

## 课程目标

- 掌握Koa基本用法
- 理解Koa设计思路
- 路由
- 静态文件服务
- 模板引擎

## koa

- 安装: `npm i koa -S`
- 中间件机制、请求、响应处理

```
const Koa = require("koa");
const app = new Koa();

// 响应时间输出中间件
app.use(async (ctx, next) => {
  await next();
  // 获取响应头, 印证执行顺序
  const rt = ctx.response.get('X-Response-Time');
  console.log(`输出计时: ${ctx.method} ${ctx.url} - ${rt}`);
});

// 响应时间统计中间件
app.use(async (ctx, next) => {
  const start = Date.now();
  console.log('开始计时');
  await next();
  const ms = Date.now() - start;
  ctx.set('X-Response-Time', `${ms}ms`);
  console.log('计时结束');
});

// 响应
app.use(ctx => {
  console.log('响应用户请求');
  ctx.status = 200; // 设置响应状态码
  ctx.type = 'html'; // 设置响应类型, 等效于ctx.set('Content-Type', 'text/html')
  ctx.body = "<h1>Hello Koa</h1>"; //设置响应体
});

//开始监听端口, 等同于http.createServer(app.callback()).listen(3000);
app.listen(3000);
```

- 错误处理

```

const Koa = require("koa");
const app = new Koa();

//错误处理中间件
app.use(async (ctx, next) => {
  try {
    await next();
  } catch (error) {
    // 响应用户
    ctx.status = error.statusCode || error.status || 500;
    ctx.body = error.message;

    // 触发应用层级错误事件
    ctx.app.emit("error", error, ctx);

    console.log('捕获到错误: ', error.message);
  }
});

// 响应时间统计
//...

// 触发错误
app.use(async (ctx, next) => {
  // throw new Error('未知错误');
  ctx.throw(401, '认证失败')
});

// 响应
//...

//全局错误事件
app.on("error", err => {
  console.error("全局错误处理: ", err.message);
});

//开始监听端口, 等同于http.createServer(app.callback()).listen(3000);
app.listen(3000);

```

- 路由: `npm i -S koa-router`

```

// ./routes/index.js
const Router = require("koa-router");
const router = new Router();
router.get("/", ctx => {
  ctx.body = "index";
});
module.exports = router;

// ./routes/users.js
const Router = require("koa-router");

```

```
const router = new Router({prefix: '/users'});
router.get("/", ctx => {
  ctx.body = "users list";
});
module.exports = router;

// ./app.js
// ...其他中间件
const index = require('./routes/index');
const users = require('./routes/users');

app.use(index.routes());
app.use(users.routes());
```

- 静态文件服务: `npm i -S koa-static`

```
// app.js
const static = require("koa-static");
app.use(static(__dirname + '/public')); // 放错误处理中间件后面
```

- 模板引擎: `npm i koa-hbs@next -S`

- 引入并配置, app.js

```
const hbs = require('koa-hbs')
app.use(hbs.middleware({
  viewPath: __dirname + '/views', //视图根目录
  defaultLayout: 'layout', //默认布局页面
  partialsPath: __dirname + '/views/partials', //注册partial目录
  disableCache: true //开发阶段不缓存
}));
```

创建views,views/partials,layout.hbs,index.hbs layout.hbs:

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Document</title>
</head>
<body>
  {{{body}}}
</body>
</html>
```

渲染, ./routes/index.js

```
router.get("/", async (ctx) => {
  await ctx.render('index')
});
```

- handlebars基本使用:

传值, routes/users.js

```
router.get("/", async ctx => {
  await ctx.render("users", {
    title: "用户列表",
    subTitle: "handlebars语法",
    isShow: true,
    username: "jerry",
    users: [{ username: "tom", age: 20 }, { username: "jerry", age: 20 }]
  });
});
```

显示, views/users.hbs

```
{{!-- 1.插值绑定 --}}
<h1>{{subTitle}}</h1>

{{!-- 2.注释 --}}

{{!-- 3.HTML内容 --}}
<p>{{htmlStr}}</p>

{{!-- 4.条件语句 --}}
{{#if isShow}}
<p>{{username}}, 欢迎你! </p>
{{else}}
<a href>请登录</a>
{{/if}}

{{!-- 5.循环 --}}
<ul>
  {{#each users}}
    <li>{{username}} - {{age}}</li>
  {{/each}}
</ul>
```

- 部分视图: 提取通用内容至独立文件

- 创建./views/partials/nav.hbs
- 引用, ./views/layout.hbs

```
{{>nav}}
```

- 帮助方法: 扩展handlebars的功能函数

创建./utils/helpers.js

```
const hbs = require("koa-hbs");
const moment = require("moment");

hbs.registerHelper("date", (date, pattern) => {
  try {
    return moment(date).format(pattern);
  } catch (error) {
    return "";
  }
});
```

补充数据, routes/users.js

```
users: [
  { username: "tom", age: 20, birth: new Date(1999, 2, 2) },
  { username: "jerry", age: 20, birth: new Date(1999, 3, 2) }
]
```

调用helper, views/users.hbs

