Tapestry5HowToCreateAComponentEventResultProcessor

When Tapestry receives an Event Request an Event Handler is called. The return type of the event handler can determine what is sent to the browser. The information of default accepted return types can be found here: http://tapestry.apache.org/tapestry5/guide/pagenav.html

If you want to create a custom return type you have to do the following steps (This example creates a Return Type that represents a callback handler for directly streaming to the output):

1) Define the interface of the return type you want to use

In this case an interface similar to the a StreamResponse is generated with the difference that no InputStream is returned for request handling but instead a callback writes to an OutputStream

```
package at.ac.uibk.dbisinformatik.services;

import java.io.BufferedOutputStream;
import java.io.IOException;
import java.io.OutputStream;

import org.apache.tapestry5.StreamResponse;
import org.apache.tapestry5.services.Response;

/**
 * Callback for directly writing to the outputstream from an action method
 * This class was inspired by {@link StreamResponse}
 *
 * @author robert.binna@uibk.ac.at
 *
 */
public interface OutputStreamResponse {

    /**
     * Returns the content type to be reported to the client.
     */
    String getContentType();

    /**
     * Implements a callback to directly write to the output stream.
     * The stream will be closed after this method returns.
     * The provided stream is wrapped in a {@link BufferedOutputStream} for efficiency.
     */
    public void writeToStream(OutputStream out) throws IOException;

    /**
     * Prepares the response before it is sent to the client. This is the place to set any response headers (e.
g.
     * content-disposition).
     *
     * @param response Response that will be sent.
     */
    void prepareResponse(Response response);

}
```

2) Define the ComponentEventResultProcessor that is capable of handling the type created in step 1

For our example the ComponentEventResultProcessor only passes the OutputStream of the Response to the Callback.

```
package at.ac.uibk.dbisinformatik.services;

import java.io.BufferedOutputStream;
import java.io.IOException;
import java.io.OutputStream;

import org.apache.tapestry5.internal.services.StreamResponseResultProcessor;
import org.apache.tapestry5.services.ComponentEventResultProcessor;
import org.apache.tapestry5.services.Response;

/**
 * ComponentEventResultProcessor that enables EventHandlers to return Callbacks
 * that can stream arbitrary data to the client.
 *
 * The class was inspired by {@link StreamResponseResultProcessor}
 *
 * @author robert.binna@uibk.ac.at
 */
public class OutputStreamResponseResultProcessor implements
                ComponentEventResultProcessor<OutputStreamResponse> {

    private static final int BUFFER_SIZE = 5000;

    private final Response response;

    public OutputStreamResponseResultProcessor(Response response) {
            this.response = response;
    }

    /**
     * Handles OutputStreamResponse
     *
     * @param Callback for streaming arbitrary data to the client using the response {@link OutputStream}
     *
     * @see ComponentEventResultProcessor#processResultValue(Object)
     */
     @Override
     public void processResultValue(OutputStreamResponse streamResponse)
                throws IOException {
        OutputStream out = null;
        try {
                streamResponse.prepareResponse(response);
                out = new BufferedOutputStream(response.getOutputStream(streamResponse.getContentType()),
BUFFER_SIZE);
                streamResponse.writeToStream(out);
                out.flush();
                out.close();
                out = null;
        } finally {
                if(out != null) { //can only be the case if an Exception was thrown because out was set to null
before
                        try {
                                out.close();
                        } catch(IOException ioe) {
                                //ignores this IO exception because an exception is already on the way
                        }
                }
        }
     }

}
```

3) Register the newly created ComponentEventResultProcessor in your Module (e.g. AppModule)

```
 /**
  * Adds ComponentEventResultProcessors
  *
  * @param configuration the configuration where new ComponentEventResultProcessors are registered by the type
they are processing
  * @param response the response that the event result processor handles
  */
  public void contributeComponentEventResultProcessor(MappedConfiguration<Class<?>,
ComponentEventResultProcessor<?>> configuration, Response response)
  {
    configuration.add(OutputStreamResponse.class, new OutputStreamResponseResultProcessor(response));
  }
```