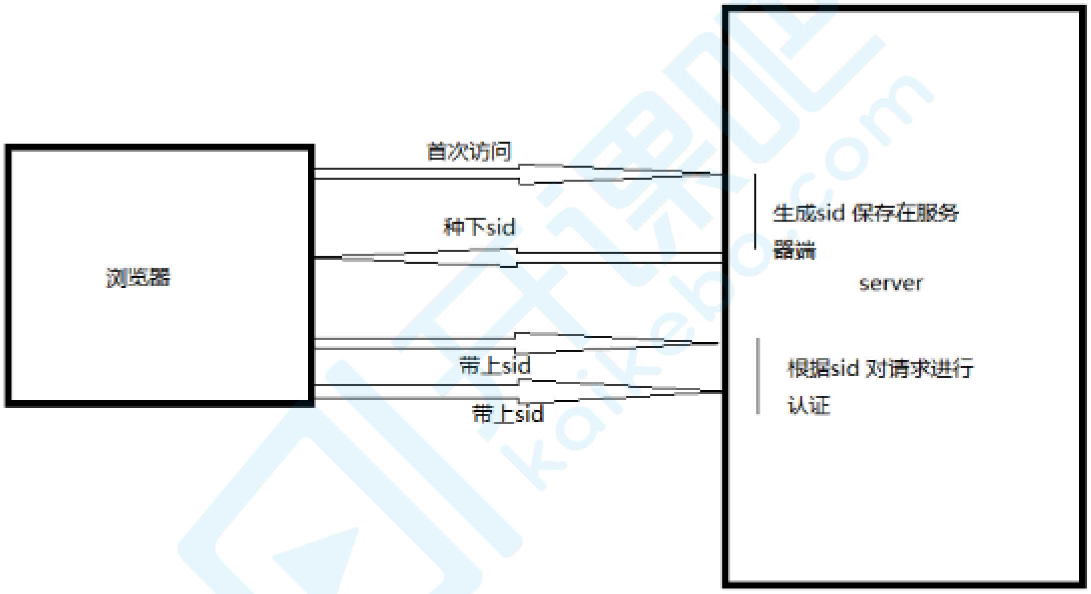
Koa实战

课程目标

- 掌握三种常见鉴权方式：Session/Cookie、Token、OAuth

session-cookie方式

- 原理



> 实现原理：
>
> 1. 服务器在接受客户端首次访问时在服务器端创建seesion，然后保存seesion(我们可以将seesion保存在内存中，也可以保存在redis中，推荐使用后者)，然后给这个session生成一个唯一的标识字符串,然后在响应头中种下这个唯一标识字符串。
> 2. 签名。这一步通过秘钥对sid进行签名处理，避免客户端修改sid。（非必需步骤）
> 3. 浏览器中收到请求响应的时候会解析响应头，然后将sid保存在本地cookie中，浏览器在下次http请求的请求头中会带上该域名下的cookie信息，
> 4. 服务器在接受客户端请求时会去解析请求头cookie中的sid，然后根据这个sid去找服务器端保存的该客户端的session，然后判断该请求是否合法。

- koa中的session使用：`npm i koa-session -S`

```
app.keys = ['some secret', 'another secret'];
```

```
const SESS_CONFIG = {
  key: 'kkb:sess',
  maxAge: 86400000,
  httpOnly: true,
  signed: true,
};

app.use(session(SESS_CONFIG, app));

app.use(ctx => {
  if (ctx.path === '/favicon.ico') return;
  let n = ctx.session.count || 0;
  ctx.session.count = ++n;
  ctx.body = '第' + n + '次访问';
});
```

- 使用redis存储session
  - 安装：`npm i -S koa-redis`
  - 配置使用：

```
const redisStore = require('koa-redis');
const redis = require('redis')
const client = redis.createClient(6379, "localhost");

app.use(session({
  key:'kkb:sess',
  store: redisStore({client})
}, app));

app.use(ctx => {
  //...
  client.keys('*',(err,keys)=>{
    console.log(keys);
    keys.forEach(key=>{
      client.get(key,(err,val)=>{
        console.log(val);
      })
    })
  })
});
```

