React项目

课堂目标

1. 创建布局页面
2. Ant-design-pro应用
3. 用户登录认证
4. 创建商品页面
5. 添加购物车

资源

1. antd-pro

step01 - 迁出种子项目

```
git clone -b step01 https://github.com/57code/umi-test.git
```

step02 - 创建布局页面

- antd布局组件使用
- 修改layouts/index.js

```
import { Layout } from "antd";
import styles from "./index.css";

const { Header, Footer, Content } = Layout;

export default function(props) {
  return (
    <Layout>
      {/* 页头 */}
      <Header className={styles.header}>导航</Header>
      {/* 内容 */}
      <Content className={styles.content}>
        <div className={styles.box}>{props.children}</div>
      </Content>
      {/* 页脚 */}
      <Footer className={styles.footer}>开课吧</Footer>
    </Layout>
  );
}
```

- 样式设置，index.css

```
.header {
  color: white;
}
.content {
  margin: 16px;
}
.box {
  padding: 24px;
  background: #fff;
  min-height: 500px;
}
.footer {
  text-align: center;
}
```

- 配置式路由需要手动修改布局页面，config.js

```
{
      path: "/",
      component: "../layouts",
      routes: [
        // 移动之前路由配置到这里
      ]
}
```

- 导航菜单移动至顶部，layouts/index.js

```
import { Menu } from "antd";
import Link from "umi/link";

export default function(props) {
  return (
    <Layout>
      <Header className={styles.header}>
        {/* 新增内容 */}
        <img className={styles.logo} src="https://img.kaikeba.com/logo-new.png"/>
        <Menu
          theme="dark"
          mode="horizontal"
          defaultSelectedKeys={["2"]}
          style={{ lineHeight: "64px" }}
        >
          <Menu.Item key="1">
            <Link to="/">商品</Link>
          </Menu.Item>
          <Menu.Item key="2">
            <Link to="/users">用户</Link>
          </Menu.Item>
          <Menu.Item key="3">
            <Link to="/about">关于</Link>
```

```
            </Menu.Item>
          </Menu>
        </Header>
        ...
      </Layout>
    );
  }
```

- 样式设置，index.css

```
.logo {
  float: left;
  width: 160px;
  margin: 16px 16px 0 0;
}
```

- 登录页面不需要布局，将登录路由配置移至顶层

```
{ path: "/login", component: "./login" },
{
  path: "/",
  component: "../layouts",
  ...
}
```

- 解决导航页签在PrivateRoute重定向后未激活

```
const selectedKeys = [props.location.pathname];
<Menu selectedKeys={selectedKeys}>
  <Menu.Item key="/"></Menu.Item>
  <Menu.Item key="/users"></Menu.Item>
  <Menu.Item key="/about"></Menu.Item>
</Menu>
```

step03 - 用户登录认证

- 利用ant-design-pro中Login、Exception、图表等业务组件加速开发进度
- 引入ant-design-pro，安装：`npm install ant-design-pro --save`
- 测试：修改404页面提示内容，404.js

```
// umi的配置，已经自动支持antd-pro的按需加载
import {Exception} from 'ant-design-pro'
export default function() {
  return (
    <Exception type="404" backText="返回首页"></Exception>
  );
}
```

- 登录页构建，login.js：

```
import React, { Component } from "react";
// import { Button } from "antd";
import styles from "./login.css";
import router from "umi/router";
import { Login } from "ant-design-pro";

const { UserName, Password, Submit } = Login; // 通用的用户名、密码和提交组件

// 改为类形式组件，可持有状态
export default class extends Component {
  // let from = props.location.state.from || "/"; // 重定向地址
  onSubmit = (err, values) => {
    console.log(err, values);
  };
  render() {
    return (
      <div className={styles.loginForm}>
        {/* logo */}
        <img className={styles.logo}
          src="https://img.kaikeba.com/logo-new.png"/>
        {/* 登录表单 */}
        <Login onSubmit={this.onSubmit}>
          <UserName
            name="username"
            placeholder="kaikeba"
            rules={[{ required: true, message: "请输入用户名" }]}
          />
          <Password
            name="password"
            placeholder="123"
            rules={[{ required: true, message: "请输入密码" }]}
          />
          <Submit>登录</Submit>
        </Login>
      </div>
    );
  }
}
```

样式，login.css

```
.loginForm {
  width: 50%;
  margin: 100px auto;
  text-align: center;
}
.logo {
  margin-bottom: 30px;
}
```

- 登录接口mock，创建./mock/login.js

```
export default {
  "post /api/login"(req, res, next) {
    const { username, password } = req.body;
    console.log(username, password);
    if (username == "kaikeba" && password == "123") {
      return res.json({
        code: 0,
        data: {
          token: "kaikebaisgood",
          role: "admin",
          balance: 1000,
          username: "kaikeba"
        }
      });
    }
    if (username == "jerry" && password == "123") {
      return res.json({
        code: 0,
        data: {
          token: "kaikebaisgood",
          role: "user",
          balance: 100,
          username: "jerry"
        }
      });
    }
    return res.json({
      code: -1,
      msg: "密码错误"
    });
  }
};
```

