EXTENDING EA WITH CUSTOM SCRIPTS TO CATER FOR SPECIFIC NEEDS AND USER REQUESTS

LONDON EAUG 2018

Guillaume FINANCE

Blog: www.umlchannel.com
Add-in: www.eautils.com
Twitter: @umlchannel
01. INTRODUCTION
02. EA SCRIPTS OVERVIEW
03. SCRIPTS HELPERS
04. EA-MATIC TRIGGERED SCRIPTS
05. DEBUG & HINTS
06. DEMONSTRATION
INTRODUCTION
EA tool customization

Model-based approach pillars

Tool customization
INTRODUCTION
EA tool customization

Custom project structure

Stereotypes/UML Profiles, toolbars

Custom searches
INTRODUCTION

EA tool customization

Custom EA User Interface (13.5)

New features using EA API...

EA Scripts
Why do we need scripts?

- Replace repetitive tasks performed by users
- Work with Excel
  - Custom Excel publication
  - Update details in Excel
- Custom tools and solutions integration
- Model QA
- And much more!
Why do we need scripts?

**INTRODUCTION**

We need scripts to ensure consistency and maintain control over elements such as stakeholders, client... and third-party tools. In the internal use, scripts help ESA users (analysts, architects...) and manage method and processes, model consistency...
INTRODUCTION

Alternatives in using EA API

**EA API (Interop.EA)**

- **EA Scripts**
  - None - import file in the project, available for all users

- **EA add-ins**
  - Message or Input box, system output...

- **3rd party tools, ext. scripts**

- **Excel macros**

**Language(s)**

- **VBScript, JavaScript, JScript**
INTRODUCTION

Alternatives in using EA API

- **EA Scripts**
  - None - import file in the project, available for all users
- **EA add-ins**
  - Installation on all clients, available to use with any project
- **3rd party tools, ext. scripts**
- **Excel macros**

**Installation**

**User Interface**

**Language(s)**

- VBScript, JavaScript, JScript
- C#, VB.Net, Java, C++...
INTRODUCTION

Alternatives in using EA API

<table>
<thead>
<tr>
<th>EA DATA</th>
<th>EA Scripts</th>
<th>EA add-ins</th>
<th>3rd party tools, ext. scripts</th>
<th>Excel macros</th>
</tr>
</thead>
</table>

- **Installation**:
  - None - import file in the project, available for all users
  - Installation on all clients, available to use with any project
  - Optional

- **User Interface**:
  - Message or Input box, system output...
  - Enhanced windows, system output...
  - Linked with the chosen language

- **Language(s)**:
  - VBScript, JavaScript, JScript
  - C#, VB.Net, Java, C++...
  - C#, VB.Net, Java, C++, Python, Perl...
INTRODUCTION
Alternatives in using EA API

<table>
<thead>
<tr>
<th>EA Scripts</th>
<th>EA add-ins</th>
<th>3rd party tools, ext. scripts</th>
<th>Excel macros</th>
</tr>
</thead>
<tbody>
<tr>
<td>None - import file in the project, available for all users</td>
<td>Installation on all clients, available to use with any project</td>
<td>Optional</td>
<td>n/a (Excel required)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Installation</th>
<th>User Interface</th>
<th>Language(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>None - import file in the project, available for all users</td>
<td>Message or Input box, system output...</td>
<td>VBScript, JavaScript, JScript</td>
</tr>
<tr>
<td>Installation on all clients, available to use with any project</td>
<td>Enhanced windows, system output...</td>
<td>C#, VB.Net, Java, C++...</td>
</tr>
<tr>
<td>Optional</td>
<td>Linked with the chosen language</td>
<td>C#, VB.Net, Java, C+++, Python, Perl...</td>
</tr>
<tr>
<td>n/a (Excel required)</td>
<td>Excel</td>
<td>Excel VBA</td>
</tr>
</tbody>
</table>
INTRODUCTION
Alternatives in using EA API

