

Command

دیزاین پترن ها در React

Behavioral Patterns

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



Behavioral Patterns

Command

الگوی Command درخواست ها را به صورت یک شیء کپسوله می کند. این الگو به شما اجازه می دهد درخواست ها را به تأخیر بیندازید، به صورت پویا تعریف کنید یا امکان Undo را فراهم کنید. این الگو برای مدیریت تعاملات کاربر، تغییرات وضعیت و عملیات پیچیده بسیار مفید است.

فرض کنید می خواهیم یک Todo List بسازیم که قابلیت های افزودن، حذف و نمایش را داشته باشد. در این مثال قصد داریم که سناریو Undo و Redo را نیز پیاده سازی کنیم. همچنین جهت بهبود تجربه کاربری کلیدهای میانبر را نیز تعریف می کنیم.

بخش Interface

ابتدا دو Interface برای اینکه تمام دستورات باید از این رابط استفاده کند و title که مشخص می کند هر آیتمی که در لیست تعریف می کنیم باید دارای دو فیلد id و title باشد.



```
● ● ●  
export default interface Command {  
  execute(): void;  
  undo(): void;  
}
```



```
● ● ●  
export default interface Item {  
  id: string;  
  title: string;  
}
```

بخش CommandInvoker

این بخش که Invoker نامیده می شود مسئول اجرای دستورات است و می تواند تاریخچه دستورات را نیز نگهداری کند.



CommandInvoker

```
import Command from './interfaces/command.interface';
import Item from './interfaces/item.interface';

export default class CommandInvoker {
  private items: Item[] = [];
  public history: Command[] = [];
  public historyIndex: number = -1;

  get(): Item[] {
    return [...this.items];
  }

  executeCommand(command: Command): void {
    if (this.historyIndex < this.history.length - 1) {
      this.history = this.history.slice(0, this.historyIndex + 1);
    }
    command.execute();
    this.history.push(command);
    this.historyIndex++;
  }

  undo(): void {
    if (this.historyIndex >= 0) {
      this.history[this.historyIndex].undo();
      this.historyIndex--;
    }
  }

  redo(): void {
    if (this.historyIndex < this.history.length - 1) {
      this.historyIndex++;
      this.history[this.historyIndex].execute();
    }
  }
}
```

```
executeCommand(command: Command): void {
  if (this.historyIndex < this.history.length - 1) {
    this.history = this.history.slice(0, this.historyIndex + 1);
  }
  command.execute();
  this.history.push(command);
  this.historyIndex++;
}
```

بخش Utils

به جهت اینکه افزودن آیتم و حذف کردن آیتم هم در بخش Utils و هم در بخش RemovelItemCommand کاربرد دارد لذا منطق آن ها در این فایل قرار می دهیم.



```
import CommandInvoker from '../command-invoker';
import Item from '../interfaces/item.interface';

const utils = {
  insert: (invoker: CommandInvoker, item: Item) => invoker['items'].push(item),
  delete: (invoker: CommandInvoker, item: Item) =>
    (invoker['items'] = invoker['items'].filter((x) => x.id !== item.id)),
};

export default utils;
```

بخش AddItemCommand

این Command وظیفه افزودن آیتم به لیست را بر عهده دارد.



AddItemCommand

```
import { v4 as uuid } from 'uuid';
import CommandInvoker from '../command-invoker';
import ICommand from '../interfaces/command.interface';
import Item from '../interfaces/item.interface';
import utils from './utils';

export default class AddItemCommand implements ICommand {
  private item: Item;
  private invoker: CommandInvoker;

  constructor(todoList: CommandInvoker, title: string) {
    this.invoker = todoList;
    this.item = {
      id: uuid(),
      title: title,
    };
  }

  execute(): void {
    utils.insert(this.invoker, this.item);
  }

  undo(): void {
    utils.delete(this.invoker, this.item);
  }
}
```

بخش RemoveItemCommand

این Command وظیفه حذف کردن آیتم از لیست را بر عهده دارد.