- 用户信息保存和登录动作编写，创建./src/models/user.js

```
import axios from "axios";
import router from "umi/router";

// 初始状态：本地缓存或空值对象
const userinfo = JSON.parse(localStorage.getItem("userinfo")) || {
  token: "",
  role: "",
  username: "",
  balance: 0
};

// 登录请求方法
function login(payload) {
  return axios.post("/api/login", payload);
}

export default {
  namespace: "user", // 可省略
```

```
  state: userinfo,
  effects: {
    // action: user/login
    *login({ payload }, { call, put }) {
      const { data: {code, data: userinfo} } = yield call(login, payload);
      if (code == 0) {
        // 登录成功：缓存用户信息
        localStorage.setItem("userinfo", JSON.stringify(userinfo));
        yield put({ type: "init", payload: userinfo });
        router.push('/');
      } else {
        // 登录失败：弹出提示信息，可以通过响应拦截器实现
      }
    }
  },
  reducers: {
    init(state, action) {
      // 覆盖旧状态
      return action.payload;
    }
  }
};
```

- 请求登录，login.js

```
import { connect } from "dva";

@connect()
export default class extends Component {
  onSubmit = (err, values) => {
    console.log("用户输入：", values);
    if (!err) {
      // 校验通过，提交登录
      this.props.dispatch({ type: "user/login", payload: values });
    }
  };
  ...
}
```

- 登录失败处理：

  ~ 设置响应状态码，./mock/login.js

```
// 设置401状态码
return res.status(401).json({
    code: -1,
    msg: "密码错误"
});
```

  ~ 响应拦截，创建./src/interceptor.js

```
import axios from "axios";
```

```
import { notification } from "antd";

const codeMessage = {
  202: "一个请求已经进入后台排队（异步任务）。",
  401: "用户没有权限（令牌、用户名、密码错误）。",
  404: "发出的请求针对的是不存在的记录，服务器没有进行操作。",
  500: "服务器发生错误，请检查服务器。"
};

// 仅拦截异常状态响应
axios.interceptors.response.use(null, ({ response }) => {
  if (codeMessage[response.status]) {
    notification.error({
      message: `请求错误 ${response.status}: ${response.config.url}`,
      description: codeMessage[response.status]
    });
  }
  return Promise.reject(err);
});
```

~执行拦截器设置代码，创建./src/global.js

```
// 全局入口
import interceptor from './interceptor'
```

~ saga中异常处理，修改./src/models/user.js

```
*login({ payload }, { call, put }) {
      try {
        // 同之前，删除else部分
      } catch (error) {
        // 登录失败：错误信息已在拦截器实现，可执行其他业务
      }
}
```

step04 - 商品列表

- 数据mock，./mock/goods.js
- 图片素材，./public/courses/*.png

```
// 从远程分支step04-seed迁出新分支step04
git pull origin step04-seed:step04
```

- 修改商品数据模型，src\pages\goods\models\goods.js

```
export default {
  namespace: "goods",
  state: { // 初始状态包括课程和分类
```

```
    courses: {}, // 课程
    tags: [] // 分类
  },
  effects: {
    *getList(action, { call, put }) {
      // 解构出courseData并初始化状态
      const { data: { data: courseData } } = yield call(getGoods);
      yield put({ type: "initGoods", payload: courseData });
    }
  },
  reducers: {
    initGoods(state, { payload }) {
      // 解构出tags和courses并返回
      const { tags, data: courses } = payload;
      return { ...state, tags, courses };
    },
  }
};
```

- 显示课程分类页签，src\pages\goods.js

```
import { TagSelect } from "ant-design-pro";

@connect(
  state => ({
    courses: state.goods.courses, // 映射课程数据
    tags: state.goods.tags, // 映射标签数据
  }),
  {...}
)
class Goods extends Component {
  // 页签变更
  tagSelectChange = (tags) => {
    console.log(tags);
  };
  render() {
    if (this.props.loading.models.goods) {
      return <div>加载中...</div>;
    }
    return (
      <div>
        {/* 分类标签 */}
        <TagSelect onChange={this.tagSelectChange}>
          {this.props.tags.map(tag => {
            return (
              <TagSelect.Option key={tag} value={tag}>
                {tag}
              </TagSelect.Option>
            );
          })}
        </TagSelect>
      </div>
    );
```

```
  }
}
export default Goods;
```

