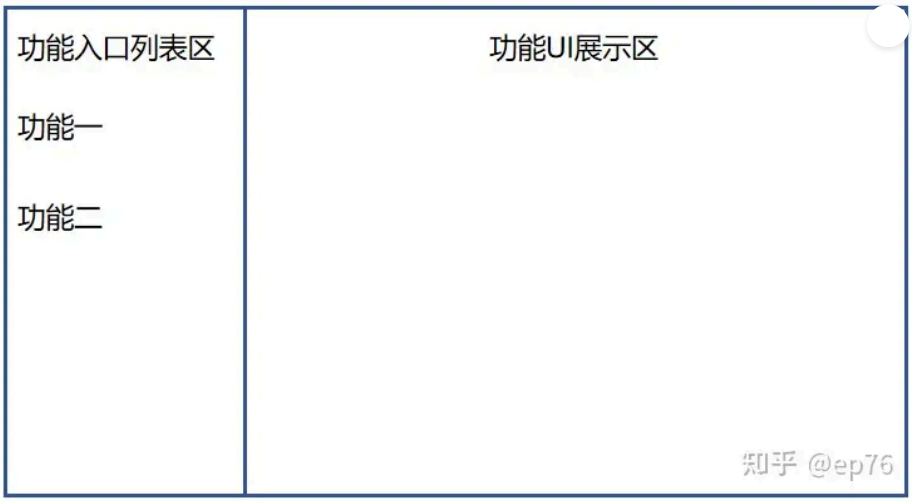
React + typescript + antd开发前端应用（五）React路由

ep76 关注

上一篇：React + typescript + antd开发前端应用（四）代码拆分

所谓路由，直观来看就是浏览器页面切换，就是通过浏览器url的不同，让浏览器展现不同页面。以一个简单的示例图进行说明：



页面分区示例图

从以上图片看，页面被分成左右两个区域，左侧是功能菜单区域，右侧是功能UI展示区域。点击左侧某功能菜单时，会在右侧显示对应功能的UI。在传统的web应用开发场景下，一般可以采用HTML的<frame>标签或<iframe>标签配合服务器端路由即可实现类似需求。但在现代web前端应用开发场景下，开发的通常是单页应用，有个专业的简称叫SPA（Single Page Application），在SPA场景下，通常就不能使用<frame>或者<iframe>标签了，而是要用React的路由组件来完成开发。

1、为项目添加路由组件依赖

以管理员权限打开cmd，依次执行以下命令：

```
npm i react-router-dom -S
npm i @types/react-router-dom -S
```

2、修改应用入口文件

还是从hello world开始：

```
import React from 'react';
import ReactDOM from 'react-dom/client';

ReactDOM.createRoot(document.getElementById("root") as HTMLElement).render(
  <React.StrictMode>
    <div>React路由练习</div>
  </React.StrictMode>
);
```

3、创建/（根）路由组件

创建src\pages\RootPage.tsx文件作为根路由对应的组件，根路由组件主要用于显示应用首页：

```
import React from 'react';

function RootPage() {
    return (
        <React.Fragment>
            <div>这是Root页面</div>
        </React.Fragment>
    );
}

export default RootPage;
```

4、创建路由定义文件

路由定义文件主要解决path和页面组件之间的对应关系，创建src\routers\RoutesDef.tsx作为路由定义文件：

```
import { createBrowserRouter } from 'react-router-dom';
import RootPage from '../pages/RootPage';//导入刚刚创建的组件

const routes = createBrowserRouter([
    {
        path: '/',//路由path
        element: <RootPage /> //path根对应的组件，又import语句生成
    }
]);

export default routes;
```

createBrowserRouter函数是创建路由定义的函数，参数就是所有的路由对象，path是路由的url，element是url对应的组件。

5、修改index.tsx文件，使用刚刚创建的路由

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import { RouterProvider } from 'react-router-dom';//导入路由定义文件
import routes from './routers/RoutesDef';

