name: '张三',
        age: 20
    }
});
```

```
<div>
    <span>{{data.name}}</span>
    <span>{{data.age}}</span>
</div>
```

目录 Contents

2. 模板引擎语法

2.1 模板语法

- art-template同时支持两种模板语法：标准语法和原始语法。
- 标准语法可以让模板更容易读写，原始语法具有强大的逻辑处理能力。

标准语法： {{ 数据 }}

原始语法： <%=数据 %>

2. 模板引擎语法

2.2 输出

将某项数据输出在模板中，标准语法和原始语法如下：

- 标准语法：{{ 数据 }}
- 原始语法：<%=数据 %>

```
<!-- 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>
```

2. 模板引擎语法

2.3 原文输出

如果数据中携带HTML标签，默认模板引擎不会解析标签，会将其转义后输出。

- 标准语法：{{@ 数据 }}
- 原始语法：<%-数据 %>

```
<!-- 标准语法 -->
<h2>{{@ value }}</h2>
<!-- 原始语法 -->
<h2><%- value %></h2>
```

2. 模板引擎语法

2.4 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) { %> ... <% } %>
```

2. 模板引擎语法

2.5 循环

- 标准语法：{{each 数据}} {{/each}}
- 原始语法：<% for() { %> <% } %>

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

2.6 子模版

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。

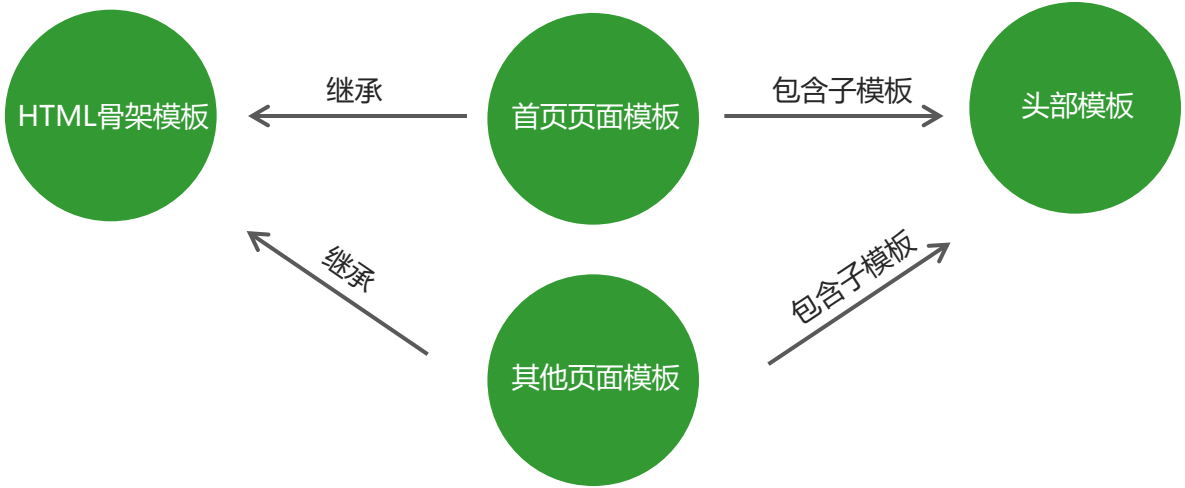
- 标准语法：{{include '模板'}}
- 原始语法：<%include('模板') %>

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

2. 模板引擎语法

2.7 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。



2. 模板引擎语法

2.7 模板继承

坑一：填充css内容

坑二：填充js内容

坑三：填充页面主体内容

填充一：main.css

填充二：index.js

填充三： Hello

2. 模板引擎语法

2.8 模板继承示例

```
<!doctype html>
<html>
    <head>
        <meta charset="utf-8">
        <title>HTML骨架模板</title>
        {{block 'head'}}{{/block}}
    </head>
    <body>
        {{block 'content'}}{{/block}}
    </body>
</html>
```

2. 模板引擎语法

2.8 模板继承示例

```
<!--index.art 首页模板-->
{{extend './layout.art'}}
{{block 'head'}} <link rel="stylesheet" href="custom.css"> {{/block}}
{{block 'content'}} <p>This is just an awesome page.</p> {{/block}}
```

2. 模板引擎语法

2.9 模板配置

1. 向模板中导入变量 template.defaults.imports.变量名 = 变量值;
2. 设置模板根目录 template.defaults.root = 模板目录
3. 设置模板默认后缀 template.defaults.extname = `'.art'`

目录 Contents

3. 案例

3.1 案例介绍 – 学生档案管理

目标：模板引擎应用，强化node.js项目制作流程。

知识点：http请求响应、数据库、模板引擎、静态资源访问。

3.2 制作流程

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
5. 实现静态资源访问
6. 实现学生信息添加功能
7. 实现学生信息展示功能

3.3 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)
})
```

3.4 第三方模块 serve-static

功能：实现静态资源访问服务

步骤：

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)
```