Node笔记 第4天

Node 结合使用 Express 开启服务

```
// 导入Express
var express = require('express')
// 创建实例化服务
var appServer = express()
// 发起请求监听请求，响应请求
appServer.get('请求路径',function(err,data) {
    //响应的操作
})
// 绑定端口，开启服务
appServer.listen(3000,function() {
    console.log('The appServer is running at port 3000...')
})
```

服务器自动重启

- 我们可以使用一个第三方的工具 nodemon 来帮我们解决修改代码重启服务器问题。
- nodemon 是一个基于 Node.js 开发的第三方命令工具，我们使用的时候要独立安装

```
# npm 全局安装
npm install --global nodemon

# 安装完之后使用
nodemon 文件名  #来执行操作文件
```

只要是通过 nodemon 启动的服务器，则它会监听你的文件变化，当文件发生变化时，自动回帮我们重启服务器。

在 Express 中配置使用 art-template

- 首先 安装：

```
npm install --save art-template
npm install --save
```

- 在这里不需要再引入 art-template,因为express-art-template 这个包需要依赖 art-template 所以必须独立安装一下 art-template。

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

```
// 主要配置代码
// 第一个参数表示：当渲染以 .art 结尾的文件的时候，使用 art-template 模板引擎
app.engine('art', require('express-art-template'));
// 可要可不要
app.set('view options', {
    debug: process.env.NODE_ENV !== 'production'
});
// 使用案例
app.get('/', function (request, response) {
    // 经过配置，现在response对象可以使用render方法，默认是没有的
    // render 方法的第一个参数不能放 路径，默认会去 views 目录下找，
    // 也就是说 Express 有个约定：开发人员把所以的视图文件放在了 views 目录下
    // 如果你想修改默认 render 的路径，不用他规定的 views则可以这样：
    // appServer.set('views',render函数的默认路径)
    response.render('index.art', {
        user: {
            name: 'aui',
            tags: ['art', 'template', 'nodejs']
        }
    });
});
```

如果你希望修改默认的 `views` 视图渲染存储目录，可以：

```
// 注意第一个参数 views 千万不要写错
appServer.set('views',render函数的默认路径)
```

在 Express 中获取表单 GET 请求体数据

Express 内置了一个 API ： `req.query` 来获取 GET 请求体数据，所以可以直接使用

```
request.query
```

在 Express 中获取表单 POST 请求体数据

在 Express 中没有内置获取表单 POST 请求体数据的 API，这里我们需要使用一个第三方包：`body-parser`

安装：

```
npm install --save body-parser
```

配置：

```
var express = require('express')
// 引包
```

```
var bodyParser = require('body-parser')

var app = express()
// 配置 body-parser
// 只要加入这个配置，则在 req 请求对象上会多出来一个属性：body
// 也就是说可以直接通过 req.body 来获取表单请求体数据了
app.use(bodyParser.urlencoded({ extended: false }))
app.use(bodyParser.json())
```

使用：

```
app.use(function (req, res) {
  res.setHeader('Content-Type', 'text/plain')
  res.write('you posted:\n')
//    可以通过 req.body 来获取表单 POST 请求体数据
  res.end(JSON.stringify(req.body, null, 2))
})
```

Express 中提供的提取路由 API 使用

入口文件：app.js:

```
// 职责：
// 创建服务
// 做一些服务相关配置
//       例如：模板引擎
//             body-parser
//             开放静态资源 等...
// 挂载路由
// 监听端口启动服务
var express = require('express')
var appServer = express()
var router = require('./router')

appServer.use(router)

appServer.listen(3000,function() {
    console.log('The appServer is running...')
})
```

路由文件：router.js:

```
// 职责：
// 处理路由
// 根据不同的请求方法+请求路径设置具体的处理函数
// 模块职责要单一，不要乱写
// 我们划分模块的目的就是为了增强项目代码的可维护性
```

```
// 提升开发效率
var express = require('express')
var router = express.Router()

router.get('请求地址',function(request,response) {
    response.send('hello world')
})
module.exports = router
```