- 案例：通过session实现用户鉴权
  - 登录页面，./public/login.html

```
<html>
  <head>
    <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
    <script src="https://unpkg.com/axios/dist/axios.min.js"></script>
  </head>

  <body>
```

```
    <div id="app">
      <div>
        <input v-model="username" />
        <input v-model="password" />
      </div>
      <div>
        <button @click="login">Login</button>
        <button @click="logout">Logout</button>
        <button @click="getUser">GetUser</button>
      </div>
      <div>
        <button @click="logs=[]">Clear Log</button>
      </div>
      <!-- 日志 -->
      <ul>
        <li v-for="(log,idx) in logs" :key="idx">
          {{ log }}
        </li>
      </ul>
    </div>
    <script>
      axios.defaults.withCredentials = true;
      axios.interceptors.response.use(response => {
        app.logs.push(JSON.stringify(response.data));
        return response;
      });
      var app = new Vue({
        el: "#app",
        data: {
          username: "test",
          password: "test",
          logs: []
        },
        methods: {
          login: async function() {
            await axios.post("/users/login", {
              username: this.username,
              password: this.password
            });
          },
          logout: async function() {
            await axios.post("/users/logout");
          },
          getUser: async function() {
            await axios.get("/users/getUser");
          }
        }
      });
    </script>
  </body>
</html>
```

- 登录/注销接口，./routes/users.js

```
router.post("/login", async ctx => {
  const { body } = ctx.request;
  console.log("body", body);

  ctx.session.userinfo = body.username;
  ctx.body = {
    ok: 1,
    message: "登录成功"
  };
});
router.post("/logout", async ctx => {
  delete ctx.session.userinfo;
  ctx.body = {
    ok: 1,
    message: "登出系统"
  };
});
router.get("/getUser", async ctx => {
    ctx.body = {
      ok: 1,
      message: "获取数据成功",
      userinfo: ctx.session.userinfo
    };
});
```

- 路由守卫中间件，./middleware/auth.js

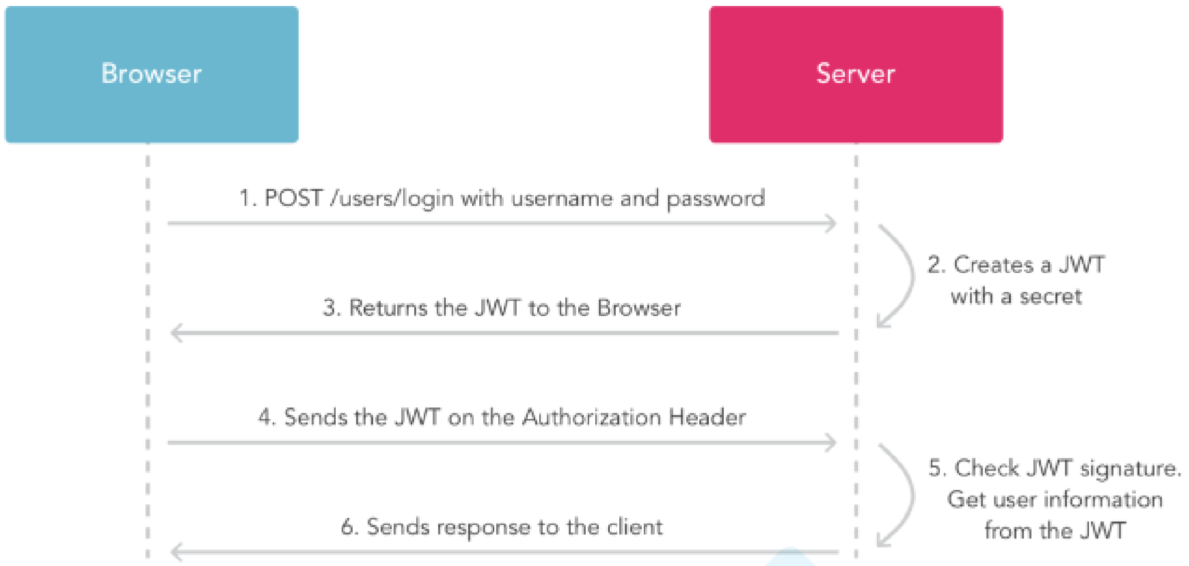
```
module.exports = async (ctx, next) => {
  if (!ctx.session.userinfo) {
    ctx.body = {
      ok: 0,
      message: "用户未登录"
    };
  } else {
    await next();
  }
};
```

- 应用守卫，./routes/users.js

```
router.get("/getUser", require("./middleware/auth"), async ctx => {...})
```

Token 验证

- 原理


Browser
Server
1. POST /users/login with username and password
2. Creates a JWT with a secret
3. Returns the JWT to the Browser
4. Sends the JWT on the Authorization Header
5. Check JWT signature. Get user information from the JWT
6. Sends response to the client

