基于nodejs的网站前后端开发流程

1. 分析需求
2. 架构设计，技术选型{

模式：后端渲染

设计模式：MVC

前端模板引擎：jade

后端服务器：node

后端框架：express

Session中间件：Express-session

Session存储中间件：connect-mongo

表单接受及文件上传中间件：express-formidable

配置文件读取：config-lite

Mongodb驱动： Mongolass

根据objectid生成时间戳：objectid-to-timestamp

sha1加密：Sha1

日志：Winston

基于winston用于express的日志中间件

数据库：mongodb

}

1. 目录结构

Myblog

Models：存放操作数据库的文件

Public：存放静态文件如样式图片等

Routes：存放路由文件

Views：存放视图文件

App.Js：应用的主文件

Package.Json：存储项目名，描述，作者，依赖等信息

1. 编写入口文件
2. 引入依赖模块并实例化express
3. 设置视图路径
4. 设置视图引擎
5. 设置静态资源路径
6. 监听端口
7. 配置路由

Mongodb windows开机自启mongod --dbpath D:\MongoDB\data --logpath=D:\MongoDB\logs\mongodb.log –install

Net start mongodb

可视化mongodb软件 robomongo

QA(质量保证)

1. 页面测试 测试页面表示和前端功能 使用Mocha
2. 跨页测试 设计多个组件，集成测试 使用 Zombie.js，无头浏览器 Selenium,PhantomJS,Zombie
3. 逻辑测试 对业务逻辑进行单元测试和集成测试。只测js，与表示功能分开

