Koa实战 - Restful API

课程目标

- 掌握Koa中编写Restful风格API
- 掌握Koa中文件上传、表单验证、图形验证码、发送短信等常见任务

编写RESTful API

- Representational State Transfer翻译过来是"表现层状态转化"，它是一种互联网软件的架构原则。因此复合REST风格的Web API设计，就称它为RESTful API
- RESTful特征：
  - 每一个URI代表一种资源(Resources)，比如：`http://kaikeba.com/courses`；
  - 客户端和服务器之间，传递这种资源的某种表现层，比如：`http://kaikeba.com/courses/web`；
  - 客户端通过HTTP动词，对服务器端资源进行操作，实现"表现层状态转化"，比如：`POST http://kaikeba.com/courses`
- URL设计
  - HTTP动词：表示一个动作
    - GET：读取（Read）
    - POST：新建（Create）
    - PUT：更新（Update）
    - PATCH：更新（Update），部分更新
    - DELETE：删除（Delete）
  - 宾语：表示动作的目标对象
    - 是一个名词

```
// 推荐
GET /users
// 不推荐
GET /getUsers
```

    - 通常是复数

```
// 推荐
GET /users
GET /users/1
// 不推荐
GET /user
GET /user/1
```

    - 避免多级

```
// 推荐
GET /authors/12?categories=2
// 不推荐
GET /authors/12/categories/2
```

- 状态码
  - 状态码要精确：
    - `1xx`：相关信息
    - `2xx`：操作成功
    - `3xx`：重定向
    - `4xx`：客户端错误
    - `5xx`：服务器错误
- 服务器响应
  - 返回JSON

```
//客户端请求
GET /users/1 HTTP/1.1
Accept: application/json

//服务端响应
HTTP/1.1 200 OK
Content-Type: application/json

{
  "ok": 1,
  "data": {"name":"tom"}
}
```

  - 错误时不要返回200状态码

```
//不推荐
HTTP/1.1 200 OK
Content-Type: application/json

{
  "ok":0,
  "data": { "error": "期待2个参数，实际收到1个。" }
}
```

```
//推荐
HTTP/1.1 400 Bad Request
Content-Type: application/json

{
  "error": "不合法的附件",
  "detail": { "uname": "用户名为必填项" }
}
```

- 范例：用户信息管理api实现

```
const users = [{ id: 1, name: "tom" }, { id: 2, name: "jerry" }];
router.get("/", ctx => {
  console.log("GET /users");
  const { name } = ctx.query; // ?name=xx
  let data = users;
  if (name) {
    data = users.filter(u => u.name === name);
  }
  ctx.body = { ok: 1, data };
});
router.get("/:id", ctx => {
  console.log("GET /users/:id");
  const { id } = ctx.params; // /users/1
  const data = users.find(u => u.id == id);
  ctx.body = { ok: 1, data };
});
router.post("/", ctx => {
  console.log("POST /users");
  const { body: user } = ctx.request; // 请求body
  user.id = users.length + 1;
  users.push(user);
  ctx.body = { ok: 1 };
});
router.put("/", ctx => {
  console.log("PUT /users");
  const { body: user } = ctx.request; // 请求body
  const idx = users.findIndex(u => u.id == user.id);
  if (idx > -1) {
    users[idx] = user;
  }
  ctx.body = { ok: 1 };
});
router.delete("/:id", ctx => {
  console.log("DELETE /users/:id");
  const { id } = ctx.params; // /users/1
  const idx = users.findIndex(u => u.id == id);
  if (idx > -1) {
    users.splice(idx, 1);
  }
  ctx.body = { ok: 1 };
});
```

- 解决跨域：`npm i koa2-cors`

```
var Koa = require('koa');
var cors = require('koa2-cors');

var app = new Koa();
app.use(cors());
```

> 参考文档：理解RESTful架构、RESTful API 最佳实践

文件上传

- 安装koa-multer：`npm i koa-multer -S`
- 配置：./routes/users.js

```
const upload = require("koa-multer")({ dest: "./public/images" });
router.post("/upload", upload.single("file"), ctx => {
  console.log(ctx.req.file); // 注意数据存储在原始请求中
  console.log(ctx.req.body); // 注意数据存储在原始请求中
  ctx.body = "上传成功";
});
```

- 调用接口，./public/upload-avatar.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta http-equiv="X-UA-Compatible" content="ie=edge" />
    <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
    <script src="https://unpkg.com/element-ui/lib/index.js"></script>
    <link
      rel="stylesheet"
      href="https://unpkg.com/element-ui/lib/theme-chalk/index.css"
    />
    <style>
      .avatar-uploader .el-upload {
        border: 1px dashed #d9d9d9;
        border-radius: 6px;
        cursor: pointer;
        position: relative;
        overflow: hidden;
      }
      .avatar-uploader .el-upload:hover {
        border-color: #409eff;
      }
```

