### 什么是express

> express是nodeJS的WEB开发框架,除了express外,还有其他比较流行的框架,比如koa,egg,hapi等

> http://expressjs.com/

> http://www.expressjs.com.cn/

### express 安装

$ npm install express --save

### 基本使用

### 路由结构 app.METHOD(PATH, HANDLER)

\* app 是一个 express 实例

\* METHOD 是某个 HTTP 请求方式中的一个

> \* get 查询

> \* post 新增

> \* put 修改

> \* delete 删除

\* PATH 是服务器端的路径

> \* 匹配字符串

> \* 使用正则中的?、+、\* 和 ()

> \* 使用正则

\* HANDLER 是当路由匹配到时需要执行的回调函数

> \* 使用多个回调函数处理路由需要指定next

\* app.all(PATH, HANDLER)

> all 不是一个http请求,主要用来加载中间件

### 路由模块化

\* 用express.Router实例化一个router对象

\* 使用router.METHOD(PATH, HANDLER)来处理路由

\* 导出router对象

\* 用app.use(PATH,router对象)来使用导出的router对象

### 托管静态文件

\* app.use(express.static(静态资源目录));

> 所有的静态资源的请求路径都会相对于静态资源目录,

> 例如:如果请求路径的是/css/index.css,设置的静态资源目录是static,则服务器端获取文件的路径是static/css/index.css

\* app.use('虚拟路径',express.static(静态资源目录))

> 设置了虚拟路径后,静态资源请求时需要加上虚拟路径

> 虚拟路径是根本不存在的目录

### 中间件介绍

> \* 中间件其实就是一个函数,在收到请求和返回响应之间处理一些操作,这个函数可以访问请求对象(req), 响应对象(res)

> \* 如果当前中间件没有终结请求-响应循环，则必须调用 next() 方法将控制权交给下一个中间件，否则请求就会挂起。

> \* next()代表中间件数组中的下一个函数,在express内部有一个专门存放发出响应之前要执行的所有函数的数组,即中间件数组,每次执行app.use(fn)都会把函数添加到数组中

### 常用中间件

\* body-parse 处理post请求中间件

> \* 安装并引入

> \* 使用中间

```

//解析json

app.use(bodyParser.json());

//解析urlencoded内容

app.use(bodyParser.urlencoded({ extended: false }));

app.post('/add',(req,res)=>{

console.log(req.body)//post数据

})

```

### 模板的使用

\* 安装

npm i swig --save

\* 集成

```javascript

swig.setDefaults({

cache: false

})

//1. 配置应用模板

//第一个参数是模板名称,同时也是模板文件的扩展名

//第二个参数是解析模板的方法

app.engine('html', swig.renderFile);

//2.配置模板的存放目录

//第一参数必须是views

//第二个参数是模板存放的目录

app.set('views', './views')

//3.注册模板引擎

//第一个参数必须是view engine

//第二个参数是模板名称,也就是app.engine的第一个参数

app.set('view engine', 'html')

app.get('/',(req,res)=>{

//4.渲染模板

//第一个参数是相对于模板目录的文件

//第二个参数是传递给模板的数据

res.render('index',{

title:'跨猪网',

content:'我是内容'

})

})

```

### swig的使用

\* 变量

```javascript

{{ name }}

{{ obj.name }}

```

\* 判断

```javascript

{ % if name === 'Tom' % }

hello Tom

{ % endif % }

{ % if name === 'Tom' % }

hello Tom

{ % elseif name === 'Leo' % }

hello Leo

{ % else % }

hello guy

{ % endif % }

```

\* 循环

```javascript

{ % for key, val in arr % }

<p>{ { key } } -- { { val } }</p>

{ % endfor % }

```

\* 继承模板

> \* 定义父模板

> \* 父模板里定义block块,子模板可以对这block继承

```html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>{% block title %}{% endblock %}</title>

{% block head %}{% endblock %}

</head>

<body>

{% block content %}{% endblock %}

</body>

</html>

```

```html

{% extends './layout.html' %}

{% block title %} index {% endblock %}

{% block content %}

<div>

<h1>hello kuazhu</h1>

<div>

{% endblock %}

```

\* 包含模板

```html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>{% block title %}{% endblock %}</title>

{% block head %}{% endblock %}

</head>

<body>

{% block content %}{% endblock %}

</body>

{% include "./footer.html" %}

</html>

```