mocha -u tdd -R spec qa/tests-unit.js 全局安装mocha后运行测试文件的代码，spec：一种错误报告，详细些

mocha -u tdd -R spec qa/tests-crosspage.js 2>/dev/null 不显示调用堆栈跟踪

1. 去毛 找可能出错的代码 使用JSHint，jsLint
2. 链接检查 单元测试 使用LinkChecker

浏览器发来的请求头的相关信息

Req：**请求对象，下面是express为其添加的一些方法和属性**

req.params •

一个数组，包含命名过的路由参数。

req.param(name) •

返回命名的路由参数，或者 GET 请求或 POST 请求参数。建议你忽略此方法。

req.query •

一个对象，包含以键值对存放的查询字符串参数（通常称为 GET 请求参数）。

req.body •

一个对象，包含 POST 请求参数。这样命名是因为 POST 请求参数在 REQUEST 正文中传

递，而不像查询字符串在 URL 中传递。要使 req.body 可用，需要中间件能够解析请求

正文内容类型，我们将在第 10 章进行详细介绍。

req.route •

关于当前匹配路由的信息。主要用于路由调试。

req.cookies/req.singnedCookies •

一个对象，包含从客户端传递过来的 cookies 值。详见第 9 章。

req.headers •

从客户端接收到的请求报头。

Header里的几个参数

Upgrade-Insecure-Requests:1：代表浏览器支持吧http连接转换为https连接。

Referer:引用页，告诉服务器请求从什么地址来。

If-Modified-Since & If-None-Match：表示数据缓存的相关信息。

决定是否缓存的过程

1.客户端请求一个页面（A）。

2.服务器返回页面A，并在给A加上一个ETag。

3.客户端展现该页面，并将页面连同ETag一起缓存。

4.客户再次请求页面A，并将上次请求时服务器返回的ETag一起传递给服务器。

5.服务器检查该ETag，并判断出该页面自上次客户端请求之后还未被修改，直接返回响应304（未修改——Not Modified）和一个空的响应体。

req.accepts([types]) •

一个简便的方法，用来确定客户端是否接受一个或一组指定的类型（可选类型可以是

单个的 MIME 类型，如 application/json 、一个逗号分隔集合或是一个数组）。写公共

API 的人对该方法很感兴趣。假定浏览器默认始终接受 HTML。

req.ip •

客户端的 IP 地址。

req.path •

请求路径（不包含协议、主机、端口或查询字符串）。

req.host •

一个简便的方法，用来返回客户端所报告的主机名。这些信息可以伪造，所以不应该用

于安全目的。

req.xhr •

一个简便属性，如果请求由 Ajax 发起将会返回 true 。

req.protocol •

用于标识请求的协议（ http 或 https ）。

req.secure •

一个简便属性，如果连接是安全的，将返回 true 。等同于 req.protocol==='https' 。

req.url/req.originalUrl •

有点用词不当，这些属性返回了路径和查询字符串（它们不包含协议、主机或端口）。

req.url 若是出于内部路由目的，则可以重写，但是 req.orginalUrl 旨在保留原始请求

和查询字符串。

req.acceptedLanguages •

一个简便方法，用来返回客户端首选的一组（人类的）语言。这些信息是从请求报头中

解析而来的。

Res：**响应对象**

res.status(code) •

设置 HTTP 状态代码。Express 默认为 200（成功），所以你可以使用这个方法返回状态

404（页面不存在）或 500（服务器内部错误），或任何一个其他的状态码。对于重定向

（状态码 301、302、303 和 307），有一个更好的方法： redirect 。

res.set(name,value) •

设置响应头。这通常不需要手动设置。

res.cookie（name,vaue,[options]）,res.clearCookie(name,[options]) •

设置或清除客户端 cookies 值。需要中间件支持，详见第 9 章。

res.redirect([status],url) •

重定向浏览器。默认重定向代码是 302（建立）。通常，你应尽量减少重定向，除非永

久移动一个页面，这种情况应当使用代码 301（永久移动）。

res.send(body),res.send(status,body) •

向客户端发送响应及可选的状态码。Express 的默认内容类型是 text/html 。如果你想改

为 text/plain ，需要在 res.send 之前调用 res.set('Content-Type','text/plain\') 。如

果 body 是一个对象或一个数组，响应将会以 JSON 发送（内容类型需要被正确设置），

不过既然你想发送 JSON，我推荐你调用 res.json 。

res.json(json),res.json(status,json) •

向客户端发送 JSON 以及可选的状态码。

res.jsonp(json),req.jsonp(status,json) •

向客户端发送 JSONP 及可选的状态码。

res.type(type) •

一个简便的方法，用于设置 Content-Type 头信息。基本上相当于 res.set('Content-

Type','type') ，只是如果你提供了一个没有斜杠的字符串，它会试图把其当作文件的

扩展名映射为一个互联网媒体类型。比如， res.type('txt') 会将 Content-Type 设为

text/plain 。此功能在有些领域可能会有用（例如自动提供不同的多媒体文件），但是

通常应该避免使用它，以便明确设置正确的互联网媒体类型。

res.format(object) •

这个方法允许你根据接收请求报头发送不同的内容。这是它在 API 中的主要用途，我们

将会在第 15 章详细讨论。这里有一个非常简单的例子： res.format({'text/plain':'hi

there','text/html':'<b>hi there</b>'}) 。

res.attachment([filename]),res.download(path,[filename],[callback]) •

这两种方法会将响应报头 Content-Disposition 设为 attachment ，这样浏览器就会选

择下载而不是展现内容。你可以指定 filename 给浏览器作为对用户的提示。用 res.

download 可以指定要下载的文件，而 res.attachment 只是设置报头。另外，你还要将

内容发送到客户端。

res.sendFile(path,[option],[callback]) •

这个方法可根据路径读取指定文件并将内容发送到客户端。使用该方法很方便。使用静

态中间件，并将发送到客户端的文件放在公共目录下，这很容易。然而，如果你想根据

条件在相同的 URL 下提供不同的资源，这个方法可以派上用场。

res.links(links) •

设置链接响应报头。这是一个专用的报头，在大多数应用程序中几乎没有用处。

res.locals,res.render(view,[locals],callback) •

res.locals 是一个对象，包含用于渲染视图的默认上下文。 res.render 使用配置的模

请求和响应对象 ｜ 55

板引擎渲染视图（不能把 res.render 的 locals 参数与 res.locals 混为一谈，上下文

在 res.locals 中会被重写，但在没有被重写的情况下仍然可用）。 res.render 的默认响

应代码为 200，使用 res.status 可以指定一个不同的代码。