# express

Express是一个简洁、灵活的node.js Web应用开发框架，它提供一系列强大的功能

- 模板解析
- 静态文件服务
- 中间件
- 路由控制

# 路由控制

- get方法 —— 根据请求路径来处理客户端发出的GET请求

```
app.get(path,function(request, response));
```

- path为请求的路径
- 第二个参数为处理请求的回调函数，有两个参数分别是 `request` 和 `response` ，代表请求信息和响应信息。
- `app.all()` 函数可以匹配所有的 **HTTP方法** ，也就是说它可以过滤所有路径的请求，

# 中间件

- 中间件(middleware)就是处理HTTP请求的 **函数** ，用来完成各种 **特定的任务** ，比如检查用户是否登录、分析数据、以及其他在需要最终将数据发送给用户之前完成的任务。
- 它最大的特点就是，一个中间件 **处理完** ，可以把相应数据再传递给 **下一个** 中间件。
- 如果调用回调函数的 `next` 参数表示将请求数据传递给下一个中间件。

```
app.use([path], function(request, response, next){}); //可选参数path默认为"/"
```

# 获取请求参数

- req.host 返回请求头里取的 **主机名** (不包含端口号)。
- req.path 返回请求的URL的 **路径名** 。
- req.query 是一个可获取客户端 `get` 请求路径参数的对象属性，包含着被解析过的 **请求参数对象** ，默认为{}。
- req.params 是一个 **路径参数** 对应的对象。

# send

send()方法向浏览器发送一个响应信息，并可以智能处理不同类型的数据。并地输出响应时会自动进行一些设置，比如`HEAD`信息、`HTTP`缓存支持等等。

- 当参数为一个String时，Content-Type默认设置为"text/html"。

```
res.send([body|status], [body]);
```

- 当参数为Array或Object时，Express会返回一个JSON

```
res.send({ user: 'tobi' }); //{"user":"tobi"}
```

- 不能使用数字作为参数，如果要返回入码要用`res.sendStatus`方法

1. 指定渲染模板引擎

```
app.set('view engine', 'ejs');
```

2. 设置放模板文件的目录

```
app.set('views',path.join(__dirname,'/'));
```

3.render函数，对网页模板进行渲染 在渲染模板时 locals 可为其模板传入量值，在模板中就可以调用所传变量了，

```
: res.render(view, [locals], callback);
```

4.原理

```
var fs = require('fs'); // 此模板引擎依赖 fs 模块
app.engine('ntl', function (filePath, options, callback) { // 定义模板引擎
fs.readFile(filePath, function (err, content) {
 if (err) return callback(new Error(err));
 // 这是一个功能极其简单的模板引擎
 var rendered = content.toString().replace('#title#', '<title>'+ options.title +'</titl
 .replace('#message#', '<h1>'+ options.message +'</h1>');
 return callback(null, rendered);
})
});
app.set('views', './views'); // 指定视图所在的位置
```

# 静态文件服务中间件

`express.static` 是 `Express` 内置的唯一一个中间件。是基于 serve-static 开发的，负责托管 Express 应用内的静态资源。

- 如果要在网页中加载静态文件（css、js、img），就需要另外指定一个 **存放静态文件的目录**，
- 项目目录下添加一个存放静态文件的目录为 **public**
- 在 **public** 目录下在添加三个存放js、css、img的目录,把相关文件放到相应的目录下
- 当浏览器发出静态文件请求时，服务器端就会到 **这个目录** 下去寻找相关文件
- 每个应用可有 **多个** 静态目录

```
app.use(express.static(require('path').join(__dirname, 'public')),{options});
```

| | | | |
|---|---|---|---|
| **dotfiles** | 是否对外输出文件名以点（.）开头的文件。可选值为 “allow”、“deny” 和 “ignore” | String | “ignore” |
| **etag** | 是否启用 etag 生成 | Boolean | true |
| **extensions** | 设置文件 **扩展名** 备份选项 | Array | [] |
| **index** | 发送 **目录索引** 文件，设置为 false 禁用目录索引。 | Mixed | “index.html” |
| **lastModified** | 设置 Last-Modified 头为文件在操作系统上的 **最后修改日期** 。可能值为 true 或 false。 | Boolean | true |
| **maxAge** | 以毫秒或者其字符串格式设置 `Cache-Control` 头的 `max-age` 属性 | Number | 0 |
| **redirect** | 当路径为目录时，**重定向** 至 “/”。 | Boolean | true |
| **setHeaders** | 设置 `HTTP头` 的函数。 | Function | |

# post方法

根据请求路径来处理客户端发出的Post请求

```
var bodyParser = require('body-parser');
app.use(bodyParser.urlencoded({extended:true}));
app.post(path,function(req, res));
```

`req.body` 属性解析客户端的 **post** 请求参数，通过它可获取请求路径的参数值。