ReactDOM.createRoot(document.getElementById("root") as HTMLElement).render(
  <React.StrictMode>
    <RouterProvider router={routes}></RouterProvider>{/** 使用路由定义数据 */}
  </React.StrictMode>
);
```

完成以上修改后，在工程根目录下使用命令npm run start启动服务，可以在浏览器中看到相关页面如下：

浏览器显示RootPage组件

如果浏览器首页显示该页面，说明/路由创建成功。建议初学者在开发过程中随时查看结果以保证我们的每一步修改都不会影响各程序的正常运行。

6、修改/路由对应组件为左右两个区域

根据之前的设计草图，将/对应的页面组件修改为左右布局：

```
import React from 'react';

function RootPage() {
    return (
        <React.Fragment>
            {/** 采用原生样式，将div向左浮动，将页面分割为左右布局 */}
            <div style={{float: 'left', width: '200px', height: '600px'}}>
                左侧功能区
                <ul>
                    <li><a href={'#'}>功能一</a></li>
                    <li><a href={'#'}>功能二</a></li>
                </ul>
            </div>
            <div style={{height: '600px'}}>
                右侧主操作区
            </div>
        </React.Fragment>
    );
}

export default RootPage;
```

7、创建功能一和功能二对应的页面组件

创建功能一页面组件src\pages\PageOne.tsx：

```
import React from 'react';

function PageOne() {
    return (
        <React.Fragment>
            <h2 style={{textAlign: 'center'}}>页面组件一</h2>
        </React.Fragment>
    );
}

export default PageOne;
```

创建功能二页面组件src\pages\PageTwo.tsx：

```
import React from 'react';

function PageTow() {
    return (
        <React.Fragment>
            <h1 style={{textAlign: 'center', color: 'red'}}>页面组件二</h1>
        </React.Fragment>
    );
}
```

```
export default PageTow;
```

8、修改路由定义文件

新建的两个组件作为/路由的子路由来定义

```
import { createBrowserRouter } from 'react-router-dom';
import RootPage from '../pages/RootPage';
import PageOne from '../pages/PageOne';
import PageTow from '../pages/PageTwo';

const routes = createBrowserRouter([
    {
        path: '/',
        element: <RootPage />,
        children: [{  //新建的两个页面被定义为/路由的子路由
            path: '/one',
            element: <PageOne />
        } , {
            path: '/two',
            element: <PageTow />
        }]
    }
]);

export default routes;
```

9、修改/路由组件内容

一是修改<a>的href属性值，与路由定义中过的path属性值对应；二是在右侧div中增加<Outlet />组件作为路由组件的页面渲染出口：

```
import React from 'react';
import { Outlet } from 'react-router-dom';

function RootPage() {
    return (
        <React.Fragment>
            <div style={{float: 'left', width: '200px', height: '600px'}}>
                左侧功能区
                <ul>
                    <li><a href={'/one'}>功能一</a></li>{/** 修改href的值为路由定义的path
                    <li><a href={'/two'}>功能二</a></li>{/** 修改href的值为路由定义的path
                </ul>
            </div>
            <div style={{height: '600px'}}>
                右侧主操作区
                <Outlet /> {/** 添加子组件的渲染出口 */}
            </div>
        </React.Fragment>
    );
}

export default RootPage;
```

完成这些修改后，再次回到浏览器
读者可能会留意到，当点击功能一

“的动画，说明点击<a>的时候，浏览器发起了一次服务器GET请求，属于服务器端路由。也可以通过浏览器F12开发者工具的”网络“页签查看是否发起了GET请求。

10、修改为客户端路由

使用React路由模块提供的<Link>组件，将页面路由修改为客户端路由：

