

## Play Framework – Enumerator.outputStream

November 12, 2012

#playframework #iteratee

A few weeks ago, we've introduced a new feature in Play Framework: the `Enumerator.outputStream` method, allowing you to work with Java API requiring an `OutputStream` to generate content, for instance the `java.util.zip` API.

**Now, let's see how easy it is to serve a big Zip generated on-the-fly without memory load with Play Framework.**

### The Zip generation example

```
1 package controllers
2
3 import play.api._
4 import play.api.mvc._
5
6 object Application extends Controller {
7
8   def zip = Action {
9     import play.api.libs.iteratee._
10    import java.util.zip._
11
12    val r = new java.util.Random()
13
14    val enumerator = Enumerator.outputStream { os =>
15      val zip = new ZipOutputStream(os);
16      Range(0, 100).map { i =>
17        zip.putNextEntry(new ZipEntry("test-zip/README-"+i+".txt"))
18        zip.write("Here are 100000 random numbers:\n".map(_._1.toByte).toArray)
19        // Let's do 100 writes of 1'000 numbers
20        Range(0, 100).map { j =>
21          zip.write((Range(0, 1000).map(_ => r.nextLong).map(_._1.toString).mkString("\n")).map(_._1.toB
22        }
23        zip.closeEntry()
24      }
25      zip.close()
26    }
27    Ok.stream(enumerator >>> Enumerator.eof).withHeaders(
28      "Content-Type" -> "application/zip",
29      "Content-Disposition" -> "attachment; filename=test.zip"
```

```

30     )
31   }
32
33   def index = Action {
34     Ok(views.html.index("Your new application is ready."))
35   }
36
37 }

```

Application.scala hosted with ❤ by GitHub

[view raw](#)

This demo shows how to **generate a zip file on-the-fly** and directly **stream it** to an HTTP client **without loading it in memory or storing it in a file**.

It uses an `Enumerator` created with the `Enumerator.outputStream` method. The `OutputStream` provided by the method is then plugged to the Java's `ZipOutputStream`.

For the example, we have generated a zip containing 100 text files, and each text files contains 100'000 random long numbers (yes, 100'000 !).

The zip size is approximatively 100 Mb. (and is generated in about 3Mb/s in my machine in localhost, but this can be improved)

The huge benefit of this is the download starts instantly, it means the Zip is streamed while it is generated.

## Show me the code!

Internally, it is implemented with a `Concurrent.unicast`, and a simple implementation of an `OutputStream` pushing into the unicast's channel:

```

1  /** Create an Enumerator of bytes with an OutputStream.
2   */
3  def outputStream(a: java.io.OutputStream => Unit): Enumerator[Array[Byte]] = {
4    Concurrent.unicast[Array[Byte]] { channel =>
5      val outputStream = new java.io.OutputStream(){
6        override def close() {
7          channel.end()
8        }
9        override def flush() {}
10       override def write(value: Int) {
11         channel.push(Array(value.toByte))
12       }
13       override def write(buffer: Array[Byte]) {
14         write(buffer, 0, buffer.length)
15       }
16       override def write(buffer: Array[Byte], start: Int, count: Int) {

```

```
17         channel.push(buffer.slice(start, start+count))
18     }
19 }
20 a(outputStream)
21 }
22 }
```

---

Enumerator.scala hosted with ❤ by GitHub

[view raw](#)

## About Iteratee and Enumerator

If you want to learn more about Iteratee concepts in Play Framework, I recommend you [this article](#).

---

### Last Posts

---

[gl-react v3](#)

[Relay, scrolling connections](#)

[🐛 There are some OpenGL in the Project September fashion app!](#)

[Universal GL Effects for Web and Native](#)

[Introducing gl-react](#)

[Making performant React applications](#)

[\[FR\] webglparis talk: GLSL.io initiative and WebGL Transitions](#)

[Cellular Automata in IBEX](#)

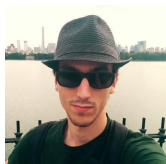
[IBEX, my js13k game](#)

[48 hours to prototype an Ant Sim Game](#)

### About the author

---

My name is Gaëtan Renaudeau ([@greweb](#))



I work at [ProjectSeptember](#) where I've developed [gl-react](#). I enjoy hacking technology, experimenting with HTML5 techs like WebGL and WebAudio. I develop HTML5 games for fun, usually in [ludumdare game jam](#).

I speak French, English and learn Chinese.

### Last Tweets

---

### lichess TV

---