```
<li>{{username}} - {{age}} - {{date birth 'YYYY/MM/DD'}}</li>
```

N多帮助方法<https://github.com/helpers/handlebars-helpers>, 用法:

```
const helpers = require('handlebars-helpers');
helpers.comparison({ handlebars: hbs.handlebars });

{{#and a b}}a, b都是true{{/and}}
```

#### ◦ 高级应用: 代码搬家

定义代码块, views/users.hbs

```
{{#contentFor 'jquery'}}
<script>
  $(function(){
    console.log('content for jquery')
  })
</script>
{{/contentFor}}
```

代码搬家, views/layout.hbs

```
<script src="https://cdn.bootcss.com/jquery/3.3.1/jquery.min.js"></script>
{{#block 'jquery'}}{{/block}}
```

#### • 案例: 开课吧首页

- mongoose连接, ./models/mongoose.js

```
const mongoose = require("mongoose");

// 1.连接
mongoose.connect("mongodb://localhost:27017/test", { useNewUrlParser: true });

const conn = mongoose.connection;
conn.on("error", () => console.error("连接数据库失败"));
conn.once("open", () => console.log("连接数据库成功"));
```

- o vip课程数据准备, ./models/vip.js

```
const mongoose = require("mongoose");

const schema = mongoose.Schema({
  name: String,
  url: String,
  poster: String,
  icon: String,
  description: String,
  cooperation: [String]
});

const model = mongoose.model("vip", schema);

// 测试数据
async function testdata() {
  await model.deleteMany();
  await model.insertMany([
    {
      name: "Web全栈架构师",
      url: "/vip-course/web",
      poster: "https://img.kaikeba.com/web_vip.png",
      icon: "https://img.kaikeba.com/web_menu.png",
      description: "授课深度对标百度T6-T7, 进入BAT等一线大厂, 实现年薪30万+",
      cooperation: [
        "https://img.kaikeba.com/baidu.png",
        "https://img.kaikeba.com/toutiao.png"
      ]
    },
    {
      name: "Python爬虫商业项目班",
      url: "/vip-course/python",
      poster: "https://img.kaikeba.com/python_vip_new.png",
      icon: "https://img.kaikeba.com/python_menu.png",
      description: "廖雪峰亲自传授如何用python, 让天下没有爬不到的数据",
      cooperation: [
        "https://img.kaikeba.com/baidu.png",
        "https://img.kaikeba.com/toutiao.png"
      ]
    },
    {
      name: "百万年薪架构师",
```

```

url: "/vipcourse/javaMillion",
poster: "https://img.kaikeba.com/javaMillion_vip.png",
icon: "https://img.kaikeba.com/javaMillion_menu.png",
description: "人工智能时代，互联网高可用高并发的架构核心技术深度揭秘",
cooperation: [
  "https://img.kaikeba.com/baidu.png",
  "https://img.kaikeba.com/toutiao.png"
]
},
{
  name: "人工智能特训营",
  url: "/vip-course/ai",
  poster: "https://img.kaikeba.com/robot_vip_new.png",
  icon: "https://img.kaikeba.com/robot_menu.png",
  description: "课程研发结合BAT企业用人需求，基于14个商业项目案例深度教学",
  cooperation: [
    "https://img.kaikeba.com/baidu.png",
    "https://img.kaikeba.com/toutiao.png"
  ]
},
{
  name: "UXD全栈设计师",
  url: "/vip-course/uxd",
  poster: "https://img.kaikeba.com/uxd_vip.png",
  icon: "https://img.kaikeba.com/uxd_menu.png",
  description: "深度系统地学习，华丽变身UXD全栈设计师，开创“薪”时代",
  cooperation: [
    "https://img.kaikeba.com/baidu.png",
    "https://img.kaikeba.com/toutiao.png"
  ]
},
{
  name: "JavaEE 企业级分布式高级架构师",
  url: "/vipcourse/java",
  poster: "https://img.kaikeba.com/java_vip.png",
  icon: "https://img.kaikeba.com/java_menu.png",
  description:
    "教研团队与一线企业技术leader深度合作，以企业所需，重新定义Java进阶课程",
  cooperation: [
    "https://img.kaikeba.com/baidu.png",
    "https://img.kaikeba.com/toutiao.png"
  ]
}
]);

testdata();

module.exports = model;

```

- o vip课程查询中间件，./middleware/get-vip.js

```
const vip = require("../models/vip");

module.exports = async (ctx, next) => {
  if (ctx.accepts("html") === 'html') {
    ctx.state.vipCourses = await vip.find();
  }

  await next();
};
```

- 引入中间件, app.js

```
const mongoose = require('./models/mongoose')
const getVip = require('./middleware/get-vip')

//... static
app.use(getVip)
```

- 内容渲染: 导航, ./views/partials/nav.hbs, [素材](#)
- 样式: ./public/styles/index.css, [素材](#)
- 课程列表, ./views/index.hbs, [素材](#)
- 课程排序调整
- 显示视频与否
- 缓存查询结果