- 案例：令牌认证
  - 登录页，public/login-token.html

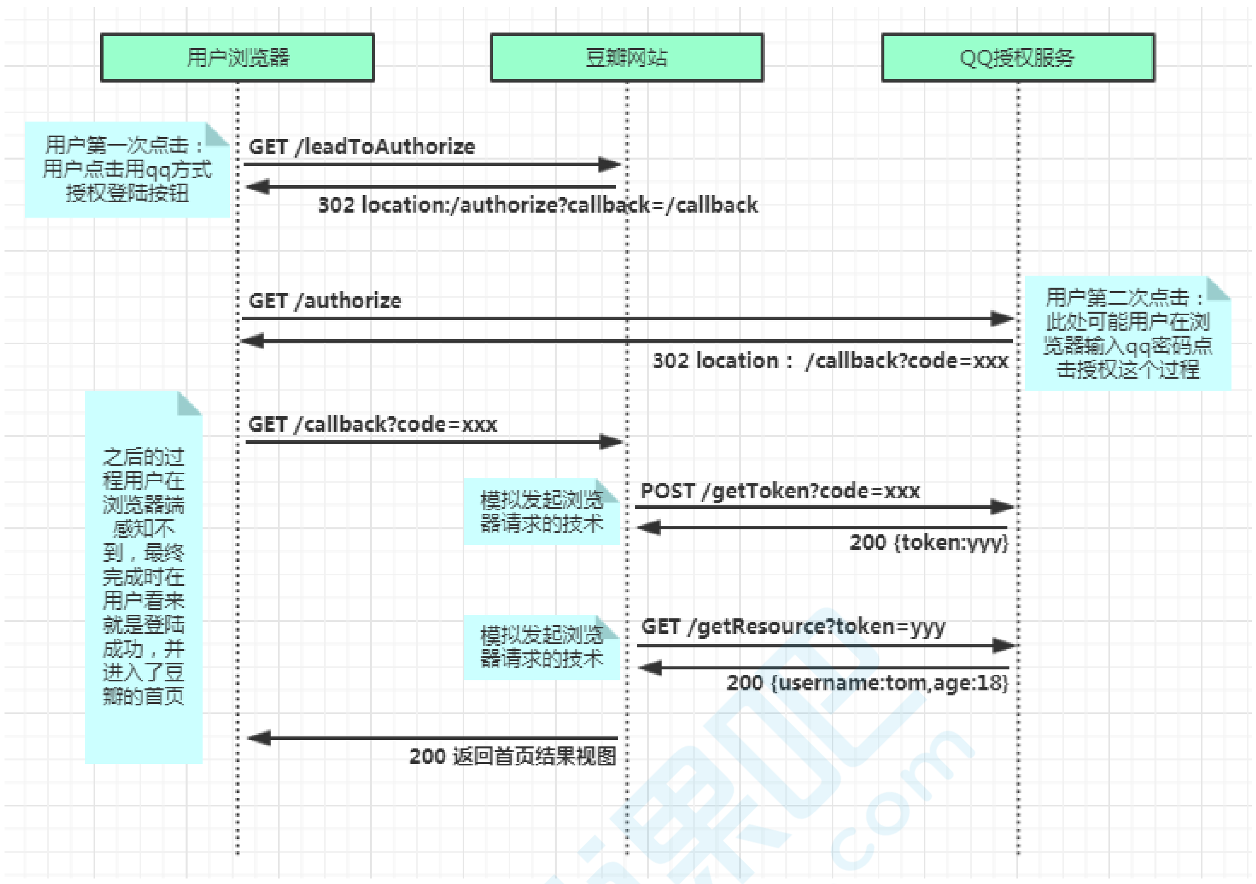
```
<html>
  <head>
    <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
    <script src="https://unpkg.com/axios/dist/axios.min.js"></script>
  </head>

  <body>
    <div id="app">
      <div>
        <input v-model="username" />
        <input v-model="password" />
      </div>
      <div>
        <button v-on:click="login">Login</button>
        <button v-on:click="logout">Logout</button>
        <button v-on:click="getUser">GetUser</button>
      </div>
      <div>
        <button @click="logs=[]">Clear Log</button>
      </div>
      <!-- 日志 -->
      <ul>
        <li v-for="(log,idx) in logs" :key="idx">
          {{ log }}
        </li>
      </ul>
    </div>
    <script>
      axios.interceptors.request.use(
        config => {
          const token = window.localStorage.getItem("token");
          if (token) {
            // 判断是否存在token，如果存在的话，则每个http header都加上token
            // Bearer是JWT的认证头部信息
            config.headers.common["Authorization"] = "Bearer " + token;
          }
          return config;
        },
        err => {
          return Promise.reject(err);
        }
      );

      axios.interceptors.response.use(
        response => {
          app.logs.push(JSON.stringify(response.data));
          return response;
        },
        err => {
          app.logs.push(JSON.stringify(response.data));
          return Promise.reject(err);
        }
      );
      var app = new Vue({
        el: "#app",
        data: {
          username: "test",
          password: "test",
          logs: []
        },
        methods: {
          login: async function() {
            const res = await axios.post("/users/login-token", {
              username: this.username,
              password: this.password
            });
            localStorage.setItem("token", res.data.token);
          },
          logout: async function() {
            localStorage.removeItem("token");
          },
          getUser: async function() {
            await axios.get("/users/getUser-token");
          }
        }
      });
    </script>
  </body>
</html>
```

