koa

1. cd 到某一个文件夹
2. npm init　　回车 ,把项目初始化完成
3. 根据package.json 文件　　在项目中创建一个新的入口文件　　默认叫 index.js
4. 安装 服务器框架 koa2
   - cnpm install koa --save
   - --save : 将安装的框架koa 保存到 package.json文件中
5. 编辑index.js 代码
   - 如果不会写,可以去koa官网上找到简单的服务器代码

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

app.use(async ctx => {
  ctx.body = "Hello world"
});
console.log("服务器运行在3000端口,...")
app.listen(3000)
```

6. 运行 node index.js

7. 让服务器具有静态资源访问的功能
   - 7.1 在 项目中 创建 static 文件夹
   - 7.2 在服务器中 安装 koa静态资源框架　　cnpm install koa-static --save
   - 7.3 通过 koa-static 框架 配置静态资源文件夹
     - 7.3.1 导入 koa-static 模块　　`const static = require('koa-static')`
     - 7.3.2 app 使用 koa-static 模块　　`app.use(static(__dirname + "/static"))`

8. 让 koa 服务器具有路由功能
   - 8.1 安装 koa-router 模块　　cnpm install koa-router --save
   - 8.2 导入 koa-router 模块　　`const router = require('koa-router')();`
   - 8.3 app 使用 koa-router 模块　　`app.use(router.routes());`
   - 8.4 给自己的app服务器 编写不同的路由地址

```
router.get('/', ctx => {
    ctx.response.body = 'This is homepage';
});
.get('/xiaomi', ctx => {
    ctx.response.body = "我是小米界面"
})
.get('/iphone', ctx => {
    ctx.response.body = "我是苹果界面"
});
```

   - 8.5 把路由模块拆分到另外一个 luyou.js 文件中
     - 8.5.1 把导入路由的代码 从 启动文件 index.js 中拷贝到luyou.js文件中　　同时 在 index.js 中删除 `const router = require('koa-router')();`
     - 8.5.2 把 路由 路径的配置代码拷贝到luyou.js文件中　　同时 在 index.js 中删除

```
router.get('/', ctx => {
    ctx.response.body = 'This is homepage';
});
.get('/xiaomi', ctx => {
    ctx.response.body = "我是小米界面"
});
```

     - 8.5.3 在 luyou.js 文件最后 将 路由模块 router 进行导出, 方便别人导入　　`module.exports = router`
     - 8.5.4 在 app 的 index.js 文件中把 app.use 路由 中间件的代码给修改掉　　`// app.use(router.routes())` `app.use(require("./luyou.js").routes())`

9. koa服务器 接口参数解析　　解析地址参数, 使用 koa-bodyparser 模块
   - 9.1 安装 koa-bodyparser　　cnpm install koa-bodyparser --save
   - 9.3 使用 bodyparser 中间件　　app.use(bodyparser())
   - 9.4 GET 方式传过来的参数, 放在 ctx.request.query 中
   - 9.2 导入 koa-bodyparser　　`const bodyparser = require('koa-bodyparser')`
   - 9.5 POST 方式传过来的参数, 放在 ctx.request.body 中

10. mysql
    - 10.1 详见数据库搭建和环境使用步骤
    - mysql 模块
      - 10.1 安装　　cnpm install mysql --save
      - 10.2 导入 mysql　　`const mysql = require('mysql');`
      - 10.3 配置mysql的链接参数

```
var connection = mysql.createConnection({
  host     : '127.0.0.1',
  user     : 'root',
  password : '1234567',
  database : 'school'
});
```

      - 10.4 启动链接　　`connection.connect();`
      - 10.5 执行sql语句

```
connection.query('select * from info', function (error, results, fields) {
  if (error) {
    throw error;
  }
  console.log('结果是: ', results);
});
```

      - 10.6 关闭链接　　`connection.end();`

11. 登录注册功能　　综合上面的步骤用ajax 实现登录注册功能

12. 文件上传
    - 12.1 前端页面 创建form表单　　enctype="mutipart/form-data"
    - 12.2 服务器的支持文件上传,并且配置文件上传大小
      - 12.2.1 安装koa-body模块　　cnpm install koa-body --save
      - 12.2.2 导入并配置该模块

