VU Storage

Ent SP

All extensions (scriptable variables and components) have access to a VU storage component that can pass seamlessly pass data between extensions.

Accessing the storage

Scriptable variables can access the VU storage through the **SessionContext.SetObject()** and **SessionContext.GetObject()** methods.

Components can access the VU storage through the **RuntimeVU.SetObject(), RuntimeVU.GetObject(), RuntimeSession.SetObject(),** and **RuntimeSession.GetObject()** methods.

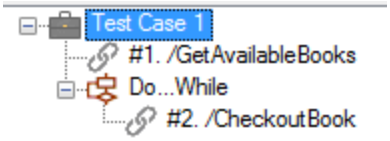
Example

Suppose a test case with 2 requests:

- /GetAvailableBooks returns a list of available books
- /CheckoutBook checks out a book

The requirement is to call /GetAvailableBooks to get a list of books, and then for each available book, call /CheckoutBook

The strategy is to call /GetAvailableBooks, then create a Do...While loop around /CheckoutBook as shown below

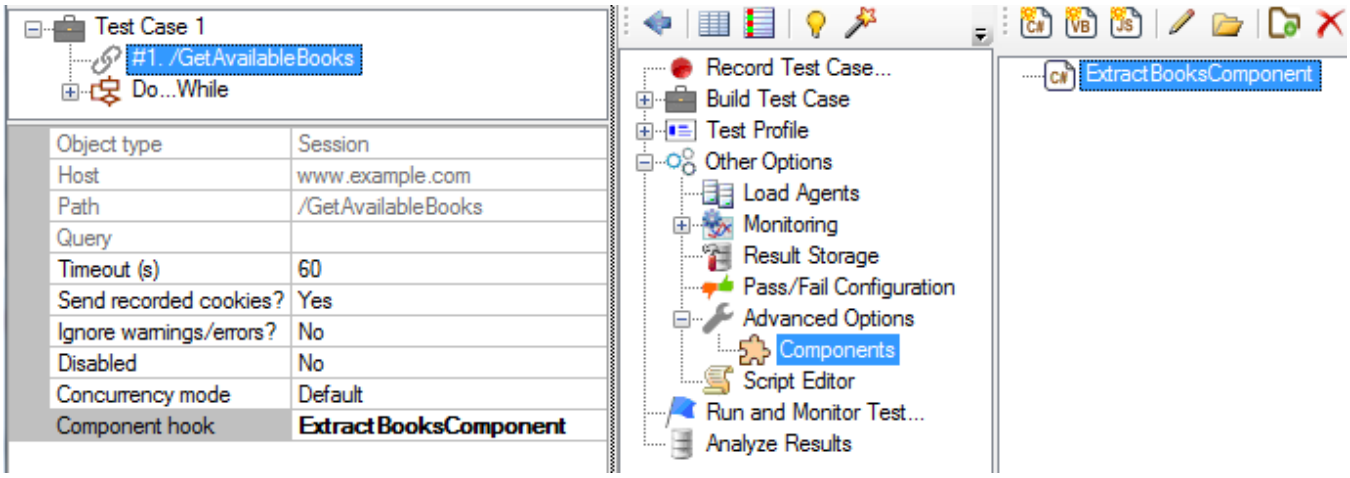


The first step is to extract all the books from the /GetAvailableBooks response into a list and store the list in the VU storage under the key *ListOfBooks*. We will create a component for that:

Extract all books

```
class ExtractBooksComponent : IExternalComponent2
{
    //---- Implement IExternalComponent2

    /// <summary>
    /// Fired after a response is received.
    /// </summary>
    /// <param name="session">The session object</param>
    public void OnAfterResponse(RuntimeSession session)
    {
                var listOfBooks = new List<string>();

                var body = session.GetResponseBody();

                var regex = new System.Text.RegularExpressions.Regex("{some regular expression}");

                //--- regex.Matches(body) ... Match all books and add them to listOfBooks

                session.SetObject("ListOfBooks", listOfBooks);
    }
}
```

