link: null
title: 珠峰架构师成长计划
description: 需要生成一个tsconfig.json文件来告诉ts-loader如何编译代码TypeScript代码
keywords: null
author: null
date: null
publisher: 珠峰架构师成长计划
stats: paragraph=163 sentences=327, words=3043

1. 初始化项目

```
mkdir zhufeng-react-typescript2020
cd zhufeng-react-typescript2020
cnpm init -y
touch .gitignore
type nul > .gitignore
```

```
cnpm i react react-dom @types/react @types/react-dom react-router-dom @types/react-router-dom react-transition-group @types/react-transition-group react-swipe
@types/react-swipe  -S
cnpm i webpack webpack-cli webpack-dev-server html-webpack-plugin hoist-non-react-statics -D
cnpm i typescript ts-loader source-map-loader -D
cnpm i redux react-redux @types/react-redux redux-thunk  redux-logger @types/redux-logger -S
cnpm i connected-react-router -S
```

包名 作用 react @types/react react核心库 react-dom @types/react-dom react操作DOM库 react-router-dom @types/react-router-dom react路由库 react-transition-group @types/react-transition-group react动画库 react-swipe @types/react-swipe react轮播图组件库 webpack webpack核心库 webpack-cli webpack命令行文件 webpack-dev-server webpack开发服务器 html-webpack-plugin webpack用于生成html的插件 redux 全局状态管理库 react-redux @types/react-redux 连接react和redux的库 redux-thunk 可以让store派发一个函数的中间件 redux-logger @types/redux-logger 可以在状态改变前后打印状态的中间件 typescript JavaScript 语言扩展 ts-loader 可以让Webpack使用TypeScript的标准配置文件

编译TypeScript代码 source-map-loader 使用任意来自Typescript的sourcemap输出，以此通知webpack何时生成自己的sourcemaps,这让你在调试最终生成的文件时就好像在调试TypeScript源码一样

- ts-loader可以让Webpack使用TypeScript的标准配置文件tsconfig.json编译TypeScript代码。
- source-map-loader使用任意来自Typescript的sourcemap输出，以此通知webpack何时生成自己的sourcemaps,这让你在调试最终生成的文件时就好像在调试TypeScript源码一样。

需要生成一个tsconfig.json文件来告诉ts-loader如何编译代码TypeScript代码

```
tsc --init
```

```
{
  "compilerOptions": {
    "outDir": "./dist",
    "sourceMap": true,
    "strict": true,
    "noImplicitAny": true,
    "strictNullChecks": true,
    "module": "commonjs",
    "target": "es5",
    "jsx": "react",
    "baseUrl": ".",
    "paths": {
      "@/*": [
        "src/*"
      ]
    }
  },
  "include": [
    "./src/**/*"
  ]
}
```

- outDir 指定输出目录
- sourceMap：把 ts 文件编译成 js 文件的时候，同时生成对应的sourceMap文件
- noImplicitAny：如果为true的话，TypeScript 编译器无法推断出类型时，它仍然会生成 JavaScript 文件，但是它也会报告一个错误
- module：代码规范
- target：转换成es5
- jsx：react模式会生成React.createElement，在使用前不需要再进行转换操作了，输出文件的扩展名为.js
- include：需要编译的目录。

webpack.config.js

```
const webpack = require("webpack");
const HtmlWebpackPlugin = require("html-webpack-plugin");
const path = require("path");
module.exports = {
  mode: "development",
  entry: "./src/index.tsx",
  output: {
    filename: "bundle.js",
    path: path.join(__dirname, "dist"),
  },
  devtool: "source-map",
  devServer: {
    hot: true,
    contentBase: path.join(__dirname, "dist"),
    historyApiFallback: {
      index: "./index.html",
    },
  },
  resolve: {
    extensions: [".ts", ".tsx", ".js", ".json"],
    alias: {
      "@": path.resolve("src"),
    },
  },

  module: {
    rules: [
      {
        test: /\.tsx?$/,
        loader: "ts-loader",
      },
      {
        enforce: "pre",
        test: /\.tsx$/,
        loader: "source-map-loader",
      },
    ],
  },

  plugins: [
    new HtmlWebpackPlugin({
      template: "./src/index.html",
    }),
    new webpack.HotModuleReplacementPlugin(),
  ],
};
```

src\index.tsx

src\index.html

```
  typescript
```

package.json

```
{
    "scripts": {
      "build":"webpack",
      "dev": "webpack-dev-server"
    }
}
```