```
const koa_body = require('koa-body')
app.use(koa_body({
  multipart: true,
  formidable: {
    maxFileSize: 1024 * 1024 * 200
  }
}))
```

      - 12.2.3 路由地址中,接受到上传的文件并保存
        - 12.2.3.1 获取到接受的文件 var file = ctx.request.files.file
        - 12.2.3.2 导入文件操作系统模块fs, 并读取 file 文件中的数据流　　`const fs = require('fs')` `var read_stream = fs.createReadStream(file.path)`
        - 12.2.3.3 根据fs模块创建写入数据流, 并指向地址　　`var write_stream = fs.createWriteStream("xxx.jpg")`
        - 12.2.3.4 通过管道pipe 将 read_stream 中的数据 流入 到 write_stream 中　　`read_stream.pipe(write_stream)`
      - 12.2.4 form 表单的文件上传, 跳转页面, 是有form表单的默认事件触发的
        1. 页面跳转了
        2. 没办法在上传页面获取到上传结果信息
        3. 解决办法, 阻止form的默认提交事件, event.preventDefault();　　必然导致, 数据也不提交了, 自己写 ajax 进行文件上传, 算了,别写了 还是找插件吧
    - 12.3 前端插件 jquery.form.js 做文件上传
      1. 导入jquery
      2. 导入 jquery.form.js
      3. 监听form表单的提交事件　　`$('你的form表单').on("submit" , function () { })`
      4. 在监听到submit事件的回调函数中写代码
         - 阻止该提交的默认事件　　event.preventDefault();
         - 调用插件的上传方法　　`$("form表单").ajaxSubmit(()})`

13. svn使用
    1. 拥有自己的svn账号
    2. 项目创建这在svn平台上创建 项目
    3. 把所有开发者拉入到项目
    4. 带有?的文件, 需要先add 在commit　　(除了 node_modules模块)
    5. 带有 M 的文件, 直接commit
    6. 带有 C 的文件, 先找人解决冲突, 然后commit
    7. 带有 D 的文件, 直接commit
    8. commit 之前 一定要 先 update

14. mongoDB数据库使用
    - 14.1 详见 mongoldb数据库的使用详情
    - koa中mongo模块的使用
      1. 安装 mongoodb模块　　cnpm install mongodb --save

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";
MongoClient.connect(url, { useNewUrlParser: true }, function (err, db) {
    if (err) throw err;
    console.log("数据库已创建");
    var dbase = db.db("数据库名");
    dbase.createCollection('site', function (err, res) {
        if (err) throw err;
        console.log("创建集合!");
        db.close();
    });
});
```

      2. 连接并使用mongodb

15. 导入socket 及时通讯功能
    - 0. net 模块搭建socket

```
var net = require('net');

var a = net.createServer(function(sock) {

    sock.on('data', function(shuju) {
        sock.write('我收到你的数据了: "' + shuju + "\n");
    });
    sock.on('error', function(exception) {
        console.log('服务端错误socket error:' + exception);
        sock.end();
    });

    sock.on('close', function(data) {
        console.log('服务端CLOSED: ');
    });

});
a.listen(8088, '127.0.0.1');
```

```
var net = require('net');
var client = new net.Socket();
//客户端连接
client.connect(8088, '127.0.0.1', function() {
    //客户端向服务端socket发送数据
    client.write('我是客户端 刘伟');
});

client.on('data', function(data) {
    console.log("从服务器接受的数据:" + data + "      \n");
});

// 为客户端添加“close”事件处理函数
client.on('close', function() {
    console.log('客户端: 监听到到服务器 close 动作了...');
})
```

    - 1. 安装 ws 模块　　cnpm install ws --save

```
var WebSocketServer = require('ws').Server;
var server_obj = new WebSocketServer({ port: 8088 });

// 监听客户端链接
server_obj.on('connection', function(client) {
    // 监听客户端消息事件
    client.on('message', function(msg) {

    });
    // 监听错误事件
    client.on('error', function(data) {

    });
    // 监听客户端关闭事件
    client.on('close', function(data) {

    });
});
```

    - 2. 前端网页脚本中实现客户端代码

```
var my_ws = new WebSocket('ws://127.0.0.1:8088')
// 链接服务器事件
my_ws.onopen = function() {
    console.log("客户端: 我连接上8088服务器了")
    my_ws.send("hello server")
}
// 消息监听事件
my_ws.onmessage = function(data) {

}
// 错误监听事件
my_ws.onerror = function(data) {

}
// 关闭监听事件
my_ws.onclose = function(data) {

}
```