Express基础与实战

标签： 未归档

Express介绍

Express是一个简洁、灵活的node.js Web应用开发框架,是目前最流行的基于Node.js的Web开发框架. 它提供一系列强大的功能，比如：

- 路由控制
- 参数获取
- 中间件
- send和sendFile
- 静态文件服务
- 模板解析
- 重定向

本地安装

```
$ npm install express
```

引用express以后我们就可以得到一个函数，调用它我们就可以调用express的各种方法

```
var express = require('express');
var app = express();
app.listen(3000);
```

app是什么?

```
app.listen = function listen() {
  var server = http.createServer(this);
  return server.listen.apply(server, arguments);
};
```

> 本质上讲app是一个请求监听函数，当http服务器接收到请求的时候会交由app函数来处理

使用路由

路由就是服务器根据客户端的请求URL路径不同进行不同的处理，返回不同的内容

如何定义路由?

app上有很多的方法，方法名和HTTP请求名一一对应，如
`get`，`post`，`put`，`head`，`delete`，`options` 等，它用来匹配HTTP客户端的请求方式
这种定义路由的方式指的是如果客户端的请求方法为GET,路径和path相等的话则会使用使用第二个函数参数来进行处理请求
可以定义多个路由，这些路由会组成一个线性结构，在请求到来的时候会从上往下依次匹配，如果匹配上一个路由，则会交由路由函数处理并结束请求，如果一直到最后也没有匹配上会报一个错，表示无法处理请求

```
/*
 * get表示匹配客户端的GET请求
 * 第一个参数path为客户端的请求的路径
 * request 代表请求对象，和http服务器中的请求对象是同一个，如果想获取客户端的请求信息从这个对象中获取