2. 创建组件

src\components\Counter.tsx

```
import * as React from 'react';
export interface Props {
    number: number
}
export default class Counter extends React.Component<Props>{
    render() {
        const { number } = this.props;
        return (
            <div>
                <p>{number}p>
            div>
        )
    }
}
```

src\components\Todos\types.tsx

```
export type Todo = {
    id:number;
    text:string
}
```

src\components\Todos\TodoItem.tsx

```
import * as React from "react";
import { Todo } from './types';
const todoItemStyle: React.CSSProperties = {
  color: "red",
  backgroundColor: "green",
};
interface Props {
  todo: Todo;
}
const TodoItem: React.FC = (props: Props) => (
  <li style={todoItemStyle}>{props.todo.text}li>
);
TodoItem.defaultProps;

export default TodoItem;
```

src\components\Todos\TodoInput.tsx

```
import * as React from "react";
import { Todo } from './types';
interface Props {
  addTodo:(todo:Todo)=>void
}
interface State {
  text:string
}
let id=0;
export default class TodoInput extends React.Component<Props, State> {
  constructor(props: Props) {
    super(props);
    this.state = {
      text: "",
    };
  }
  public render() {
    const { text } = this.state;
    const { handleChange, handleSubmit } = this;

    return (
      <form onSubmit={handleSubmit}>
        <input type="text" value={text} onChange={this.handleChange} />
        <button type="submit">添加button>
      form>
    );
  }

  handleChange = (e: React.ChangeEvent) => {
    this.setState({ text: e.target.value });
  }

  handleSubmit = (e: React.FormEvent) => {
    e.preventDefault();
    let text = this.state.text.trim();
    if (!text) {
      return;
    }
    this.props.addTodo({id:id++,text});
    this.setState({ text: "" });
  }
}
```

src\components\Todos\index.tsx

```
import * as React from 'react';
import TodoInput from './TodoInput';
import TodoItem from './TodoItem';
import { Todo} from './types';
const ulStyle: React.CSSProperties = {
    width: "100px"
};
export interface Props {
}
export interface State {
    todos: Todo[]
}
export default class Todos extends React.Component<Props,State>{
    state = {todos:new Array()};
    addTodo = (todo:Todo) =>{
        this.setState({todos:[...this.state.todos,todo]});
    }
    render() {
        return (
            <div>
                <TodoInput addTodo={this.addTodo}/>
                <ul style={ulStyle}>
                    {
                        this.state.todos.map(todo=><TodoItem key={todo.id} todo={todo}/>)
                    }
                ul>
            div>
        )
    }
}
```

src\index.tsx

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import Counter from "./components/Counter";
import Todos from "./components/Todos";
ReactDOM.render(<><Counter number={100} /><Todos/></>, document.getElementById("root"));
```

3. 默认属性

src\components\Todos\TodoInput.tsx

```
import * as React from "react";
import { Todo } from './types';

+let defaultProps = {
+  setting:{
+    maxLength: 6,
+    placeholder: '请输入待办事项'
+  }
+}
+export type DefaultProps = Partial;
+interface OwnProps {
+  addTodo:(todo:Todo)=>void
+}
+type Props = OwnProps & DefaultProps;
interface State {
  text:string
}
let id=0;
export default class TodoInput extends React.Component {
+   static defaultProps: Required = defaultProps;
  constructor(props: Props) {
    super(props);
    this.state = {
      text: ""
    };
  }
  public render() {
    const { text } = this.state;
+   const { setting } = this.props as Props & Required;
    const { handleChange, handleSubmit } = this;
    return (

+
        value={text} onChange={handleChange} />
        添加

    );
  }

  handleChange = (e: React.ChangeEvent) => {
    this.setState({ text: e.target.value });
  }

  handleSubmit = (e: React.FormEvent) => {
    e.preventDefault();
    let text = this.state.text.trim();
    if (!text) {
      return;
    }
    this.props.addTodo({id:id++,text});
    this.setState({ text: "" });
  }
}
```

4. 高阶组件

src\high.tsx