- 显示课程列表，src\pages\goods.js

```
import { Card, Row, Col, Skeleton, Icon } from "antd";

class Goods extends Component {
  constructor(props) {
    super(props);
    // displayCourses为需要显示的商品数组
    this.state = {
      displayCourses: new Array(8).fill({}) // 填充数组用于骨架屏展示
    };
  }
  // 数据传入时执行一次tagSelectChange
  componentWillReceiveProps(props){
    if(props.tags.length){
      this.tagSelectChange(props.tags, props.courses)
    }
  }
  // 额外传入课程列表数据
  tagSelectChange = (tags, courses = this.props.courses) => {
    console.log(tags);
    // 过滤出要显示的数据
    let displayCourses = [];
    tags.forEach(tag => {
      displayCourses = [...displayCourses, ...courses[tag]];
    });
    this.setState({ displayCourses });
    console.log(displayCourses);
  };
  render() {
    // 使用骨架屏做加载反馈，loading属性不再需要
    // if (this.props.loading.models.goods) {
    //   return <div>加载中...</div>;
    // }
    return (
      <div>
        {/* 分类标签 */}
        {/* 商品列表 */}
        <Row type="flex" justify="start">
          {this.state.displayCourses.map((item, index) => {
            return (
              <Col key={index} style={{ padding: 10 }} span={6}>
                {item.name ? (
                  <Card
                    hoverable
                    title={item.name}
                    cover={<img src={"/course/" + item.img} />}
                  >
                    <Card.Meta
```

```
                      description={
                        <div>
                          <span>￥{item.price}</span>
                          <span style={{ float: "right" }}>
                            <Icon type="user" /> {item.solded}
                          </span>
                        </div>
                      }
                    />
                    <div />
                  </Card>
                ) : (
                  <Skeleton active={true} />
                )}
              </Col>
            );
          })}
        </Row>
      </div>
    );
  }
}
export default Goods;
```

- TagSelect初始状态调整：默认应当全选

```
constructor(props) {
    super(props);
    this.state = {
      //...
      tags: [], // 默认未选中任何标签
    };
}

tagSelectChange = (tags, courses = this.props.courses) => {
    // 用户行为修改状态
    this.setState({ displayCourses, tags });
};

// 组件受控
<TagSelect value={this.state.tags}>
```

step05 - 添加购物车

- 创建购物车模型，./src/models/cart.js

```
export default {
  namespace: "cart", // 可省略
  state: JSON.parse(localStorage.getItem("cart")) || [], // 初始状态：缓存或空数组
  reducers: {
    addCart(cart, action) {
      const good = action.payload;
```

```
      const idx = cart.findIndex(v => v.id == good.id);
      if (idx > -1) {
        // 更新数量
        const cartCopy = [...cart];
        const itemCopy = { ...cartCopy[idx] };
        itemCopy.count += 1;
        cartCopy.splice(idx, 1, itemCopy);
        return cartCopy;
      } else {
        // 新增
        return [...cart, { ...good, count: 1 }];
      }
    }
  }
};
```

- 请求添加购物车，src\pages\goods.js

```
@connect(
  state => ({ ... }),
  {
    addCart: item => ({ // 加购方法
      type: "cart/addCart",
      payload: item
    }),
  }
)
class Goods extends Component {
  addCart = (e, item) => {
    e.stopPropagation();
    this.props.addCart(item);
  };
  render() {
      <Card extra={
          <Icon onClick={e => this.addCart(e, item)}
            type="shopping-cart"
            style={{ fontSize: 18 }} />}>
  }
}
```

- 购物车数据同步到localStorage，新建./src/app.js, 配置dva

```
export const dva = {
  config: {
    onStateChange(state) {
      if (localStorage) {
        localStorage.setItem("cart", JSON.stringify(state.cart));
      }
    }
  }
};
```

- 页头添加购物车信息，./src/layouts/index.js

```
export default class extends Component {
    render(){
        ...
        <div style={{float:'right'}}>
            <Icon type="shopping-cart" style={{fontSize:18}}/>
            <span>我的购物车</span>
            <Badge count={5} offset={[-4,-18]}/>
        </div>
    }
}
```

~ 希望徽章小一点，添加全局样式，./src/global.css

```
.ant-badge-count{
    height: 16px;
    border-radius: 8px;
    min-width: 16px;
    line-height: 16px;
    padding: 0;
}
```

~ 显示购物车数据

```
@connect(state => ({ // 连接购物车状态
  count: state.cart.length,
  cart: state.cart
}))
export default class extends Component {
    render(){
        // 构造购物车列表菜单
        const menu = (
          <Menu>
            {this.props.cart.map((item, index) => (
              <Menu.Item key={index}>
                {item.name}x{item.count} <span>￥{item.count * item.price}</span>
              </Menu.Item>
            ))}
          </Menu>
        );
        ...
        return (
          {/* 购物车信息放在Dropdown以便展示 */}
          <Dropdown overlay={menu} placement="bottomRight">
            <div className={styles.cart}>
              {/* 购物车项目数量 */}
              <Badge count={this.props.count} offset={[-4, -18]} />
            </div>
          </Dropdown>
        )
    }
```

```
}
```