 * response 代表响应信息，和http服务器中的响应对象是同一个，如果想写入响应从使用这个对象写入
*/

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

all和星号

> - app.all函数可以匹配所有的HTTP动词
> - 路由中的星号能匹配所有的路径 语法

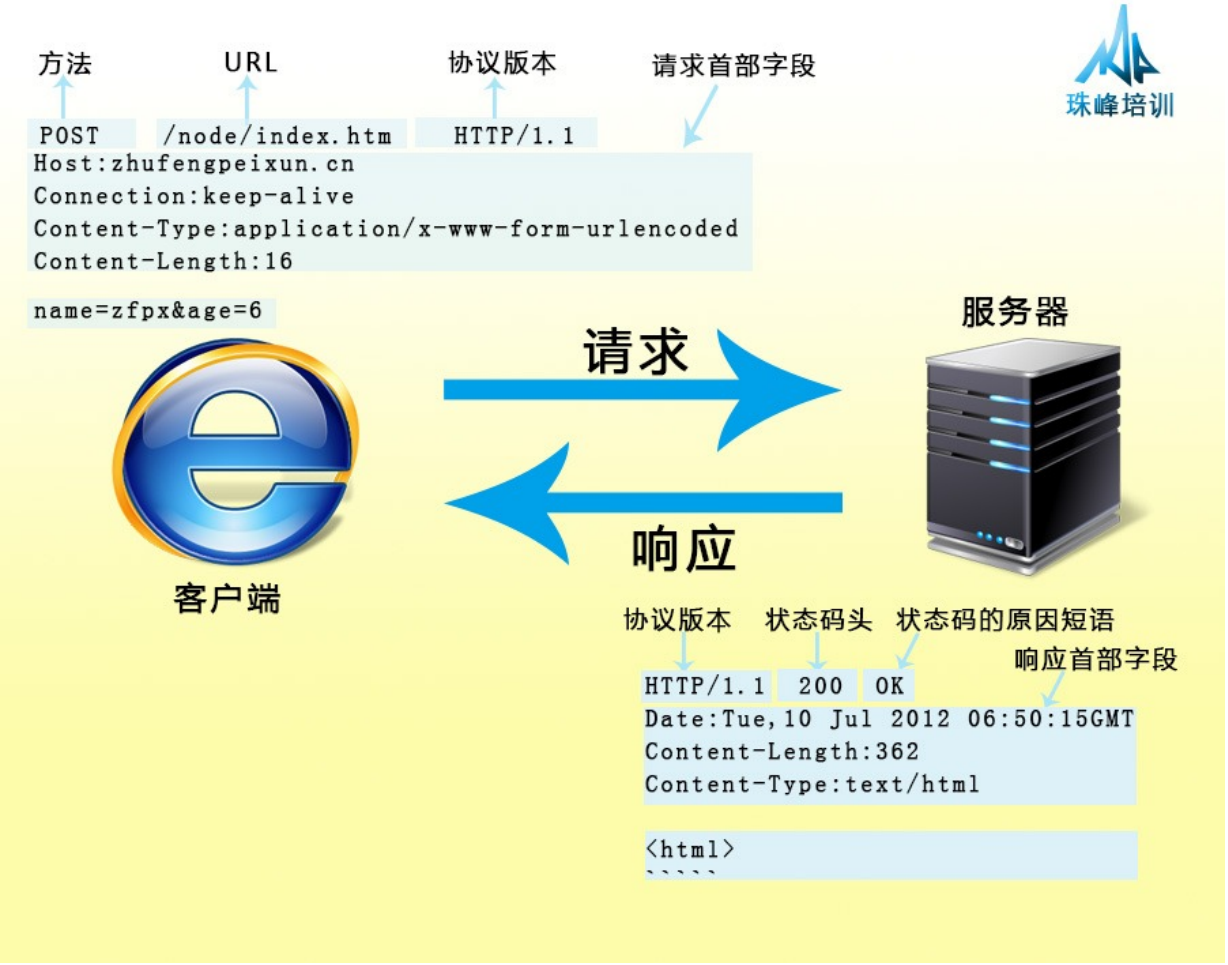
```
app.all("*",function(req,res){
 res.send("404");
})
```

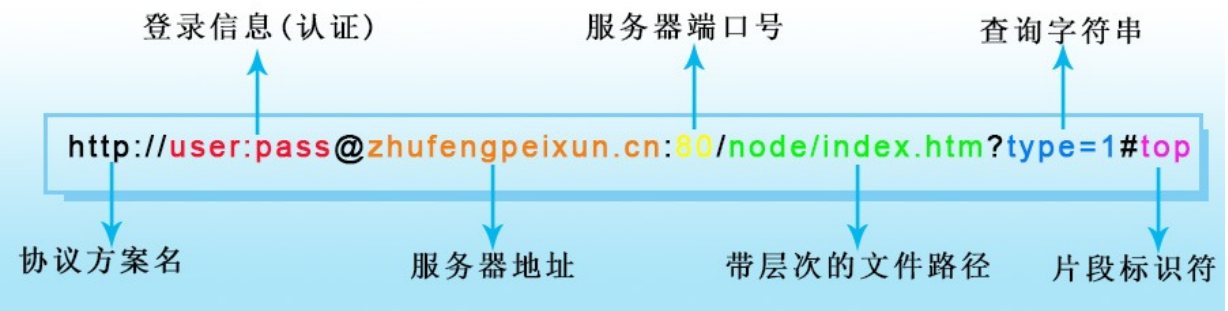
练习

启动一个HTTP服务器并实现以下路由

> - 当客户端以GET方法访问路由 `/signup` 时,返回字段符串**注册**
> - 当客户端以GET方法访问路由 `/signin` 时,返回字段符串**登录**
> - 当客户端以GET方法访问路由 `/signout` 时,返回字段符串**退出**
> - 当客户端以任何方法访问其它路径时,返回字段符串**你访问的路径不存在**

在express中获取请求参数





获取请求方法

```
req.method
```

获取路径

```
req.path
```

获取查询字符串

```
req.query
```

原理

```
var urlObj = require('url').parse(req.url,true);
req.path = urlObj.path;
req.query = urlObj.query;
```

params路径参数

`req.params` 可以用来获取请求URL中的参数值

```
app.get('/:id/:name',function(req,res){
   res.send(req.params.id+" "+req.params.name);
});
```

练习

请从给定的url中获取指定的参数

- 请求的方法
- 域名
- 端口号
- 路径名
- 路径中的用户ID

- 查询字符串中 order 参数的值