```
import React from 'react';
import { Link } from 'react-router-dom';
import { Outlet } from 'react-router-dom';

function RootPage() {
    return (
        <React.Fragment>
            <div style={{float: 'left', width: '200px', height: '600px'}}>
                左侧功能区
                <ul>
                    <li><Link to={'/one'}>功能一</Link></li>{/** 将<a>标签修改为<Link>组
                    <li><Link to={'/two'}>功能二</Link></li>
                </ul>
            </div>
            <div style={{height: '600px'}}>
                右侧主操作区
                <Outlet />
            </div>
        </React.Fragment>
    );
}

export default RootPage;
```

再次回到浏览器验证点击链接后，是否还会向服务器发起网络请求。

11、通过路由url向组件传递参数

在实际项目开发过程中通常都有类似这样的需求场景：在待办列表中选择一条记录，然后在弹出对话框中根据这个记录的主键，在弹出对话框（这个弹出对话框通常也是要由一个路由来定义）显示该待办任务的详细信息，这样的场景就需要通过url传递一些类似主键id这样的业务数据。

1、在路由目标页面定义数据加载器

修改src\pages\PageTwo.tsx组件，定义数据加载器：

```
import React from 'react';
import { useLoaderData } from 'react-router-dom';
//新增数据加载函数，函数根据url传递过来的参数，加载返回数据
export async function loader(urlBizData: any) {
    console.log(urlBizData.params.bizDataName);
    return {name: '查收数据库的数据'};//返回的示例数据，实际项目请根据场景进行修改
  }

function PageTow() {
    //获取的就是加载完成的数据
    const bizDataInfo = useLoaderData();
    console.log(bizDataInfo); //打印的就是loader返回的数据
    return (
        <React.Fragment>
            <h1 style={{text
```

```
        </React.Fragment>
    );
}

export default PageTow;
```

2、修改路由定义文件

修改路由定义文件，定义url模式和使用数据加载器，修改src\routers\RoutesDef.tsx文件内容如下：

```
import { createBrowserRouter } from 'react-router-dom';
import RootPage from '../pages/RootPage';
import PageOne from '../pages/PageOne';
import PageTow, {loader as pageLoader} from '../pages/PageTwo';//导入数据加载器

const routes = createBrowserRouter([
    {
        path: '/',
        element: <RootPage />,
        children: [{
            path: 'one',
            element: <PageOne />
        } ,  {
            path: 'two/:bizDataName',//冒号 ( : ) 具有特殊含义，将其转换为“动态段”，由<Li
            element: <PageTow />,
            loader: pageLoader,//定义数据加载器
        }]
    }
]);

export default routes;
```

3、修改<Link>组件的业务数据

修改src\pages\RootPage.tsx文件内容如下：

```
import React from 'react';
import { Link } from 'react-router-dom';
import { Outlet } from 'react-router-dom';

function RootPage() {
    return (
        <React.Fragment>
            <div style={{float: 'left', width: '200px', height: '600px'}}>
                左侧功能区
                <ul>
                    <li><Link to={'/one'}>功能一</Link></li>
                    {/** 修改to属性的值，123就是点击链接是传递的业务数据的值 */}
                    <li><Link to={'/two/123'}>功能二</Link></li>
                </ul>
            </div>
            <div style={{height: '600px'}}>
                右侧主操作区
                <Outlet />
            </div>
        </React.Fragment>
    );
}
```

```
export default RootPage;
```

以上就是Ract的路由相关内容。在实际的开发中，页面样式不应该如此简陋，后续我们将结合antd讲解如何开发一套相对完整的前端应用框架组件。

下一篇：React + typescript + antd开发前端应用（六）应用基本框架组件

编辑于 2023-10-05 02:33 · IP 属地上海

React TypeScript antd

欢迎参与讨论

还没有评论，发表第一个评论吧

推荐阅读

支持动态路由的 React Server Side Rendering 实现

云音乐技术... 发表于网易云音乐...

剖析单页面应用路由实现原理

林东洲

怎样使用React-Router实现前端路由鉴权?

React-Router 是React生态里面很重要的一环，现在React的单页应用的路由基本都是前端自己管理的，而不像以前是后端路由，React管理路由的库常用的就是就是 React-Router 。本文想写一下 Rea...

大前端奕辰

深入
统

范洪