RemoveItemCommand

```
import CommandInvoker from '../command-invoker';
import ICommand from '../interfaces/command.interface';
import Item from '../interfaces/item.interface';
import utils from './utils';

export default class RemoveItemCommand implements ICommand {
    private item: Item;
    private invoker: CommandInvoker;

    constructor(todoList: CommandInvoker, item: Item) {
        this.invoker = todoList;
        this.item = item;
    }

    execute(): void {
        utils.delete(this.invoker, this.item);
    }

    undo(): void {
        utils.insert(this.invoker, this.item);
    }
}
```

بخش useKeyboardShortcuts

این Hook را فقط به جهت یهیود تجربه کاربری ایجاد کرده ایم که کاربر بتواند با عملیات مورد نظر را انجام دهد و هیچ ربطی به منطق این الگوی طراحی ندارد.



useKeyboardShortcuts

```
import { useEffect } from 'react';

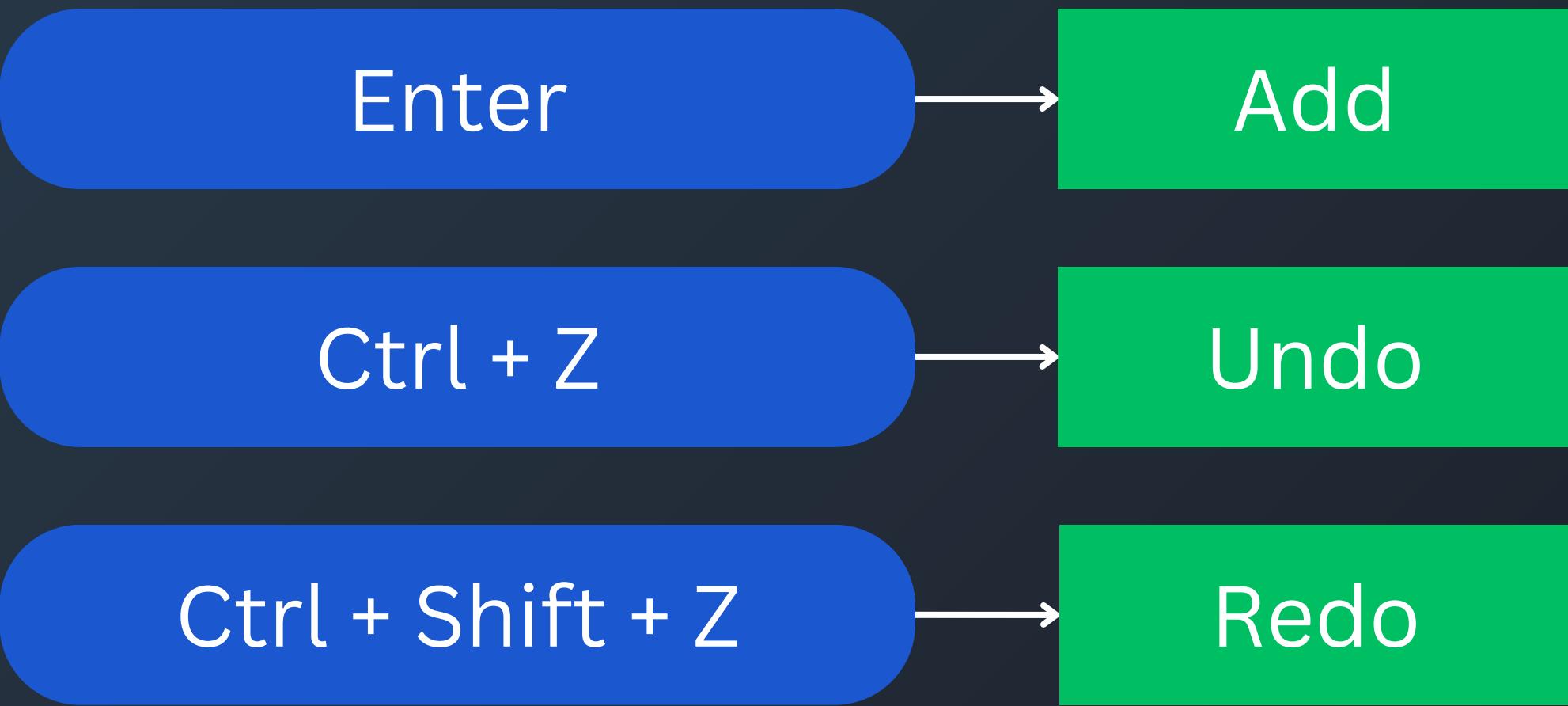
const useKeyboardShortcuts = (
  handleAdd: () => void,
  handleUndo: () => void,
  handleRedo: () => void
) => {
  useEffect(() => {
    const handleKeyDown = (event: KeyboardEvent) => {
      if (event.key === 'Enter') {
        handleAdd();
      }

      if (event.ctrlKey && event.key === 'z') {
        event.preventDefault();
        handleUndo();
      } else if (event.ctrlKey && event.shiftKey && event.key === 'Z') {
        event.preventDefault();
        handleRedo();
      }
    };

    window.addEventListener('keydown', handleKeyDown);

    return () => {
      window.removeEventListener('keydown', handleKeyDown);
    };
  }, [handleAdd, handleUndo, handleRedo]);
};

export default useKeyboardShortcuts;
```



بخش TodoList

در این بخش Component مورد نظر خود را به شکل زیر می سازیم.