```
http://www.zhufengpeixun.cn:9090/users/1?order=1
```

中间件

如果我们想编写一段针对所有路由的公共逻辑的话就可以编写中间件,中间件可以用来为 req 对象增加公共的属性和方法，统一处理响应。
中间件就是处理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、检测用户是否有权限访问等，它的特点是:

- 一个中间件处理完请求和响应可以把相应数据再传递给下一个中间件
- 回调函数的next参数是一个函数，调用它表示调用后续的中间件,并将数据传递给下一个中间件.
- 还可以根据路径来区分执行不同的中间件

```
var express = require('express');
var app = express();
var path = require('path');

app.use(function(req,res,next){
 res.setHeader('Content-Type','text/plain;charset=utf-8');
 next();
});

app.get('/',function(req,res){
 res.end('首页');
});
app.get('/about',function(req,res){
 res.end('关于我们');
});

app.listen(3000);
```

练习

请编写一个中间件,判断用户 URL 路径中是否有查询字符串参数 username ,参数的值是否为 admin ,如果是 admin ,那么就可以向下继续访问，如果不没有这个 username 参数或者参数的值不是 admin 的话，就不要继续向下执行，直接响应一个字符串"你没有权限访问"。

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

1. 当参数为一个String时，Content-Type默认设置为”text/html”
2. 当参数为Array或Object时，Express会返回一个JSON
3. 当参数为一个Number时，并且没有上面提到的任何一条在响应体里，Express会帮你设置一个响应体，比如：200会返回字符”OK”

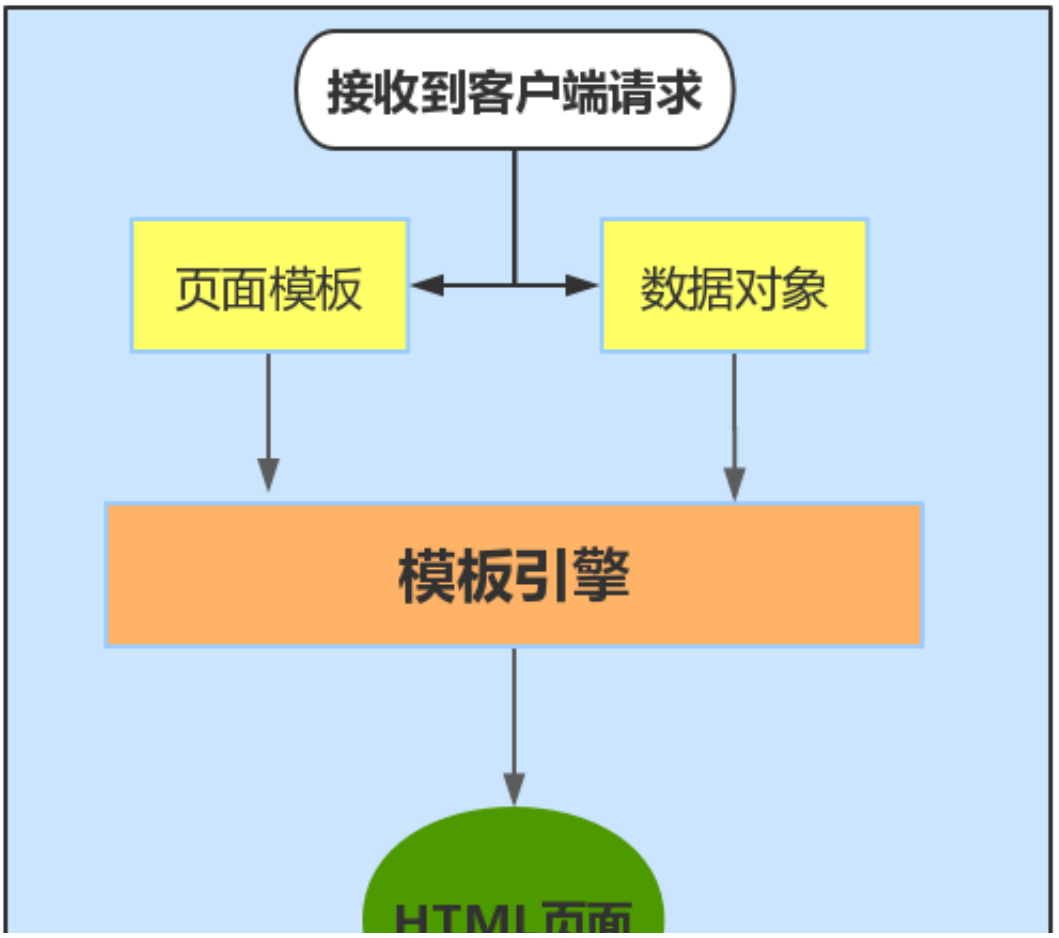
练习

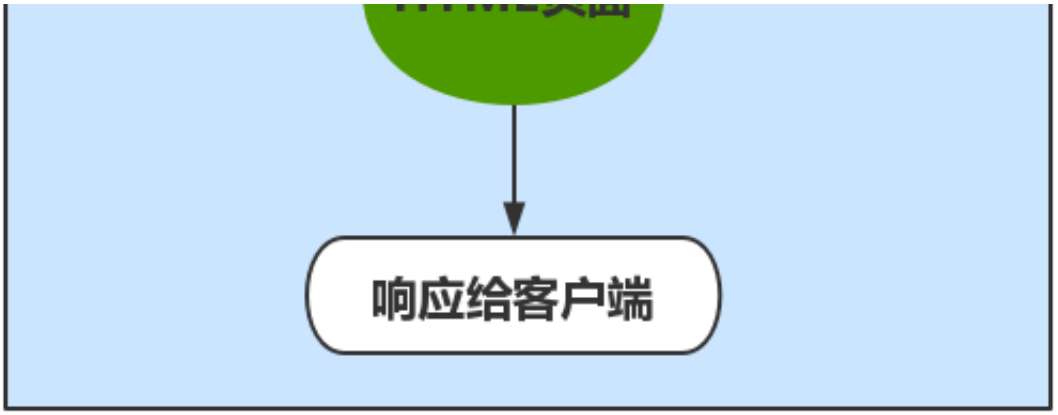
请编写三个路由

1. 当客户端访问 `/users` ,服务器返回一个用户对象数组
2. 当客户端访问一个没有配置的路径时，服务器返回 `404` 状态码，响应体为 `Not Found` 。

模板

模板引擎的功能就是将页面模板和要显示的数据结合起来生成HTML页面。在MVC架构中，模板引擎包含在服务器端。控制器得到用户请求后，从模型获取数据，调用模板引擎，模板引擎把数据和页面模板生成HTML页面发给客户端。





安装模板

