# Iterator Pattern

기술 고문 켄트 포디올로리스(Kent Podiololis)가 C 스타일의 반복문 사용에 대해 불평한다.

“ 우리가 아직도 1980년대에 살고 있는 건가요? ” — 켄트

패드로: 자바의 Iterator 인터페이스를 사용하면 돼요.

이브: 놀리지 말아요. 아무도 java.util.Iterator를 사용하고 있지 않아요.

페드로: 모든 사람이 for-each 루프 구문에서 간접적으로 그것을 사용해요. 그것은 컨테이너를 순회(traverse)하는 좋은 방법이예요.

이브: 컨테이너를 순회한다는 것이 무슨 의미죠?

페드로: 컨테이너는 공식적으로 두 개의 메소드를 제공해야 해요. 즉, next() 메소드는 다음 요소를 반환하고, hasNext() 메소드는 컨테이너가 더 많은 요소를 갖고 있으면 true를 반환해야 하지요.

이브: 좋아요. 그런데 혹시 연결 리스트(linked list)가 무엇인지 아시나요?

페드로: 단일 연결 리스트(Singly linked list) 말씀하시는 건가요?

이브: 맞아요.

페드로: 물론이죠. 그것은 노드들로 구성된 컨테이너죠. 각 노드는 데이터 값과 다음 노드의 레퍼런스를 갖고 있어요. 다음 노드가 없으면 null 값을 갖게 되고요.

이브: 맞아요. 그럼 그런 리스트를 순회하는 것과, 반복자(iterator)를 통해 순회하는 것 사이에 차이가 있을까요?

페드로: 음…

페드로는 순회하는 코드 두 개를 작성한다.

- 반복자를 통해 순회하기

```
Iterator i;
while (i.hasNext()) {
  i.next();
}
```

- 연결 리스트를 이용해 순히하기

```
Node next = root;
while (next != null) {
  next = next.next;
}
```

페드로: 그러고 보니 둘이 아주 비슷하네요… 클로저에서는 반복자에 해당하는 것이 무엇인가요?

이브: seq 함수입니다.

```
(seq [1 2 3])       => (1 2 3)
(seq (list 4 5 6))  => (4 5 6)
(seq #{7 8 9})      => (7 8 9)
(seq (int-array 3)) => (0 0 0)
(seq "abc")         => (\a \b \c)
```

페드로: 리스트를 반환하네요…

이브: 정확히는 시퀀스예요. 반복자는 단순히 시퀀스이거든요.

페드로: 사용자 자료구조도 seq 함수의 인수로 들어갈 수 있나요?

이브: clojure.lang.Seqable 인터페이스를 구현하면 가능해요.

```
(deftype RedGreenBlackTree [& elems]
  clojure.lang.Seqable
  (seq [self]
    ;; traverse element in needed order
    ))
```

페드로: 그렇다면 좋네요. 그런데 저는 시퀀스가 지연 평가될 수 있다고 들었어요. 예를 들면 getNext()를 호출할 때에만 값을 계산하게 한다든지 하는 식으로요. 리스트로 어떻게 그것을 처리하나요?

이브: 리스트는 지연 평가될 수 있어요. 클로저에서는 그런 리스트를 “ 지연 시퀀스 ” 라고 불러요.

```
(def natural-numbers (iterate inc 1))
```

이브: 위의 정의로 모든 자연수를 표현할 수 있어요. 하지만 아무런 값도 아직 요청하고 있지 않아서, OutOfMemory가 나지는 않아요.

페드로: 조금 더 설명해 주실 수 있겠어요?

이브: 유감스럽게도, 그러기에는 제가 너무 게으르네요(lazy).

페드로: 알겠습니다!

---

# 관련 문서

- Clojure Design Patterns