Set the **Component hook** property of the request to *ExtractBooksComponent*

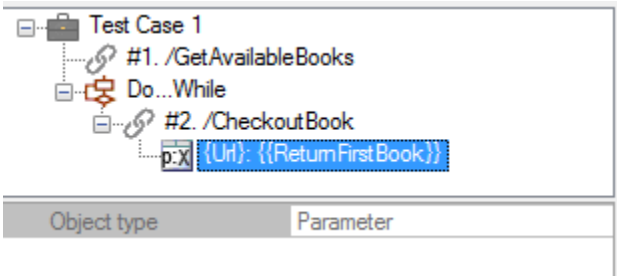


The next step is to create a scriptable variable, *ReturnFirstBook,* that will be used to parameterize /CheckoutBook request. We will create a scriptable variable that will read the VU storage for the list of available books, remove the first one, and return it.

Return first book

```
class ReturnFirstBook : IExternalVariable
{
    public ReturnFirstBook()
    {
        //Do not edit this section.
    }

    #region IExternalVariable Members

    /// <summary>
    /// Return true if the variable will be evaluated once per VU iteration. Otherwise will be evaluated on
every parameter.
    /// </summary>
    bool IExternalVariable.EvaluateOnIteration
    {
        get
        {
            return false;
        }
    }

    /// <summary>
    /// Return a value for the given session context.
    /// </summary>
    /// <param name="session">The session context object of the request consuming the variable.</param>
    /// <returns>The source variable value.</returns>
    string IExternalVariable.GetValue(SessionContext context)
    {
                var listOfBooks = context.GetObject("ListOfBooks") as List<string>;

                string book = null;

                if (listOfBooks.Count > 0)
                {
                        book = listOfBooks[0];
                        listOfBooks.RemoveAt(0);
                }
                return book;
    }

    #endregion
}
```

Then use the scriptable variable as a parameter in */CheckoutBook*.



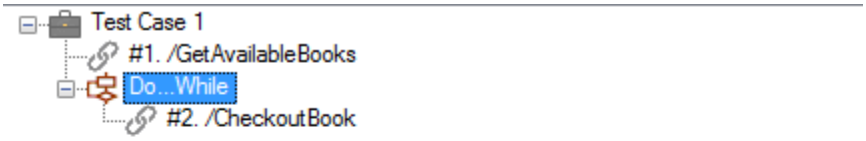
Finally, we will add a scriptable variable that will return the number of available books remaining that have not yet been checked.

Any books left?

```
using System;
using StresStimulus.Extensibility;
using System.Data;
using System.Collections.Generic;
using System.Linq;

class NumberOfBooksLeft : IExternalVariable
{
    public NumberOfBooksLeft()
    {
        //Do not edit this section.
    }

    /// <summary>
    /// Return true if the variable will be evaluated once per VU iteration. Otherwise will be evaluated on
every parameter.
    /// </summary>
    bool IExternalVariable.EvaluateOnIteration
    {
        get
        {
            return false;
        }
    }

    /// <summary>
    /// Return a value for the given session context.
    /// </summary>
    /// <param name="session">The session context object of the request consuming the variable.</param>
    /// <returns>The source variable value.</returns>
    string IExternalVariable.GetValue(SessionContext context)
    {
        var listOfBooks = context.GetObject("ListOfBooks") as List<string>;

                return listOfBooks.Count.ToString();
    }
}
```

In the Do...While loop set the following properties:

- Condition type: Match scriptable variable
- Scriptable variable to match: NumberOfBooksLeft
- Text-to-match: 0
- Exit loop if match found: Yes



Object type	While
Description	
Number of repeats (Max.)	**100**
Delay before next loop (s)	0
Condition type	**Match scriptable variable**
Scriptable variable to match	**NumberOfBooksLeft**
Text-to-match	**0**
Exit loop if match found?	Yes
Action on loop completion after max. repeats	Continue
Increment the index for iteration-bound datasets?	No