```
import * as React from "react";

function hoistNonReactStatics<T extends React.ComponentType<any>,S extends React.ComponentType<any>
>(
    TargetComponent: T,
    SourceComponent: S
): T & S;
function hoistNonReactStatics<N extends React.ComponentType<any>, O extends React.ComponentType<any>>
(targetComponent:N , sourceComponent: O):N&O {
    let keys = Object.getOwnPropertyNames(sourceComponent);
    for (let i = 0; i < keys.length; ++i) {
        const key = keys[i];
        const descriptor = Object.getOwnPropertyDescriptor(sourceComponent, key);
        Object.defineProperty(targetComponent, key, descriptor!);
    }
    return targetComponent as N&O;
}
interface Props{}
interface State{}
class Old extends React.Component<Props, State> {
    static age1: number = 10;
}

class New extends React.Component<Props, State> {
    static age2: number = 10;
}
let c = hoistNonReactStatics<typeof New, typeof  Old>(New, Old);
c.age1;
c.age2;
```

src\utils.tsx

```
import * as React from 'react';
import * as hoistNonReactStatics from 'hoist-non-react-statics';
export const defaultProps = {
  setting: {
    maxLength: 6,
    placeholder: '请输入待办事项'
  }
}

export type DefaultProps = Partial<typeof defaultProps>;

export const withDefaultInputProps =  (OldComponent: React.ComponentType) => {
  type OwnProps = Omit;
  class NewComponent extends React.Component {
     public render() {
       let props = { ...defaultProps, ...this.props} as Props;
       return (

       );
     }
  }
  return hoistNonReactStatics(NewComponent, OldComponent);
};
```

src\components\Todos\TodoInput.tsx

```
import * as React from "react";
import { Todo } from './types';
+import { withDefaultInputProps,DefaultProps } from '@/utils';
interface OwnProps {
  addTodo:(todo:Todo)=>void
}
type Props = OwnProps & DefaultProps;
interface State {
  text:string
}
let id=0;
class TodoInput extends React.Component {
  constructor(props: Props) {
    super(props);
    this.state = {
      text: ""
    };
  }
  public render() {
    const { text } = this.state;
+   const { setting } = this.props as (Props & Required);
    const { handleChange, handleSubmit } = this;
    return (

        添加

    );
  }

  handleChange = (e: React.ChangeEvent) => {
    this.setState({ text: e.target.value });
  }

  handleSubmit = (e: React.FormEvent) => {
    e.preventDefault();
    let text = this.state.text.trim();
    if (!text) {
      return;
    }
    this.props.addTodo({id:id++,text});
    this.setState({ text: "" });
  }
}
+ export default withDefaultInputProps(TodoInput);
```

5. 集成redux

src\models\todos.tsx

```
export type Todo = {
    id: number;
    text: string
}
```

src\models\index.tsx

```
import { Todo} from './todos';
export {
    Todo
}
```

src\store\action-types.tsx

```
export const ADD = 'ADD';
export const MINUS = 'MINUS';

export const ADD_TODO = 'ADD_TODO';
```

src\store\reducers\counter.tsx

```
import * as types from '../action-types';
import { Action } from '../actions';
export interface CounterState{
    number:number;
}
let initialState: CounterState = { number: 0 };
export default function (state: CounterState = initialState, action: Action): CounterState {
    switch (action.type) {
        case types.ADD:
            return { ...state, number: state.number + 1 };
        case types.MINUS:
            return { ...state, number: state.number - 1 };
        default:
            return state;
    }
}
```

src\store\reducers\todos.tsx

```
import { ADD_TODO} from '../action-types';
import { Action } from '../actions';
import { Todo} from '../../models';
export interface TodosState {
    list: Array;
}
let initialState: TodosState = { list: new Array()};
export default function (state: TodosState = initialState, action: Action): TodosState {
    switch (action.type) {
        case ADD_TODO:
            return { list: [...state.list,action.payload]};
        default:
            return state;
    }
}
```

src\store\reducers\index.tsx

```
import counter, { CounterState} from './counter';
import todos,{TodosState} from './todos';
import { combineReducers } from 'redux';
let reducers = {
    counter,
    todos
};
type ReducersType = typeof reducers;
type CombinedState = {
    [key in keyof ReducersType]: ReturnType
}
export { CombinedState, CounterState, TodosState}

let combinedReducer = combineReducers(reducers);
export default combinedReducer;
```

src\store\actions\counter.tsx

```
import { ADD, MINUS } from '../action-types';
export interface AddAction {
    type: typeof ADD
}
export interface MinusAction {
    type: typeof MINUS
}

export function add(): AddAction {
    return { type: ADD };
}
export function minus(): MinusAction {
    return { type: MINUS };
}
```

src\store\actions\todos.tsx

