

Memento

دیزاین پترن هادر React

Behavioral Patterns

به زبان ساده همراه با مثال های عملی
کاربرد دیزاین پترن ها برای برنامه نویسان React



Behavioral Patterns

Memento



الگوی Memento برای ذخیره و بازیابی وضعیت داخلی یک شیء به کار می‌رود. این الگو معمولاً برای پیاده‌سازی قابلیت‌هایی مانند بازگشت به حالت قبلی یا بازگرداندن وضعیت استفاده می‌شود. ذخیره اطلاعات بسته به نیاز می‌تواند در حافظه‌های اصلی و یا حافظه‌های جانبی انجام شود.

از جمله مواردی که می‌توان با استفاده از این الگو آن‌ها را پیاده‌سازی کرد می‌توان به عملیات Undo و Redo یا Rollback یا Check Point یا عملیات ذخیره‌سازی در بازی‌های کامپیوترا اشاره کرد.

فرض کنید می‌خواهیم یک ویرایشگر متن بسازیم که در آن قابلیت ذخیره‌سازی و بازیابی وضعیت را داشته باشیم.

بخش Memento

این بخش وضعیت Originator را ذخیره می کند و پس از ایجاد، قابل تغییر نیست. از این کلاس برای بازیابی وضعیت Originator استفاده می شود.



```
● ● ●  
export default class Memento {  
    private state: string;  
  
    constructor(state: string) {  
        this.state = state;  
    }  
  
    public getState(): string {  
        return this.state;  
    }  
}
```

بخش Editor

این بخش که همان Originator می باشد، وظیفه نگه داری از وضعیت را بر عهده دارد و می تواند Memento هایی از وضعیت خود بسازد و آن ها را بازیابی کند.



```
import Memento from './memento.class';

export default class Editor {
    private state: string = '';

    public setState(state: string): void {
        this.state = state;
    }

    public getState(): string {
        return this.state;
    }

    public saveStateToMemento(): Memento {
        return new Memento(this.state);
    }

    public restoreStateFromMemento(memento: Memento): void {
        this.state = memento.getState();
    }
}
```

بخش Caretaker

این بخش وظیفه ذخیره سازی و بازیابی Memento ها را بر عهده دارد اما تغییراتی روی آن ها ایجاد نمی کند.



Caretaker

```
import Memento from './memento.class';

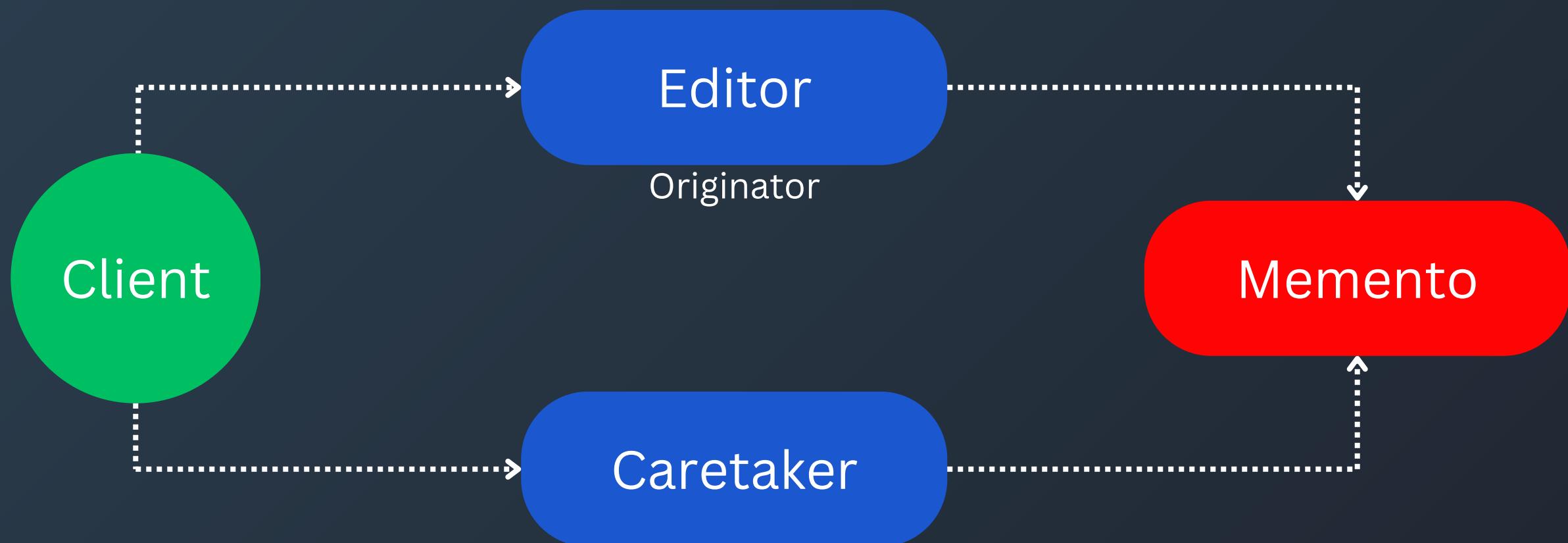
export default class Caretaker {
    private mementoList: Memento[] = [];

    public addMemento(memento: Memento): void {
        this.mementoList.push(memento);
    }

    public getMemento(index: number): Memento {
        return this.mementoList[index];
    }
}
```

