МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ» (ДГТУ)

Факультет
Информатика и вычислительная техника

Кафедра
Программное обеспечение вычислительной техники и автоматизированных систем

Использование Redux в React

Практикум

по выполнению лабораторной работы №12
по дисциплине «Веб-технологии»

Программное обеспечение вычислительной техники и автоматизированных систем
(указывается направленность (профиль) образовательной программы)

09.03.04 Программная инженерия
(указывается код и наименование направления подготовки)

Ростов-на-Дону
2024 г.

Составители: ст. пр. Жучок И.О., ст. пр. Коротков Д.С.

УДК 004.432

Базовые типы данных, управляющие конструкции и методы: метод. указания. – Ростов н/Д: Издательский центр ДГТУ, 2024.

В методическом указании рассматриваются вопросы использования контейнеров Redux в приложениях на React. Приведены задания к лабораторной работе, помогающие закрепить на практике полученные знания, и контрольные вопросы для самопроверки.
Предназначено для обучающихся по направлению 09.03.04 «Программная инженерия», профиль «Программное обеспечение вычислительной техники и автоматизированных систем».

Ответственный за выпуск:
зав. кафедрой «Программное обеспечение вычислительной техники и автоматизированных систем» Долгов В.В.

1. Теоретическая часть

Redux представляет собой контейнер для управления состоянием приложения. Redux не привязан непосредственно к React.js и может также использоваться с другими js-библиотеками и фреймворками.

Рассмотрим схему приложения, использующего Redux:

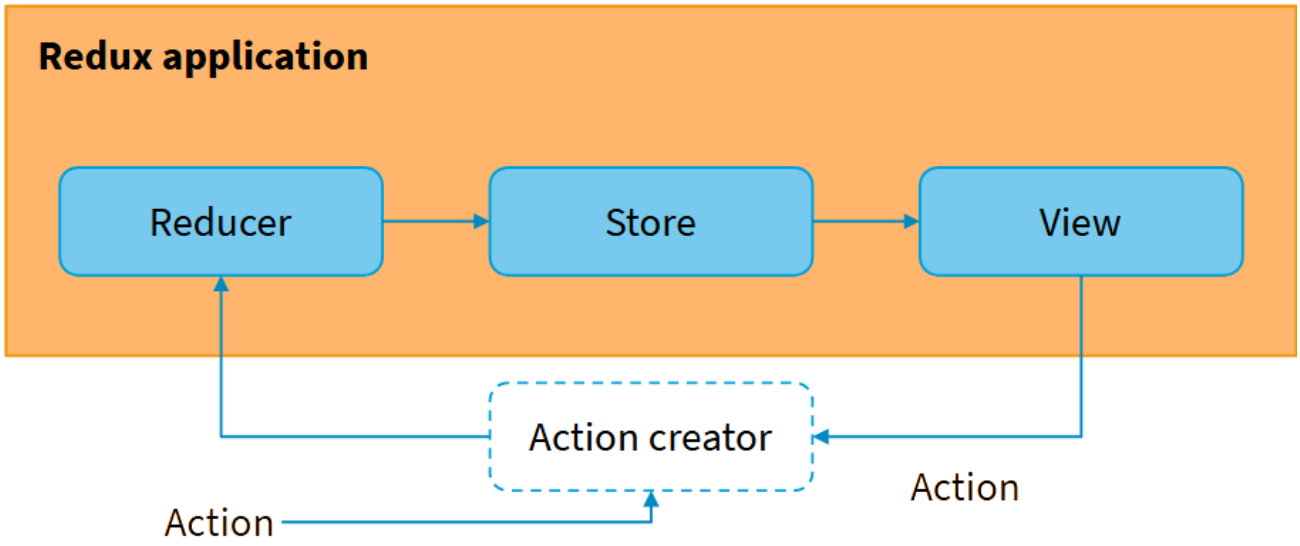

Рис 1. Приложения Redux.

Reducer – функция (или несколько функций), которая получает действие и в соответствии с этим действием изменяет состояние хранилища.

Store – хранит состояние приложения.

View – представление, отображающее контент.

Action creator – функция, которая создаёт действие.

Action – некоторый набор информации, который исходит от приложения к хранилищу и который указывает, что именно нужно сделать. Для передачи этой информации у хранилища вызывается метод dispatch().

Рассмотрим шаги для создания простого приложения, использующего контейнеры Redux.