```
      .avatar-uploader-icon {
        font-size: 28px;
        color: #8c939d;
        width: 178px;
        height: 178px;
        line-height: 178px;
        text-align: center;
      }
      .avatar {
        width: 178px;
        height: 178px;
        display: block;
      }
    </style>
    <title>文件上传</title>
  </head>
  <body>
    <div id="app">
      <!-- ajax方式上传 -->
      <el-upload
        class="avatar-uploader"
        action="/users/upload"
        :show-file-list="false"
        :on-success="handleAvatarSuccess"
        :before-upload="beforeAvatarUpload"
      >
        <img v-if="imageUrl" :src="imageUrl" class="avatar" />
        <i v-else class="el-icon-plus avatar-uploader-icon"></i>
      </el-upload>
    </div>
    <script>
      var app = new Vue({
        el: "#app",
        data() {
          return {
            imageUrl: ""
          };
        },
        methods: {
          handleAvatarSuccess(res, file) {
            this.$message.success('上传头像成功')
            this.imageUrl = URL.createObjectURL(file.raw);
          },
          beforeAvatarUpload(file) {
            const isJPG = file.type === "image/jpeg";
            const isLt2M = file.size / 1024 / 1024 < 2;

            if (!isJPG) {
              this.$message.error("上传头像图片只能是 JPG 格式!");
            }
            if (!isLt2M) {
              this.$message.error("上传头像图片大小不能超过 2MB!");
            }
```

```
            return isJPG && isLt2M;
          }
        }
      });
    </script>
  </body>
</html>
```

> 可通过设置limits、fileFilter、storage等选项限制文件尺寸、格式、存储目录和文件名等。

表单校验

- 安装koa-bouncer：`npm i -S koa-bouncer`
- 配置：app.js

```
// 为koa上下文扩展一些校验方法
app.use(bouncer.middleware());
```

- 基本使用：user.js

```
router.post("/", ctx => {
  try {
    // 校验开始
    ctx
      .validateBody("uname")
      .required("要求提供用户名")
      .isString()
      .trim()
      .isLength(6, 16, "用户名长度为6~16位");

    // ctx.validateBody('email')
    //   .optional()
    //   .isString()
    //   .trim()
    //   .isEmail('非法的邮箱格式')

    ctx
      .validateBody("pwd1")
      .required("密码为必填项")
      .isString()
      .isLength(6, 16, "密码必须为6~16位字符");

    ctx
      .validateBody("pwd2")
      .required("密码确认为必填项")
      .isString()
      .eq(ctx.vals.pwd1, "两次密码不一致");

    // 校验数据库是否存在相同值
    // ctx.validateBody('uname')
    //   .check(await db.findUserByUname(ctx.vals.uname), 'Username taken')
    ctx.validateBody("uname").check("jerry", "用户名已存在");
```

```
    // 如果走到这里校验通过

    // 校验器会用净化后的值填充 `ctx.vals` 对象
    console.log(ctx.vals);

    console.log("POST /users");
    // const { body: user } = ctx.request; // 请求body
    const user = ctx.vals;
    user.id = users.length + 1;
    users.push(user);
    ctx.body = { ok: 1 };
  } catch (error) {
    if (error instanceof bouncer.ValidationError) {
      ctx.body = '校验失败: '+error.message;
      return;
    }
    throw error
  }
});
```

图形验证码

- 安装trek-captcha：`npm i trek-captcha -S`
- 使用：./routes/api.js

```
const captcha = require("trek-captcha");
router.get("/captcha", async ctx => {
  const { token, buffer } = await captcha({ size: 4 });
  ctx.body = buffer;
});
```

- 图片显示，upload-avatar.html

```
<!-- 验证码 -->
<img src="/api/captcha" id="captcha" />
<script>
    document.getElementById('captcha').onclick = function() {
      captcha.src = "/users/captcha?r=" + Date.now();
    };
</script>
```

发送短信

- 秒滴短信API
- 安装依赖：`npm i -S moment md5 axios`
- 接口编写，./routes/api.js

```
router.get("/sms", async function(ctx) {
```

```
  // 生成6位随机数字验证码
  let code = ran(6);

