00:00 -05:02

Mongoose 连接两座孤岛之间的桥梁

到目前为止，我们已经跑起来了一个很简单的 Express 应用，并且 MongoDB 数据库也运行起来了，但是 Express 应用和 MongoDB 数据库之间没有任何关联，犹如茫茫大海中的两座孤岛。本节课程将介绍另外一位小伙伴 Mongoose，它将是连接 Express 应用和 MongoDB 数据库之间的桥梁，让它们二者能互通信息。

安装 mongoose 包

```
npm install --save mongoose
```

mongoose

使用 Mongoose

打开 `index.js` 文件，首先导入 mongoose 中间件：

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

然后调用 Mongoose#connect 方法连接本案例使用的 `react-hand-in-hand` 数据库：

```
mongoose.connect('mongodb://localhost:27017/r
```

传递给 `mongoose.connect` 方法的第一个参数是 MongoDB 数据库对应的连接字符串 URI，简单分析一下本案例数据库的 URI 字符串，它由三部分组成：

- mongodb:// 代表使用的协议
- localhost:27017 代表要连接的服务器地址，一个 MongoDB 服务器实例运行在本地 27017 端口
- react-hand-in-hand 代表要连接的数据库名字

`mongoose.connect` 语句执行之后，我们并不能确定 Mongoose 成功连接上了 MongodDB 数据库，还得添加以下检验代码：

```
var db = mongoose.connection;
db.on('error', function(err){
  console.log('connection failed!', err);
});
db.once('open', function() {
  console.log('success!')
});
```

调用 Mongoose#connection 方法，得到一个 Connection 的对象实例 `db`， `Connection` 继承了 Node.js 的EventEmitter 类的实例方法 on 和 once。

若连接数据库没有成功，则会触发 `error` 事件，随即执行 `error` 事件监听函数，在命令行中会报告错误信息：

```
function(err){
  console.log('connection failed!', err);
}
```

若连接数据库成功了，则会触发 `open` 事件，随后执行一次 `open` 事件监听函数，在命令行中打印 `success!` 字样：

```
db.once('open', function() {
  console.log('success!')
});
```

修改配置文件

最好还是把 `react-hand-in-hand` 数据库的 URI 字符串放到配置文件中，避免开发环境和生产环境使用的 MongoDB 数据库名字不一致，打开 `config.js` 文件，添加一行代码：

```
module.exports = {
  ...
  uri: 'mongodb://localhost:27017/react-hand-
};
```

相应地就得修改 `index.js` 文件中的代码，导入配置参数 `uri`：

```
var uri = require('./config.js').uri
```

接下来，修改传递给 `mongoose.connect` 方法的第一个参数，用 `uri` 替换掉那一长串字符：

```
mongoose.connect(uri);
```

然后在 `config.default.js` 文件中也添加一条记录：

```
module.exports = {
  ...
  url: 'mongodb://localhost:27017/xxx'
};
```