1. Создать новую папку с проектом и открыть её
2. Указать пакеты npm, которые нам будут нужны, для этого создать файл package.json со следующим содержанием:

```
{
  "name": "reduxapp",
  "description": "A React.js project using Redux",
```

```
  "version": "1.0.0",
  "author": "metanit.com",
  "scripts": {
    "dev": "webpack serve",
    "build": "webpack"
  },
  "dependencies": {
    "react": "18.0.0",
    "react-dom": "18.0.0",
    "react-redux": "7.2.8",
    "redux": "4.1.2",
     "immutable": "4.0.0"
  },
  "devDependencies": {
    "@babel/cli": "7.17.0",
    "@babel/core": "7.17.0",
    "@babel/preset-react": "7.16.0",
    "babel-loader": "8.2.0",
    "webpack": "5.70.0",
    "webpack-cli": "4.10.0",
    "webpack-dev-server": "4.7.0"
  }
}
```

3. Настроить сервер webpack, для этого создать файл webpack.config.js следующего содержания:

```
const path = require("path");

module.exports = {
    mode: "development",
    entry: "./app/app.jsx", // входная точка - исходный файл
    output:{
        path: path.resolve(__dirname, "./js"),     // путь к каталогу выходных файлов - папка public
        publicPath: "/js/",
        filename: "main.js"       // название создаваемого файла
    },
    devServer: {
     historyApiFallback: true,
     static: {
      directory: path.join(__dirname, "/"),
    },
     port: 8081,
```

```
    open: true
  },
  module:{
    rules:[   //загрузчик для jsx
      {
        test: /\.jsx?$/, // определяем тип файлов
        exclude: /(node_modules)/,  // исключаем из обработки папку node_modules
        loader: "babel-loader",   // определяем загрузчик
        options:{
          presets:[ "@babel/preset-react"]    // используемые плагины
        }
      }
    ]
  }
}
```

4. Создать главную страницу сайта index.html:

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>Hello Redux</title>
</head>
<body>
  <div id="app"></div>
  <script src="js/main.js"></script>
</body>
</html>
```

5. Создать папку app
6. Создадим файл с действиями app/actions.jsx:

```
const addPhone = function (phone) {
  return {
    type: "ADD_PHONE",
    phone
  }
};
const deletePhone = function (phone) {
  return {
    type: "DELETE_PHONE",
    phone
  }
};
module.exports = {addPhone, deletePhone};
```

7. Создать reducer app/reducer.js:

```
const Map = require("immutable").Map;

const reducer = function(state = Map(), action) {
 switch (action.type) {
  case "SET_STATE":
     return state.merge(action.state);
  case "ADD_PHONE":
     return state.update("phones", (phones) => [...phones, action.phone]);
  case "DELETE_PHONE":
     return state.update("phones",
        (phones) => phones.filter(
           (item) => item !== action.phone
        )
     );
 }
 return state;
}
module.exports = reducer;
```

8. Создать представление app/view.jsx:

```
const React = require("react");
const connect = require("react-redux").connect;
const actions = require("./actions.jsx");

class PhoneForm extends React.Component {
 constructor(props) {
  super(props);
  this.phoneInput = React.createRef();
 }
 onClick() {
  if (this.phoneInput.current.value !== "") {

    const itemText = this.phoneInput.current.value;
    this.phoneInput.current.value ="";
    return this.props.addPhone(itemText);
  }
 }
 render() {
  return <div>
        <input ref={this.phoneInput} />
        <button onClick = {this.onClick.bind(this)}>Добавить</button>
     </div>
```

```
  }
};

class PhoneItem extends React.Component {
  constructor(props) {
    super(props);
  }
  render() {

    return <div>
              <p>
                 <b>{this.props.text}</b><br />
                 <button onClick={() =>
this.props.deletePhone(this.props.text)}>Удалить</button>
              </p>
           </div>
  }
};

class PhonesList extends React.Component {
  constructor(props) {
    super(props);
  }
  render() {
    return <div>
       {this.props.phones.map(item =>
         <PhoneItem key={item}
                text={item}
                deletePhone={this.props.deletePhone}/>
       )}
     </div>
  }
};

class View extends React.Component {

   render() {
      return <div>
         <PhoneForm addPhone={this.props.addPhone}/>
         <PhonesList {...this.props} />
   </div>
  }
```

```
};

function mapStateToProps(state) {
  return {
    phones: state.get("phones")
  };
}

module.exports = connect(mapStateToProps, actions)(View);
```

9. Создаём основной файл приложения app/app.jsx:

```
const React = require("react");
const ReactDOM = require("react-dom/client");
const redux = require("redux");
const Provider = require("react-redux").Provider;
const reducer = require("./reducer.jsx");
const AppView = require("./view.jsx");

const store = redux.createStore(reducer);

store.dispatch({
  type: "SET_STATE",
  state: {
    phones: [ "Xiaomi Mi 10", "Samsung Galaxy Note20" ]
  }
});