TodoList

```
import React, { useState } from 'react';
import AddTodoCommand from './pattern/commands/add-item.command';
import RemoveTodoCommand from '../pattern/commands/remove-item.command';
import CommandInvoker from '../pattern/command-invoker';
import useKeyboardShortcuts from './useKeyboardShortcuts.hook';
import Item from '../pattern/interfaces/item.interface';

const TodoList: React.FC = () => {
  const [itemTitle, setItemTitle] = useState('');
  const [items, setItems] = useState<Item[]>([]);
  const [todoList] = useState(new CommandInvoker());

  const handleAdd = () => {
    if (!itemTitle.trim()) return;
    const addCommand = new AddTodoCommand(todoList, itemTitle);
    todoList.executeCommand(addCommand);
    setItems(todoList.get());
    setItemTitle('');
  };

  const handleRemove = (item: Item) => {
    const removeCommand = new RemoveTodoCommand(todoList, item);
    todoList.executeCommand(removeCommand);
    setItems(todoList.get());
  };

  const handleUndo = () => {
    todoList.undo();
    setItems(todoList.get());
  };

  const handleRedo = () => {
    todoList.redo();
    setItems(todoList.get());
  };

  useKeyboardShortcuts(handleAdd, handleUndo, handleRedo);

  return (
    <div>
      <input type='text' value={itemTitle} onChange={(e) => setItemTitle(e.target.value)} placeholder='Add new todo' />
      <button onClick={handleAdd}>Add Todo</button>
      <button onClick={handleUndo} disabled={todoList.historyIndex < 0}> Undo </button>
      <button onClick={handleRedo} disabled={todoList.historyIndex >= todoList.history.length - 1}> Redo </button>
    </div>
  );
}

export default TodoList;
```

```
import React, { useState } from 'react';
import AddTodoCommand from './pattern/commands/add-item.command';
import RemoveTodoCommand from '../pattern/commands/remove-item.command';
import CommandInvoker from '../pattern/command-invoker';
import useKeyboardShortcuts from './useKeyboardShortcuts.hook';
import Item from '../pattern/interfaces/item.interface';

const TodoList: React.FC = () => {
  const [itemTitle, setItemTitle] = useState('');
  const [items, setItems] = useState<Item[]>([]);
  const [todoList] = useState(new CommandInvoker());

  const handleAdd = () => {
    if (!itemTitle.trim()) return;
    const addCommand = new AddTodoCommand(todoList, itemTitle);
    todoList.executeCommand(addCommand);
    setItems(todoList.get());
    setItemTitle('');
  };

  const handleRemove = (item: Item) => {
    const removeCommand = new RemoveTodoCommand(todoList, item);
    todoList.executeCommand(removeCommand);
    setItems(todoList.get());
  };

  const handleUndo = () => {
    todoList.undo();
    setItems(todoList.get());
  };

  const handleRedo = () => {
    todoList.redo();
    setItems(todoList.get());
  };

  useKeyboardShortcuts(handleAdd, handleUndo, handleRedo);

  return (
    <div>
      <input type='text' value={itemTitle} onChange={(e) => setItemTitle(e.target.value)} placeholder='Add new todo' />
      <button onClick={handleAdd}>Add Todo</button>
      <button onClick={handleUndo} disabled={todoList.historyIndex < 0}> Undo </button>
      <button onClick={handleRedo} disabled={todoList.historyIndex >= todoList.history.length - 1}> Redo </button>
      <ul>
        {items.map((item, index) => (
          <li key={index}>
            {item.title}
            <button onClick={() => handleRemove(item)}>Remove</button>
          </li>
        ))}
      </ul>
    </div>
  );
}

export default TodoList;
```

طرز استفاده از TodoList

در پایان به شکل زیر از TodoList استفاده می کنیم.



CommandDesignPattern

```
import TodoList from './TodoList/TodoList';

const CommandDesignPattern = () => {
  return (
    <div>
      <TodoList />
    </div>
  );
};

export default CommandDesignPattern;
```