- 登录接口
  - 安装依赖：`npm i jsonwebtoken koa-jwt -S`
  - 接口编写，routes/users.js

```
const jwt = require("jsonwebtoken");
const jwtAuth = require("koa-jwt");
const secret = "it's a secret";

router.post("/login-token", async ctx => {
  const { body } = ctx.request;

  const userinfo = body.username;
  ctx.body = {
    message: "登录成功",
    user: userinfo,

    token: jwt.sign(
      {
        data: userinfo,
        exp: Math.floor(Date.now() / 1000) + 60 * 60 //一小时后过期
      },
      secret
    )
  };
});

router.get(
  "/getUser-token",
  jwtAuth({
    secret
  }),
  async ctx => {
    //获取session
    ctx.body = {
      message: "获取数据成功",
      userinfo: ctx.state.user.data
    };
  }
);
```

OAuth(开放授权)

- 概述：三方登入主要基于OAuth 2.0。OAuth协议为用户资源的授权提供了一个安全的、开放而又简易的标准。与以往的授权方式不同之处是OAUTH的授权不会使第三方触及到用户的帐号信息（如用户名与密码），即第三方无需使用用户的用户名与密码就可以申请获得该用户资源的授权，因此OAUTH是安全的。
- OAuth登录流程


用户浏览器
豆瓣网站
QQ授权服务
用户第一次点击：用户点击用qq方式授权登陆按钮
GET /leadToAuthorize
302 location:/authorize?callback=/callback
GET /authorize
302 location： /callback?code=xxx
用户第二次点击：此处可能用户在浏览器输入qq密码点击授权这个过程
GET /callback?code=xxx
之后的过程用户在浏览器端感知不到，最终完成时在用户看来就是登陆成功，并进入了豆瓣的首页
模拟发起浏览器请求的技术
POST /getToken?code=xxx
200 {token:yyy}
模拟发起浏览器请求的技术
GET /getResource?token=yyy
200 {username:tom,age:18}
200 返回首页结果视图

- 案例：OAuth登录
  - 登录页面，login-oauth.html

```
<html>
<head>
  <title>OAuth</title>
</head>
<body>
  <div id="app">
      <a href='/users/login-github'>Github登录</a>
  </div>
</body>
</html>
```

- 登录接口，routes/users.js

```
const config = {
  client_id: "a141111525bac2f1edf2",
  client_secret: "8e37306c1451e60412754ace80edee4ca937564a"
};
const axios = require('axios')
const querystring = require('querystring')

router.get("/login-github", async ctx => {
  //重定向到认证接口,并配置参数
```

```
    const path = `https://github.com/login/oauth/authorize?
client_id=${config.client_id}`;

    //转发到授权服务器
    ctx.redirect(path);
  });
  router.get("/oauth/github/callback", async ctx => {
    const code = ctx.query.code;
    const params = {
      client_id: config.client_id,
      client_secret: config.client_secret,
      code: code
    };
    let res = await axios.post(
      "https://github.com/login/oauth/access_token",
      params
    );
    console.log(res.data);

    const access_token = querystring.parse(res.data).access_token;
    res = await axios.get(
      "https://api.github.com/user?access_token=" + access_token
    );
    console.log("userAccess:", res.data);
    ctx.redirect("/hello.html");
  });
```