```
import { ADD_TODO } from '../action-types';
import { Todo} from '@/models';
export interface AddTodoAction {
    type: typeof ADD_TODO,
    payload:Todo
}

export function add(todo:Todo): AddTodoAction {
    return { type: ADD_TODO,payload:todo };
}
```

src\store\actions\index.tsx

```
import { AddAction,MinusAction} from './counter';
import { AddTodoAction } from './todos';
export type Action = AddAction | MinusAction | AddTodoAction;
```

src\components\Counter.tsx

```
import * as React from 'react';
import { connect } from 'react-redux';
import { CombinedState, CounterState } from '../store/reducers';
import * as actions from '@/store/actions/counter';
type StateProps = ReturnType<typeof mapStateToProps>;
type DispatchProps = typeof actions;
type Props = StateProps & DispatchProps;

class Counter extends React.Component<Props>{
    render() {
        const { number } = this.props;
        return (
            <div>
                <p>{number}p>
                <button onClick={()=>this.props.add()}>+button>
            div>
        )
    }
}

let mapStateToProps = function (state: CombinedState): CounterState {
    return state.counter;
}

export default connect(mapStateToProps, actions)(Counter);
```

src\components\Todos\TodoItem.tsx

```
import * as React from "react";
import { Todo } from '@/models';
const todoItemStyle: React.CSSProperties = {
  color: "red",
  backgroundColor: "green",
};
interface Props {
  todo: Todo;
}
const TodoItem: React.FC = (props: Props) => (
  <li style={todoItemStyle}>{props.todo.text}li>
);
TodoItem.defaultProps;

export default TodoItem;
```

src\components\Todos\TodoInput.tsx

```
import * as React from "react";
import { Todo } from '@/models';
import { withDefaultInputProps,DefaultProps } from '@/utils';
interface OwnProps {
  addTodo:(todo:Todo)=>void
}
type Props = OwnProps & DefaultProps;
interface State {
  text:string
}
let id=0;

class TodoInput extends React.Component<Props, State> {
  static age:number = 10;
  constructor(props: Props) {
    super(props);
    this.state = {
      text: ""
    };
  }
  public render() {
    const { text } = this.state;
    const { setting } = this.props as Props & Required;
    const { handleChange, handleSubmit } = this;
    return (
      <form onSubmit={handleSubmit}>
        <input type="text" maxLength={setting.maxLength} placeholder={setting.placeholder}
        value={text} onChange={handleChange} />
        <button type="submit">添加button>
      form>
    );
  }

  handleChange = (e: React.ChangeEvent) => {
    this.setState({ text: e.target.value });
  }

  handleSubmit = (e: React.FormEvent) => {
    e.preventDefault();
    let text = this.state.text.trim();
    if (!text) {
      return;
    }
    this.props.addTodo({id:id++,text});
    this.setState({ text: "" });
  }
}
export default withDefaultInputProps(TodoInput);
```

src\components\Todos\index.tsx

```
import * as React from 'react';
import TodoInput from './TodoInput';
import TodoItem from './TodoItem';
import { Todo} from '@/models';
const ulStyle: React.CSSProperties = {
    width: "100px"
};
export interface Props {
}
export interface State {
    todos: Todo[]
}
class Todos extends React.Component<Props,State>{
    addTodo = (todo:Todo) =>{
       this.props.addTodo(todo);
    }
    render() {
        return (
            <div>
                <TodoInput addTodo={this.addTodo}/>
                <ul style={ulStyle}>
                    {
                        this.props.list.map(todo=><TodoItem key={todo.id} todo={todo}/>)
                    }
                ul>
            div>
        )
    }
}
let mapStateToProps = function (state: CombinedState): TodosState {
    return state.todos;
}
export default connect(mapStateToProps, actions)(Todos);
```

src\store\index.tsx

```
import { createStore } from 'redux'
import combinedReducer from './reducers';
let store = createStore(combinedReducer);
export default store;
```

src\index.tsx

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import Counter from "./components/Counter";
import Todos from "./components/Todos";
import { Provider } from 'react-redux';
import store from './store';
ReactDOM.render(
<Provider store={store}>
    <React.Fragment>
        <Counter />
        <hr/>
        <Todos />
    React.Fragment>
Provider>, document.getElementById("root"));
```