**EA Scripts**
- None - import file in the project, available for all users

**EA add-ins**
- Installation on all clients, available to use with any project
- Optional

**3rd party tools, ext. scripts**
- n/a (Excel required)

**Excel macros**
- Excel

**Installation**
- Message or Input box, system output...
- Enhanced windows, system output...
- Linked with the chosen language

**User Interface**
- C#, VB.Net, Java, C++...
- C#, VB.Net, Java, C++, Python, Perl...

**Language(s)**
- VBScript, JavaScript, JScript
- Excel VBA

**Web applications**
- Custom tools integration

**EA DATA**
- Enterprise Architect
- EA API (Interop.EA)
- EA add-ins
- 3rd party tools, ext. scripts
- Excel macros
- EA Scripts
## INTRODUCTION

Alternatives in using EA API

<table>
<thead>
<tr>
<th>Installation</th>
<th>User Interface</th>
<th>Language(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>None - import file in the project, available for all users</td>
<td>Message or Input box, system output...</td>
<td>VBScript, JavaScript, JScript</td>
</tr>
<tr>
<td>Installation on all clients, available to use with any project</td>
<td>Enhanced windows, system output...</td>
<td>C#, VB.Net, Java, C++...</td>
</tr>
<tr>
<td>Optional</td>
<td>Linked with the chosen language</td>
<td>C#, VB.Net, Java, C++, Python, Perl...</td>
</tr>
<tr>
<td>n/a (Excel required)</td>
<td>Excel</td>
<td>Excel VBA</td>
</tr>
</tbody>
</table>

Compatible with frequent releases for a large group of users

- **EA Scripts**
- **EA add-ins**
- **3rd party tools, ext. scripts**
- **Excel macros**

---

<table>
<thead>
<tr>
<th>Web applications</th>
<th>Custom tools integration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pro Cloud Server</td>
<td>OSLC RESTful API</td>
</tr>
</tbody>
</table>
EA SCRIPTS OVERVIEW

Script Group Types

A script is defined under a group

- Main group types
  - Scripts under a **Normal Group** are available from the Scripting view
  - Scripts under a **Project Browser Group** are available from the Scripting view or via a right click in the Project Browser
  - Scripts under a **Diagram Group** are available from the Scripting view or via a right click in a diagram
Use Groups to organize your scripts

Example:

Reserved for the scripts administrator/dev team

Libraries of reusable functions and routines

Users scripts, available from a diagram or the Project Browser
EA SCRIPTS OVERVIEW

Naming conventions

Define prefixes to group and identify script functions

Example:

- **QA** = Quality related (QA-1 = cleaning, QA-2 = stats, QA-3 = controls)
- **SEC** = Users, groups and permissions
- **TSK** = Automated tasks (TSK-1 = generate, TSK-2 = migrate, TSK-3 = update)
- **XLS** = Excel related scripts (XLS-1 = import/export, XLS-2 = others)
- **XT** = Extract EA information

New order
EA SCRIPTS OVERVIEW

Re-used script functions

Project Browser and Diagram scripts + Common script

Calls processDiagram with the current diagram

«Diagram script»
XLS-1-e-Excel Export Class Details

«Project Browser script»
XLS-1-c-Excel Export Class Details from Branch

Calls processPackage with the current package

«Normal script»
ExportClassDetails-Common

processPackage()
processDiagram()
exportClassElements()

Exports details of EA Elements ...
1. Located in the provided package (processPackage)
2. Or matching Diagram Objects in the provided Diagram List (processDiagram)
Both call exportClassElements private function

Re-used script

1 option explicit
2 !INC 1-ScriptsLib.ExportClassDetails-Common
EA Automation Interface provides a way of accessing the internals of Enterprise Architect models.

**EA API Methods**

**EA API Object Model**

- **Repository**
  - Provides an entry point to the model packages and collections.

- **Package**

- **Element**, **Connector**, **Diagram**, etc.

