# 应用程序级别的中间件

//中间件：处理请求的，本质是一个函数

//当请求进来，会从第一个中间件进行匹配

//   如果匹配，则进来

//      如果请求进入中间件之后，没有调用next（）则代码会停止在当前的中间件中

//   如果不匹配则继续判断匹配下一个中间件

同一个请求所经过的中间件都是同一个请求对象和响应对象

<http://expressjs.com/en/guide/using-middleware.html>

## 万能匹配

//1.不关心请求路径和请求方法的中间件，任何请求都会进入这个中间件

// 中间件本身是一个方法，该方法接收三个参数

//request  请求对象

//respone  响应对象

//next     下一个中间件 这是一个方法  调用下一个中间件

app.use(function(req,res,next){

    console.log('1'); //输出/a/以后的路径

    next()

})

## 只以‘/xx’开头

//2.关心请求路径的中间件

app.use('/a',function(req,res,next){

    console.log(req.url); //输出/a/以后的路径

})

# 路由级别中间件

app.get('/',function(req,res,next){

    fs.readFile('./df/df/f',function(err,data){

       if(err){

           Return throw err

       }

next() //成功进入下一个匹配的中间件

    })

})

# 错误处理中间件

app.get('/',function(req,res,next){

    fs.readFile('./df/df/f',function(err,data){

       if(err){

           //直接找带有4个参数的中间件,进入执行

           //不会进入别的也匹配的中间件中 ,不会进入404

           next(err)

       }

    })

})

app.get('/',function(req,res,next){

    console.log('ddd')

})

app.use(function(req,res,next){

    console.log('404')

})

app.use(function(err,req,res,next){

    console.log('error')

    res.status(500).send(err.message)

})

# 第三方中间件

Body-parser post请求

Session 存在数据

Compression

Cookie-parser

Morgan

Response-time

Serve-static