# 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

```scala
package controllers

import play.api._
import play.api.mvc._

object Application extends Controller {

  def zip = Action {
    import play.api.libs.iteratee._
    import java.util.zip._

    val r = new java.util.Random()

    val enumerator = Enumerator.outputStream { os =>
      val zip = new ZipOutputStream(os);
      Range(0, 100).map { i =>
        zip.putNextEntry(new ZipEntry("test-zip/README-"+i+".txt"))
        zip.write("Here are 100000 random numbers:\n".map(_.toByte).toArray)
        // Let's do 100 writes of 1'000 numbers
        Range(0, 100).map { j =>
          zip.write((Range(0, 1000).map(_=>r.nextLong).map(_.toString).mkString("\n")).map(_.toB
        }
        zip.closeEntry()
      }
      zip.close()
    }
    Ok.stream(enumerator >>> Enumerator.eof).withHeaders(
      "Content-Type"->"application/zip",
      "Content-Disposition"->"attachment; filename=test.zip"
    )
  }

  def index = Action {
    Ok(views.html.index("Your new application is ready."))
  }

}
```

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:

```
/** Create an Enumerator of bytes with an OutputStream.
 */
def outputStream(a: java.io.OutputStream => Unit): Enumerator[Array[Byte]] = {
  Concurrent.unicast[Array[Byte]] { channel =>
    val outputStream = new java.io.OutputStream(){
      override def close() {
        channel.end()
      }
      override def flush() {}
      override def write(value: Int) {
        channel.push(Array(value.toByte))
      }
      override def write(buffer: Array[Byte]) {
        write(buffer, 0, buffer.length)
      }
      override def write(buffer: Array[Byte], start: Int, count: Int) {
        channel.push(buffer.slice(start, start+count))
      }
    }
    a(outputStream)
  }
}
```

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