6. 使用路由

src\index.tsx

```
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import Counter from './components/Counter';
import Todos from './components/Todos';
import { Provider } from 'react-redux';
import store from './store';
+import { BrowserRouter as Router, Route, Link } from 'react-router-dom';
ReactDOM.render((

+
+                    counter
+                    todos
+
+
+

), document.getElementById('root'));
```

src\components\Counter.tsx

```
import * as React from 'react';
import { connect } from 'react-redux';
import { CombinedState, CounterState } from '../store/reducers';
+import * as actions from '@/store/actions/counter';
+import { RouteComponentProps } from 'react-router-dom';
+import { StaticContext} from 'react-router';
+type StateProps = ReturnType;
+type DispatchProps = typeof actions;
+interface Params { name:string}
+interface LocationState { }
+type Props = StateProps & DispatchProps & RouteComponentProps;

class Counter extends React.Component{
    render() {
+        const { name} = this.props.match.params;
        const { number } = this.props;
        return (

+                名称:{name}
                {number}
                this.props.add()}>+
                 this.props.history.push({ pathname: "/todos", state: { todoName: 'todoName' } })}>/todos

        )
    }
}

let mapStateToProps = function (state: CombinedState): CounterState {
    return state.counter;
}

export default connect(mapStateToProps, actions)(Counter);
```

src\components\Todos\index.tsx

```
import * as React from 'react';
import TodoInput from './TodoInput';
import TodoItem from './TodoItem';
import { Todo} from '@/models';
+import { CombinedState, CounterState, TodosState } from '@/store/reducers';
+import { RouteComponentProps } from 'react-router-dom';
+import { StaticContext } from 'react-router';
+import * as actions from '@/store/actions/todos';
+import { connect } from 'react-redux';
+type StateProps = ReturnType;
+type DispatchProps = typeof actions;
+const ulStyle: React.CSSProperties = {
+    width: "100px"
+};
+export interface State {
+    todos: Todo[]
+}
+interface Params {}
+interface LocationState { name:string }
+type Props = StateProps & DispatchProps & RouteComponentProps;
class Todos extends React.Component{
    addTodo = (todo:Todo) =>{
       this.props.addTodo(todo);
    }
    render() {
        return (

+               名称:{this.props.location.state.name}

                    {
                        this.props.list.map(todo=>)
                    }

        )
    }
}
let mapStateToProps = function (state: CombinedState): TodosState {
    return state.todos;
}
export default connect(mapStateToProps, actions)(Todos);
```

7. connected-react-router

src\history.tsx

```
import { createBrowserHistory } from 'history'
const history = createBrowserHistory()
export default history;
```

src\index.tsx

```
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import Counter from './components/Counter';
import Todos from './components/Todos';
import { Provider } from 'react-redux';
import store from './store';
import {Route, Link } from 'react-router-dom';
+import history from '@/history';
+import {ConnectedRouter} from 'connected-react-router';
ReactDOM.render((

+

                    counter
                    todos

+

), document.getElementById('root'));
```

src\store\index.tsx

```
import { createStore, applyMiddleware } from 'redux'
+import combinedReducer from './reducers';
+import { routerMiddleware } from 'connected-react-router';
+import history from '@/history';

+let store =applyMiddleware(routerMiddleware(history))(createStore)(combinedReducer);
export default store;
```

src\store\reducers\index.tsx

```
import counter, { CounterState} from './counter';
import todos,{TodosState} from './todos';
import { combineReducers } from 'redux';
+import history from '@/history';
+import { connectRouter } from 'connected-react-router'
let reducers = {
    counter,
    todos,
+    router: connectRouter(history)
};
type ReducersType = typeof reducers;
type CombinedState = {
    [key in keyof ReducersType]: ReturnType
}
export { CombinedState, CounterState, TodosState}

let combinedReducer = combineReducers(reducers);
export default combinedReducer;
```

src\store\actions\counter.tsx

```
import { ADD, MINUS } from '../action-types';
+import { push, CallHistoryMethodAction } from 'connected-react-router';
+import { LocationDescriptorObject,Path} from 'history';
+import { TodosLocationState} from '@/components/Todos';
export interface AddAction {
    type: typeof ADD
}
export interface MinusAction {
    type: typeof MINUS
}
export function add(): AddAction {
    return { type: ADD };
}
export function minus(): MinusAction {
    return { type: MINUS };
}
+export function go(path: LocationDescriptorObject): CallHistoryMethodAction {
+    return push(path);
+}
```

