GABRIELE ROMANATO

Usare Redux in React

React è una libreria JavaScript che ha rivoluzionato lo sviluppo di interfacce utente, semplificando la creazione di componenti riutilizzabili e reattive. Tuttavia, a mano a mano che le applicazioni crescono in complessità, la gestione dello state diventa una sfida sempre più rilevante. È qui che Redux entra in gioco come uno strumento potente per gestire lo state in modo efficace e prevedibile.

Cos'è Redux?

Redux è una libreria di gestione dello state per JavaScript, spesso utilizzata con React, anche se può essere adottata con altri framework o librerie. La sua idea fondamentale è quella di centralizzare lo state dell'applicazione in uno store, rendendo più facile la gestione e l'aggiornamento dello state complesso.

Principi chiave di Redux

1. **Store**: Uno store è un oggetto che tiene lo state dell'applicazione. L'unico modo per modificare lo state in un'app Redux è inviare un'azione, un oggetto che descrive cosa è successo.
2. **Azione (Action)**: Le azioni sono oggetti piani che rappresentano un cambiamento nello state. Contengono un tipo che identifica il tipo di azione e, opzionalmente, dei dati aggiuntivi.
3. **Reducer**: I reducer sono funzioni pure che specificano come lo state dell'applicazione cambia in risposta a un'azione. Ricevono lo state corrente e un'azione, restituendo il nuovo state.
4. **Dispatch**: La funzione `dispatch` è il modo con cui le azioni vengono inviate allo store. Quando chiami `dispatch` con un'azione, lo store chiama il tuo riduttore con l'azione fornita.
5. **Middleware**: Redux offre la possibilità di utilizzare middleware per eseguire azioni asincrone, gestire effetti collaterali o intervenire durante la fase di dispatch.

Come integrare Redux in una applicazione React

Installazione dei pacchetti necessari

Per utilizzare Redux in un progetto React, è necessario installare alcuni pacchetti. Puoi farlo tramite npm o yarn:

```
npm install redux react-redux
# o
yarn add redux react-redux
```

Creare uno store

```
// store.js
```

```
import { createStore } from 'redux';
import rootReducer from './reducers'; // Importa il tuo riduttore principale

const store = createStore(rootReducer);

export default store;
```

Definire le azioni

```
// actions.js
export const increment = () => {
  return {
    type: 'INCREMENT',
  };
};

export const decrement = () => {
  return {
    type: 'DECREMENT',
  };
};
```

Creare i reducer

```
// reducers.js
const initialState = {
  count: 0,
};

const counterReducer = (state = initialState, action) => {
  switch (action.type) {
    case 'INCREMENT':
      return { count: state.count + 1 };
    case 'DECREMENT':
      return { count: state.count - 1 };
    default:
      return state;
  }
};

export default counterReducer;
```

Collegare React a Redux

```
// App.js
import React from 'react';
import { useSelector, useDispatch } from 'react-redux';
import { increment, decrement } from './actions';
```

```
function App() {
  const count = useSelector((state) => state.count);
  const dispatch = useDispatch();

  return (
    <div>
      <h1>Contatore: {count}</h1>
      <button onClick={() => dispatch(increment())}>Incrementa</button>
      <button onClick={() => dispatch(decrement())}>Decrementa</button>
    </div>
  );
}

export default App;
```