ReactDOM.createRoot(
    document.getElementById("app")
)
.render(
  <Provider store={store}>
    <AppView />
  </Provider>
);
```

Для запуска приложение остаётся выполнить команду npm install для установки пакетов, а потом npm start для запуска.

Zustand - это небольшая, быстрая и масштабируемая библиотека для управления состоянием. Она использует упрощенный подход, основанный на хуках, и не требует boilerplate-кода, характерного для Redux.

2. Задание к лабораторной работе

2.1. Установить платформу Node.js, для скачивания установщика можно воспользоваться ссылкой https://nodejs.org/en/download.

2.2. Выполнить создание приложения описанного выше

2.3. Выполнить задания лабораторной работы №6 с применением контейнеров Redux.

2.4. Реализовать приложение "Список дел", используя Context API и useReducer для управления состоянием (вместо Redux).

2.5. Сравнить (в отчете) преимущества и недостатки использования Context API + useReducer по сравнению с Redux для данной задачи. Какие факторы влияют на выбор того или иного подхода?

2.6. Реализовать приложение "Список дел", используя Zustand для управления состоянием.

2.7. Сравнить (в отчете) преимущества и недостатки использования Zustand по сравнению с Redux и Context API + useReducer для данной задачи. Какие факторы влияют на выбор того или иного подхода? Влияет ли размер приложения на выбор библиотеки для управления состоянием?

3 Материально-техническое обеспечение работы

Аудитория для проведения лабораторных занятий должна быть укомплектована специализированной мебелью и индивидуальными компьютерами следующей минимальной комплектации:

- Процессор: не менее двух исполнительных ядер, совместимый с системой команд x86 и x64, с поддержкой аппаратной виртуализации.
- Оперативная память: не менее 8 Гб.
- Монитор: не менее 24" (дюймов) по диагонали.
- Наличие локальной сети со скоростью обмена не менее 1 Гб/сек.
- Наличие доступа в сеть Интернет со скоростью не менее 1 Мбит/сек.
- Наличие клавиатуры и манипулятора «мышь».

На компьютерах должно быть установлено следующее программное обеспечение:

- Операционная система: Microsoft Windows 10 (или выше) и/или одна из операционных систем семейства Linux, допускающих установку программной

платформы Node.js, например, РедОС (версия 8 или выше), Ubuntu Desktop (версия 22.04 или выше), Linux Mint (версия 21 или выше). Допускается конфигурация, когда одна из операционных систем установлена внутри виртуальной машины (гипервизора),

- Программная платформа Node.js (версия 20 или выше),
- Редактор исходного кода Visual Studio Code.

4 Порядок выполнения и сдачи работы

Для выполнения лабораторной работы рекомендуется придерживаться следующего порядка выполнения.

1. Ознакомиться с темой и целями лабораторной работы.
2. Изучить теоретический материал.
3. Подготовить рабочее окружение.
4. Разработать программный код лабораторной работы в соответствии с заданием.
5. Произвести проверку работоспособности, выполнить тестирование и отладку кода.
6. Проанализировать полученные результаты и примененные в ходе решения подходы.
7. Выполнить самооценку и рефлексию.
8. Сдать лабораторную работу преподавателю и получить от него обратную связь.

Выполнение и сдача лабораторной работы происходит непосредственно на занятии, в присутствии преподавателя. Основным отчётом выступает самостоятельно решённое задание к лабораторной работе.

Сдача исходного кода ранее сдававшихся программ или программ, код которых выложен в сети Интернет, не допускается.

5 Контрольные вопросы к лабораторной работе

1. Что такое Redux?
2. Назовите основные понятия Redux.
3. Что такое Reducer?
4. Что такое Store?
5. Зачем нужен Action creator?

6 Перечень использованных информационных ресурсов

1. Официальный сайт продукта https://ru.react.js.org/docs/getting-started.html
2. Пример простого приложения https://metanit.com/web/react/5.3.php
3. Введение в Redux и React-redux https://habr.com/ru/articles/498860/

Редактор А.А. Литвинова

ЛР № 04779 от 18.05.01.	В набор	В печать
Объем 0,5 усл.п.л., уч.-изд.л.	Офсет.	Формат 60x84/16.
Бумага тип №3.	Заказ №	Тираж 75. Цена

Издательский центр ДГТУ

Адрес университета и полиграфического предприятия:

344010, г. Ростов-на-Дону, пл. Гагарина, 1.