src\store\actions\index.tsx

```
import { AddAction,MinusAction} from './counter';
import { AddTodoAction } from './todos';
+import { CallHistoryMethodAction } from 'connected-react-router';
+export type Action = AddAction | MinusAction | AddTodoAction | CallHistoryMethodAction;
```

src\components\Counter.tsx

```
import * as React from 'react';
import { connect } from 'react-redux';
import { CombinedState, CounterState } from '../store/reducers';
import * as actions from '@/store/actions/counter';
import { RouteComponentProps } from 'react-router-dom';
+import { StaticContext} from 'react-router';
+import { LocationDescriptorObject} from 'history';
+import { TodosLocationState } from '@/components/Todos';
+type StateProps = ReturnType;
+type DispatchProps = typeof actions;
+interface Params { name:string}
+interface CounterLocationState { }
+type Props = StateProps & DispatchProps & RouteComponentProps;
class Counter extends React.Component{
    render() {
+       const { name} = this.props.match.params;
        const { number } = this.props;
+        let path: LocationDescriptorObject = { pathname: "/todos", state: { name: 'todoName' } };
        return (

+                名称:{name}
                {number}
                this.props.add()}>+
                 this.props.go(path)}>/todos

        )
    }
}

let mapStateToProps = function (state: CombinedState): CounterState {
    return state.counter;
}

export default connect(mapStateToProps, actions)(Counter);
```

src\components\Todos\index.tsx

```
import * as React from 'react';
import TodoInput from './TodoInput';
import TodoItem from './TodoItem';
import { Todo} from '@/models';
import { CombinedState, CounterState, TodosState } from '@/store/reducers';
import { RouteComponentProps } from 'react-router-dom';
import { StaticContext } from 'react-router';
import * as actions from '@/store/actions/todos';
import { connect } from 'react-redux';
type StateProps = ReturnType;
type DispatchProps = typeof actions;
const ulStyle: React.CSSProperties = {
    width: "100px"
};
export interface State {
    todos: Todo[]
}
+interface Params {}
+export interface TodosLocationState { name: string }
+type Props = StateProps & DispatchProps & RouteComponentProps;
class Todos extends React.Component{
    addTodo = (todo:Todo) =>{
       this.props.addTodo(todo);
    }
    render() {
        return (

+                名称:{this.props.location.state.name}

                   {
                       this.props.list.map(todo=>)
                   }

        )
    }
}
let mapStateToProps = function (state: CombinedState): TodosState {
    return state.todos;
}
export default connect(mapStateToProps, actions)(Todos);
```

8. redux-thunk

src\redux-thunk\index.tsx

```
import {
    Middleware,
    Dispatch,
    MiddlewareAPI,
    Action
} from "redux";

export type ThunkAction = (
    dispatch: ThunkDispatch,
    getState: () => S
) => void;
export interface ThunkDispatch {
    (action: T): T;
    (asyncAction: ThunkAction): R;
}
type ThunkDispatched = ThunkDispatch;
const thunk: Middleware<
    ThunkDispatched,
    {},
    ThunkDispatched
> = (api: MiddlewareAPI) => (
    next: Dispatch
) => (action: Function | Action): any => {
    if (typeof action === "function") {
        return action(api.dispatch, api.getState);
    }
    return next(action);
};
export default thunk;
```

src\store\index.tsx

```
import {
    createStore,
    applyMiddleware,
    Action,
    Store,
    AnyAction,
    StoreEnhancer,
    StoreEnhancerStoreCreator
} from "redux";
import combinedReducer,{CombinedState} from './reducers';
import { routerMiddleware } from 'connected-react-router';
import history from '@/history';

import thunk, { ThunkAction, ThunkDispatch } from '@/redux-thunk';
interface Ext {
    dispatch: ThunkDispatchundefined, AnyAction>
}
interface StateExt{}
let storeEnhancer: StoreEnhancer = applyMiddleware(routerMiddleware(history),thunk);

let storeEnhancerStoreCreator: StoreEnhancerStoreCreator = storeEnhancer(createStore);
let store: Store & Ext = storeEnhancerStoreCreator(combinedReducer);
let thunkAction: ThunkAction<void,CombinedState, undefined, AnyAction> = (
    dispatch: ThunkDispatchundefined, AnyAction>,
    getState: () => CombinedState
): void => { }
store.dispatch(thunkAction);
export default store;
```