Accessing Runtime Data

If you are embedding an Elixir Repertoire Engine into your application, you can easily access application data from within the Engine. JavaScript scripts are able to invoke any method within the Java VM, so you can use a simple Singleton to set data for use by the Engine.

Here is a simple class:

```
package sample;

import java.util.HashMap;

import java.util.Map;

public class Holder

{
 public static synchronized String setData(Object obj)
 {
  String key = "Holder."+m_NextId;
  ++m_NextId;
  m_Data.put(key,obj);
  return key;
 }
 public static synchronized Object getData(String key)
 {
  return m_Data.get(key);
 }
 public static synchronized void releaseData(String key)
 {
  m_Data.remove(key);
 }
 private static Map<String,Object> m_Data = new HashMap<String,Object>();
 private static int m_NextId;
}
```

Your application code can set any kind of object or collection into the Holder using:

```
String key = Holder.setData(myDataCollection);
```

You can pass the key that is returned into the Engine as a dynamic parameter. From within any script, such as an Object DataSource or RenderIf you can call:

```
var obj = Packages.sample.Holder.getData(key);
```

using the key that you obtained via the dynamic parameter.

Finally, when control returns to your application code, after calling **renderReport**, you can execute

Holder.releaseData(key);

Variations

- You can avoid passing a key at all if you are running only a single Engine thread – do not use a map, just hold one object.
- You can use a ThreadLocal to avoid passing a key when running multiple threads – do not use a map, just hold one object in a ThreadLocal.