  // 构造参数
  const to = ctx.query.to; // 目标手机号码
  const accountSid = "3324eab4c1cd456e8cc7246176def24f"; // 账号id
  const authToken = "b1c4983e2d8e45b9806aeb0a634d79b1"; // 令牌
  const templateid = "613227680"; // 短信内容模板id
  const param = `${code},1`; // 短信参数
  const timestamp = moment().format("YYYYMMDDHHmmss");
  const sig = md5(accountSid + authToken + timestamp); // 签名

  try {
    // 发送post请求
    const resp = await axios.post(
      "https://api.miaodiyun.com/20150822/industrySMS/sendSMS",
      qs.stringify({ to, accountSid, timestamp, sig, templateid, param }),
      { headers: { "Content-Type": "application/x-www-form-urlencoded" } }
    );

    if (resp.data.respCode === "00000") {
      // 短信发送成功，存储验证码到session，过期时间1分钟
      const expires = moment()
        .add(1, "minutes")
        .toDate();
      ctx.session.smsCode = { to, code, expires };

      ctx.body = {ok:1}
    } else {
        ctx.body = {ok:0, message: resp.data.respDesc}
    }
  } catch (e) {
    ctx.body = {ok:0, message: e.message}
  }
});
```

案例：用户注册

- 前端页面，register.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta http-equiv="X-UA-Compatible" content="ie=edge" />
    <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
    <script src="https://unpkg.com/element-ui/lib/index.js"></script>
    <script src="https://unpkg.com/axios/dist/axios.min.js"></script>
    <link
      rel="stylesheet"
      href="https://unpkg.com/element-ui/lib/theme-chalk/index.css"
    />
```

```
  <style></style>
  <title>文件上传</title>
</head>
<body>
  <div id="app">
    <el-form :model="regForm" ref="regForm">
      <el-form-item>
        <el-input
          type="tel"
          v-model="regForm.phone"
          autocomplete="off"
          placeholder="手机号"
        ></el-input>
      </el-form-item>
      <el-form-item>
        <el-input
          type="text"
          v-model="regForm.captcha"
          autocomplete="off"
          placeholder="图形验证码"
        ></el-input>
        <img :src="captchaSrc" @click="getCaptcha" />
      </el-form-item>
      <el-form-item>
        <el-input
          type="text"
          v-model="regForm.code"
          autocomplete="off"
          placeholder="短信验证码"
        ></el-input>
        <el-button type="primary" @click="getSmsCode()"
          >获取短信验证码</el-button
        >
      </el-form-item>
      <el-form-item>
        <el-input
          type="password"
          v-model="regForm.password"
          autocomplete="off"
        ></el-input>
      </el-form-item>
      <el-form-item>
        <el-button type="primary" @click="submitForm()">提交</el-button>
      </el-form-item>
    </el-form>
  </div>

  <script>
    var app = new Vue({
      el: "#app",
      data() {
        return {
          regForm: {
```

```
              phone: "",
              captcha: "",
              code: "",
              password: ""
            },
            captchaSrc: "/api/captcha"
          };
        },
        methods: {
          getCaptcha() {
            this.captchaSrc = "/api/captcha?r=" + Date.now();
          },
          getSmsCode() {
            axios
              .get("/api/sms?to=" + this.regForm.phone)
              .then(res => res.data)
              .then(({ code }) => (this.regForm.code = code));
          },
          submitForm() {
            axios
              .post("/students", this.regForm)
              .then(() => alert("注册成功"))
              .catch(error => alert("注册失败:" + error.response.data.message));
          }
        }
      });
    </script>
  </body>
</html>
```

- 注册接口编写，./routes/students.js

```
const Router = require("koa-router");
const router = new Router({ prefix: "/students" });
const bouncer = require("koa-bouncer");

router.post("/", async ctx => {
  try {
    // 输入验证
    const { code, to, expires } = ctx.session.smsCode;
    ctx
      .validateBody("phone")
      .required("必须提供手机号")
      .isString()
      .trim()
      .match(/1[3-9]\d{9}/, "手机号不合法")
      .eq(to, "请填写接收短信的手机号");

    ctx
      .validateBody("code")
      .required("必须提供短信验证码")
      .isString()
      .trim()
```

```
      .isLength(6, 6, "必须是6位验证码")
      .eq(code, "验证码填写有误")
      .checkPred(() => new Date() - new Date(expires) < 0, "验证码已过期");
    ctx
      .validateBody("password")
      .required("必须提供密码")
      .isString()
      .trim()
      .match(/[a-zA-Z0-9]{6,16}/, "密码不合法");
    // 入库，略
    ctx.body = { ok: 1 };
  } catch (error) {
    if (error instanceof bouncer.ValidationError) {
      console.log(error);

      ctx.status = 401;
    } else {
      ctx.status = 500;
    }
    ctx.body = { ok: 0, message: error.message };
  }
});
module.exports = router;
```