نحوه عملکرد

در فلوجارت زیر نحوه عملکرد و ارتباط بخش های Originator و Memento و Caretaker را با یکدیگر مشاهده می کنید.



بخش TextEditor

در این بخش Component خود را تحت عنوان TextEditor و به شکل زیر می سازیم.



```
import { useRef, useState } from 'react';
import Editor from './pattern/editor.class';
import Caretaker from './pattern/caretaker.class';

const TextEditor: React.FC = () => {
  const [editorState, setEditorState] = useState<string>('');
  const [historyIndex, setHistoryIndex] = useState<number>(-1);

  const editorRef = useRef(new Editor());
  const caretakerRef = useRef(new Caretaker());

  const handleChange = (e: React.ChangeEvent<HTMLTextAreaElement>) => {
    const newState = e.target.value;
    setEditorState(newState);
    editorRef.current.setState(newState);
  };

  const saveState = () => {
    caretakerRef.current.addMemento(editorRef.current.saveStateToMemento());
    setHistoryIndex(caretakerRef.current['mementoList'].length - 1);
  };

  const restoreState = () => {
    if (historyIndex > -1) {
      const memento = caretakerRef.current.getMemento(historyIndex);
      editorRef.current.restoreStateFromMemento(memento);
      setEditorState(editorRef.current.getState());
    }
  };

  const undo = () => {
    if (historyIndex > 0) {
      setHistoryIndex(historyIndex - 1);
      const memento = caretakerRef.current.getMemento(historyIndex - 1);
      editorRef.current.restoreStateFromMemento(memento);
      setEditorState(editorRef.current.getState());
    }
  };

  return (
    <div>
      <textarea
        value={editorState}
        onChange={handleChange}
        rows={4}
        cols={50}
        placeholder="Type something...">
      </textarea>
      <div>
        <button onClick={saveState}>Save State</button>
        <button onClick={restoreState}>Restore State</button>
        <button onClick={undo}>Undo</button>
      </div>
      <><Current Editor State:></p>
      <pre>{editorState}</pre>
    </div>
  );
}

export default TextEditor;
```

Save State

Restore State

Undo

وضعیت جاری متنی که نوشته ایم را ذخیره می کند.

آخرین ذخیره سازی که انجام داده ایم را بازیابی می کند.

مراحل ذخیره سازی را از آخر به اول بازیابی می کند.

طرز استفاده از TextEditor

در پایان به شکل زیر از TextEditor استفاده می کنیم.



MementoDesignPattern

```
● ● ●  
  
import TextEditor from './TextEditor';  
  
const MementoDesignPattern = () => {  
  return (  
    <div>  
      <TextEditor />  
    </div>  
  );  
};  
  
export default MementoDesignPattern;
```