```
npm install ejs
```

使用模板

使用ejs模板

```
//指定渲染模板文件的后缀名为ejs
app.set('view engine', 'ejs');
app.set('views',path.join(__dirname,'views'));
res.render('index');
```

模板使用html后缀

// 修改模板文件的后缀名为html

```
app.set( 'view engine', 'html' );
app.set('views',path.join(__dirname,'views'));
// 运行ejs模块
app.engine( '.html', require( 'ejs' ).__express ); //__express是ejs
模块的一个公共属性，表示要渲染的文件扩展名
```

渲染视图

语法

- 参数view就是模板的文件名
- 在渲染模板时 `locals` 可为其模板传入数据对象值

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

静态文件服务器

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，当浏览器发出非HTML文件请求时，服务器端就会到这个目录下去寻找相关文件

```
app.use(express.static(path.join(__dirname,'/')));
```

练习

请自己编写一个静态文件服务器中间件

重定向

redirect方法允许网址的重定向，跳转到指定的url

```
res.redirect([status], url);
```

练习

请编写一个中间件，如果客户端发过来的请求没有带查询参数 `username=admin 的话就重定向到未授权页 /error

晚间作业: 注册登陆实战

实现一个注册登录的功功，描述如下

- 客户端以GET方法访问 /signup ,会返回一个注册的包含用户名和密码两个字段的空白注册表单
- 填写这个空白注册表单，则会向当前路径 /signup 提交**post**请求，提交到后台后把此用户名和密码保存到用户数组里，然后跳转到欢迎页