Collegare lo store all'applicazione React

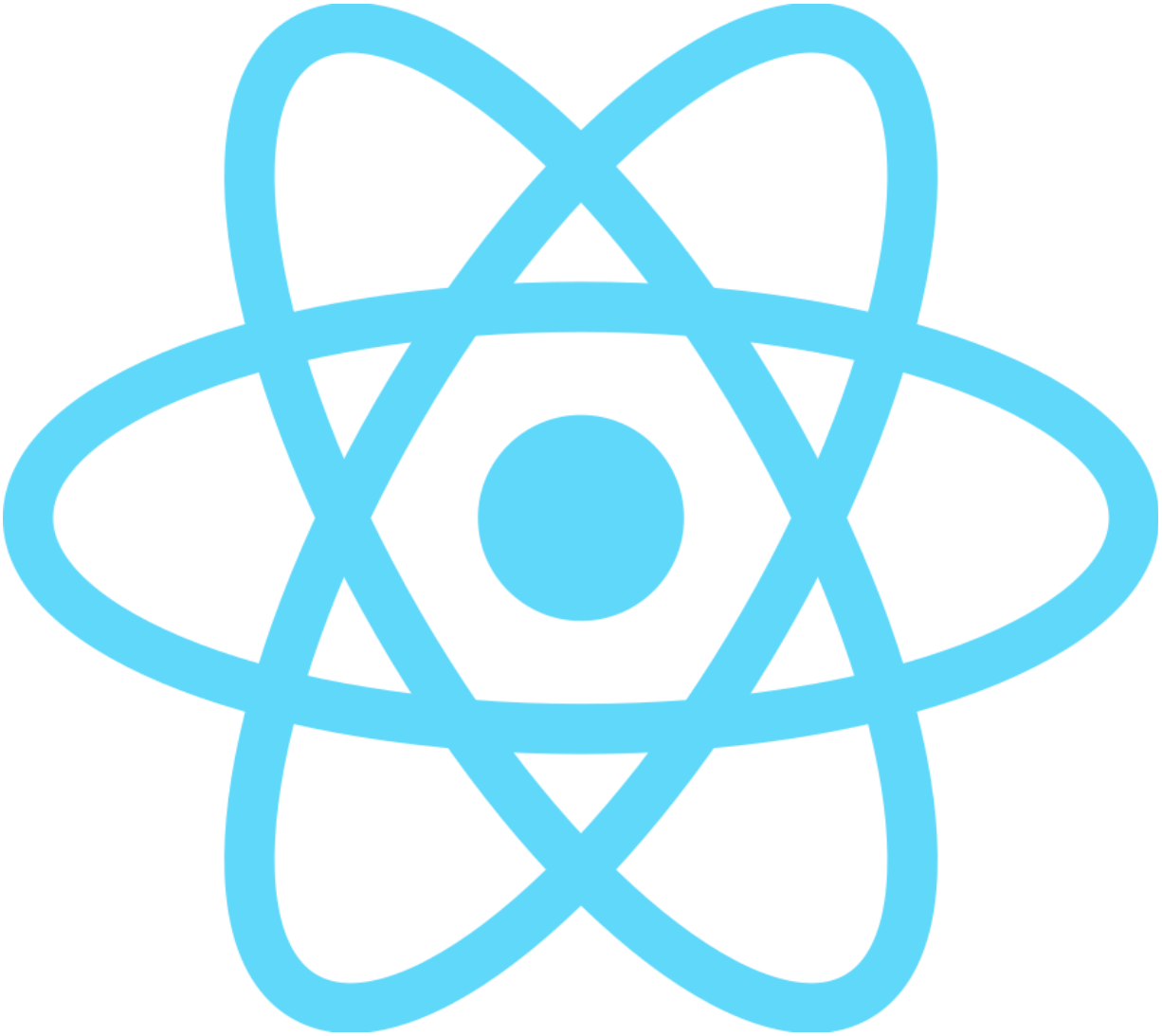
```
// index.js
import React from 'react';
import ReactDOM from 'react-dom';
import { Provider } from 'react-redux';
import store from './store';
import App from './App';

ReactDOM.render(
  <Provider store={store}>
    <App />
  </Provider>,
  document.getElementById('root')
);
```

Conclusioni

Integrare Redux in un progetto React può sembrare inizialmente complicato, ma una volta comprese le basi, offre un modo robusto e scalabile per gestire lo stato dell'applicazione. Centralizzando lo stato in uno store e seguendo il flusso unidirezionale delle azioni, Redux fornisce un'architettura chiara e prevedibile che facilita lo sviluppo e la manutenzione delle applicazioni complesse.

Applicazioni Correlate



Book Store

Un'applicazione in React con Node.js e Fastify come backend.
DockerDocker ComposeNode.jsJavaScriptFastifyReact