Lazy & Eager Collections en JavaScript

IIC3585 Diseño Avanzado de Aplicaciones Web

Segundo semestre 2016

Martín Fuenzalida
Franco Muñoz

Índice

1. Lazy e Eager evaluation

- Estrategias de evaluación
  - **Eager** (ansioso): resolver inmediatamente
  - **Lazy**: resolver cuando es necesario

- Ejemplo:

```
1  function f(x) {
2    return x + 1
3  }
4
5  print(f(2 + 3))
```

Eager evaluation:
- Resuelve `2 + 3 = 5`
- Llama `f(5)`
  - Resuleve `x + 1 = 5 + 1 = 6`
  - Retorna `6`
- Llama `print(6)`: muestra resultado

Lazy evaluation:
- Llama `f(2 + 3)`
  - Retorna `(2 + 3) + 1`
- Llama `print((2 + 3) + 1)`
  - Resuelve `(2 + 3) + 1`
  - - Resuelve `(2 + 3) = 5`
  - - Resuelve `5 + 1 = 6`
  - Muestra 6

1. Lazy e Eager evaluation

- Estrategias de evaluación
  - **Eager** (ansioso): resolver inmediatamente
  - **Lazy**: resolver cuando es necesario

- Ejemplo:

```
1  function f(x) {
2    return x + 1
3  }
4
5  print(f(2 + 3))
```

- Una **lazy collection** es una estructura que solo calcula un elemento cuando este es necesario
  - No mantiene en memoria todos los elementos
- Nos centramos en **lazy collections**

1. Lazy e Eager evaluation

- Lenguajes funcionales con lazy evaluation:
  - **Haskell** (1990), Clean(1987) y Miranda (1985)
- Muchos lenguajes permiten evaluar o emular lazy evaluation
  - Ejemplo F#

```
1  let identifier = lazy ( expression )
```

  - **Generadores**
- Algunas ventajas y desventajas:
  - A veces más eficiente
  - Menos controlable
  - Requiere guardar operaciones
- Lazy evaluation se usa muchas veces con **memoización**

2. Algunos casos de uso

- Streams de datos
- Colecciones potencialmente “infinitas”

```
1  naturals = [1..]
```

- Interfaces gráficas. Ejemplo: Gecko (Firefox 4)
- Optimizaciones
  - Ejemplo, Python 3.x `range` vs Python 2.x `range`

```
1  >>> r = range(10)
2  >>> print r
3  [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
4  >>> print r[3]
5  3
```

(Python 2.x)

```
1  >>> r = range(10)
2  >>> print(r)
3  range(0, 10)
4  >>> print(r[3])
5  3
```

(Python 3.x)

3.1 Lazy collections en **JavaScript** - Librerías

- Muchas característica de la programación funcional no son explotadas hasta ES6
- Sin ES6, se usan liberías para poder usar estas funcionalidades.
- Para lazy collections tenemos:
  - **Lazy.js**
  - stream.js

3.2 Lazy collections en **JavaScript** - ES6

- ES6 introduce generadores
  - Permiten encapsular el computo de una expresión
  - Podemos aprovechar esto para hacer lazy evaluation de forma más natural

```
1  function* naturalNumbers() {
2    let i = 0;
3    while(true) {
4      yield i++;
5    }
6  }
7
8  console.log(naturalNumbers()); // Object {value: 0, done: false}
9  console.log(naturalNumbers()); // Object {value: 1, done: false}
10 console.log(naturalNumbers()); // Object {value: 2, done: false}
11 console.log(naturalNumbers()); // Object {value: 3, done: false}
```

4. Demo

Links de interés

- Lazy and eager evaluation:
  - https://en.wikipedia.org/wiki/Eager_evaluation
  - https://en.wikipedia.org/wiki/Lazy_evaluation
- Firefox 4: Better performance with Lazy Frame Construction
- ES6 Generators
- Lazy.js
- Working with infinite sequences in javascript ES6