1、页面渲染传值：

方式一：通过渲染方法传值： res.render('gk',{ datas: '我去，竟然成功了哦！' });

渲染参数说明：**render('a',{ b})**

参数一 **a** 、要响应渲染的页面名（在**views**中有定义的**.hbs**页面）

参数二 **b** 、

可以有多个传值参数，多个参数用逗号（，）隔开 如：**{ b：111，c:[{1,2,3}]}**

在**.hbs**页面中通过插值获取 **{{b}}**

注：也可以是模板页 **{ layout: 'layout2' }** 指定**views**里的模板名，

如：**res.render('renders',{ layout: 'layout2' });**

方式二：通res.locals.数值变量名 = 'xxxxxx'; 在**.hbs**页面中通过插值获取 **{{b}}**

2、模板页： **layout.hbs** 对多个页面有公共部分的内容用模板页来定义，在渲染页面时指 定模板即可。默认不指定就是**layout.hbs**

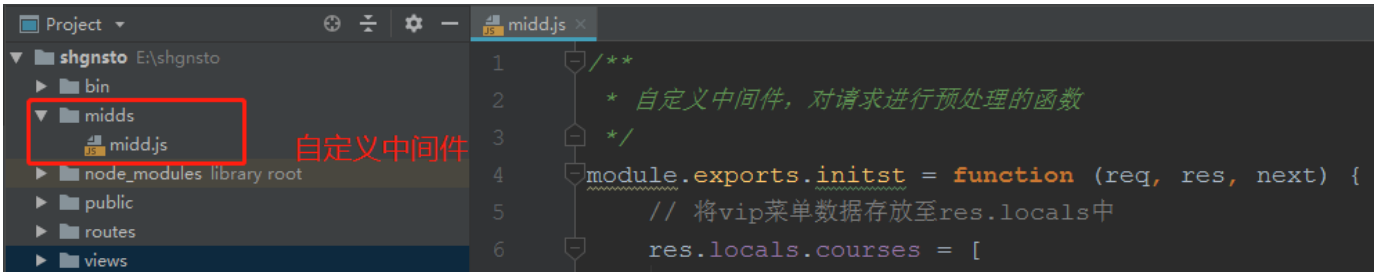
设置定义方法：在**layout.hbs**的**<body>**内添加公共的**html**即可

```
<body>
    .....html.....
</body>
```

3、中间件：就是对请求，响应对象的预处理函数，相当于过滤器，主要用来处理一些全局参数

中间件的产出步骤：

创建**JS**文件：在项目根目录创建一个新的目录用来保存所有的中间件**js**文件



编写自定义中间件：

```
module.exports.自定义函数名 = function (req, res, next) {
    res.locals.自定义变量名 = [
      {
          数据一
      },
      {
          数据二
      },
    ];
    next();// 进入后续中间件
}
```

```
/**
 * 自定义中间件，对请求进行预处理的函数
 */
module.exports.initst = function (req, res, next) {
    // 将vip菜单数据存放至res.locals中
    res.locals.courses = [
        {name: 'WEB全栈架构师'...},
        {name: 'python爬虫'...},
    ];
    next();// 进入后续中间件
}
```

注册：在**app.js**中操作

1、导出自定义中间件

const {自定义变量名} = require('./中间件目录名/中间件JS文件名');

如：const {initst} = require('./midds/midd');

```
// 导出自定义中间件
const {initst} = require('./midds/midd');
/**
 * 配置访问的地址   导入路由相关模块
 */
var indexRouter = require('./routes/index');
```

2、注册

app.use(自定义变量名);　　如：app.use(initst);

```
app.use(cookieParser());
//设置静态目录
app.use(express.static(path.join(__dirname, 'public')));

// 注册自定义中间件
app.use(initst);

/**
 * 配置访问的地址     路由相关模块的指向
 */
app.use('/', indexRouter);    //首页
```

必须要注册路由之间完成中间件的注册

3、使用:在views中的hbs页面中使用，通过一些方法获取中间件里提前定好的的值

以在layout中使用为例：

hbs的循环语句，对中间件的数据进行处理输出

{{#each 中间件函数中自定义的变量名}}

{{数值变量名}}

<img src="{{数值变量名}}">

{{/each}}

```
<div class="right-view">
    <a class="menu-item" href="/vip">VIP课程+</a>
    {{!hbs的循环语句，对中间件的数据进行处理输出}}
    {{#each courses}}
        <a href="{{url}}">{{name}}</a>
        <img src="{{icon}}">
    {{/each}}
    <a class="menu-item" href="/gkk">公开课</a>
 <!--    <a class="menu-item" href="/student">学习中心</a>    -->
</div>
```

4、获取请求的地址及请求的参数值

1、获取请求的地址级，如：**http://localhost:3000/vip/web** 当用户访问

的是**web**时

不必在**app.js**中再定义路径地址，可以通过下在路由文件中使用占位符处理

router.get('/:cxxx', function(req, res, next)　**:cxxx** 就是地址占位符

通过 req.params.cxxx就可以获取到当前用户请求的地址是web

然后就可以进行渲染页面：res.render('vip/' + req.params.cxxx)

此时还需要在**views**中定义对应的**hbs**的页面

2、获取请求参数：

get请求：**req.query.**参数名

post请求:**req.body.**参数名

```
/* GET  */
router.get('/', function(req, res, next) {
    //get请求获取请求参数
    var resqs = {
        "用户名":req.query.u,      //处理get请求数据
        "密码":req.query.p
    };
    console.log(resqs);
    res.send(JSON.stringify(resqs));
});
/* POST  */
router.post('/', function(req, res, next) {
    // 输出 JSON 格式
    var resqs = {
        "用户名":req.body.user,
        "密码":req.body.pass
    };
    console.log(resqs);
    res.send(JSON.stringify(resqs));
```

5、错误处理：

1、进入错误**error.hbs**页面

//next(new Error('没有您要的课程'));//状态码500

// next(createError(404, '没有您要的课程'));//状态码404

2、重定向

res.redirect('绝对路径地址**')** 如：**res.redirect('/vip/web')**

6、模板引擎**Handlebars**语法：模板引擎**Handlebars**语法**.hbs**

插值绑定 {{prop}}

注释 {{! content}} HTML内容 {{{htmlStr}}}

条件语句 {{#if condition}}...{{/if}} ** condition只能是布尔值或者可以转换为布尔值的值，他不能是表达式 ** 可以结合{{else if condition}}、{{else}}使用

循环语句 {{#each arr}}....{{/each}} ** each可嵌套 ** 使用this或者.表示上下文，常用语数据是值的情况 ** 使用@index，@key ** 遍历对象 @key ** 结合{{else}}，当数组为空时显示特别信息