

Iterator

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

Iterator



الگوی Iterator به شما اجازه می دهد عناصر یک مجموعه را به صورت ترتیبی و بدون آشکار کردن جزئیات داخلی آن مرور کنید و باعث جداسازی منطق پیمایش از مجموعه داده می شود، از این الگویی توان برای پیمایش لیست های مجموعه های داده در رابط کاربری استفاده کرد.

فرض کنید می خواهیم یک مجموعه را به شکلی پیمایش کنیم که منطق پیمایش را از کد اصلی جدا کنیم و به سادگی روی انواع مختلف مجموعه ها این کار را انجام دهیم لذا نیاز به استفاده از یک متده عمومی و واحد برای این کار داریم.

بخش Interface

در این بخش یک Interface به نام Iterator می سازیم که دارای دو Method به نام های next و hasNext که مشخص می کند آیا مجموعه آیتم بعدی دارد یا خیر.



Iterator



```
export default interface Iterator<T> {  
    next(): T | null;  
    hasNext(): boolean;  
}
```

بخش Interface

در این بخش یک Interface به نام Aggregate می سازیم که برای مجموعه ها کاربرد دارد و دارای یک Method به نام createIterator برای ایجاد Iterator می باشد.



Aggregate

```
● ● ●

import Iterator from './iterator.interface';

export default interface Aggregate<T> {
  createIterator(): Iterator<T>;
}
```

بخش ArrayIterator

در این کلاس Iterator و منطق پیمایش مجموعه را پیاده سازی کرده ایم.



ArrayIterator

```
import Iterator from './interfaces/iterator.interface';

export default class ArrayIterator<T> implements Iterator<T> {
  private collection: T[];
  private position: number = 0;

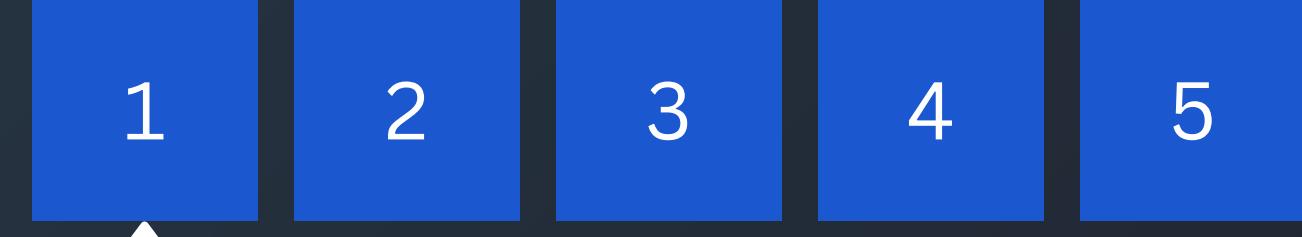
  constructor(collection: T[]) {
    this.collection = collection;
  }

  next(): T | null {
    if (this.hasNext()) {
      return this.collection[this.position++];
    }
    return null;
  }

  hasNext(): boolean {
    return this.position < this.collection.length;
  }
}
```

hasNext()

True



- هر بار که hasNext صدا زده می شود به ما می گوید که آیا آیتم بعدی در مجموعه وجود دارد یا خیر.
- هر بار که next صدا زده می شود در مجموعه یک خانه به جلو حرکت می کنیم.

بخش NumberCollection

در این کلاس Aggregate را پیاده سازی و از Iterator برای پیمایش مجموعه استفاده کرده ایم.



NumberCollection

```
import ArrayIterator from './array-iterator';
import Aggregate from './interfaces/aggregate.interface';
import Iterator from './interfaces/iterator.interface';

export default class NumberCollection implements Aggregate<number> {
    private numbers: number[];

    constructor(numbers: number[]) {
        this.numbers = numbers;
    }

    createIterator(): Iterator<number> {
        return new ArrayIterator(this.numbers);
    }
}
```

بخش NumberList

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

NumberList

```
● ● ●

import { useState } from 'react';
import NumberCollection from './pattern/number-collection';

const NumberList: React.FC = () => {
  const [numbers] = useState([1, 2, 3, 4, 5]);
  const collection = new NumberCollection(numbers);
  const iterator = collection.createIterator();

  const renderedNumbers: JSX.Element[] = [];
  while (iterator.hasNext()) {
    const item = iterator.next();
    renderedNumbers.push(<li key={item}>{item}</li>);
  }

  return (
    <div>
      <ul>{renderedNumbers}</ul>
    </div>
  );
};

export default NumberList;
```

طرز استفاده از NumberList

در پایان به شکل زیر از NumberList استفاده می کنیم.



IteratorDesignPattern



```
import NumberList from './NumberList';

const IteratorDesignPattern = () => {
  return (
    <div>
      <NumberList />
    </div>
  );
};

export default IteratorDesignPattern;
```