**Using Object methods - VBScript examples:**

```vbnet
set curElement = Repository.GetElementByID(selectedDiagObject.ElementID): element matching a diagram object

set parentPkg = Repository.GetPackageByID(curElement.PackageID): element’s parent package
```
EA SCRIPTS OVERVIEW
Run SQL queries against the model

SQL queries can be used …

- When a query or update cannot be achieved via a method.

- To select specific elements, preventing time consuming loops.
  e.g. select *datatypetype stereotyped Classes* with the *name* matching the *given variable*;

  ```sql
  SELECT Object_ID FROM t_object WHERE (Object_Type = 'Class' and (Stereotype = 'datatype'))
  and name = "&attributeType&"
  ```

- To address performance issues related. Example with a bulk creation of ~1000 attributes:

  API method call execution time: 120 minutes
  ```csharp
  set elementAttr = elt.Attributes.AddNew("&attrName&",""
  elementAttr.Update()
  ```

  API method call with a SQL insert execution time: 4 minutes
  ```sql
  insert into t_attribute(object_id, name, ea_guid) values ...
  ```
EA SCRIPTS OVERVIEW

Use SQL queries

Repository object available methods:

```
11 sub main
12 Dim SQL, className
13 Dim elements As EA.Collection
14 Dim matchingElement As EA.Element
15 className = "Demo"
16 SQL = "select object id from t object where Object_Type='Class' and Name='Demo'"
17 set elements = Repository.GetElementSet(SQL,2)
18 Session.Output "Number of classes found: " & elements.Count
19 for each matchingElement in elements
20 Session.Output "Class found: " & matchingElement.Name & " | " & matchingElement.ElementGUID
21 next
22 end sub
```
Use SQL queries

Repository object available methods:

- `Repository.GetElementSet(query, 2)`: returns a collection of EA elements according to the SQL query
- `Repository.SQLQuery(query)`: returns an XML formatted string value (SELECT queries only)

```vba
sub main
    Dim SQL, className
    Dim queryResult
    Dim watchingElement As EA.Element
    className = "Demo"
    SQL = "select t_attribute.ea_guid, t_object.Name as ClassName, t_attribute.Name, t_attribute.Type "
    SQL = SQL & "from t_attribute inner join t_object on t_attribute.Object_ID = t_object.Object_ID "
    SQL = SQL & "where t_object.Object_Type='Class' and t_object.Name='" & className & ""
    queryResult = Repository.SQLQuery(SQL)
    Session.Output queryResult
end sub
```
EA SCRIPTS OVERVIEW

Use SQL queries

Repository object available methods:

- **Repository.GetElementSet**(query, 2) : **returns** a *collection of EA elements* according to the SQL query

- **Repository.SQLQuery**(query) : **returns** an *XML formatted string value* (SELECT queries only)

- **Repository.Execute**(query) : **runs** a query (UPDATE/DELETE/INSERT)

```sql
SQL = "select t_attribute.ea_guid, t_object.Name as ClassName, t_attribute.Name, t_attribute.Type "
SQL = SQL & "from t_attribute inner join t_object on t_attribute.Object_ID = t_object.Object_ID "
SQL = SQL & "where t_object.Object_Type='Class' and t_object.Name='' & className & ''"
queryResult = Repository.SQLQuery(SQL)
Session.Output queryResult
set xmlDom = XMLParseXML(queryResult)
attributesList = XMLGetNodeTextArray(xmlDom, "//ea_guid")
for each attItem in attributesList
    Session.Output "Updating attribute with guid " & attItem
    SQL = "update t_attribute set Types='String' where ea_guid='' & attItem & ''"
    if (DEBUGMode) then
        Session.Output SQL
    end if
queryResult = Repository.Execute(SQL)
```
EA SCRIPTS OVERVIEW

Use SQL queries

Be careful when running UPDATE/DELETE/INSERT queries, it can delete or break your model!

Queries must be compatible with the used DBMS (MySQL, SQL Server, Postgres, Oracle, Jet/EAP)

- e.g. apply the right **asterisk wild card** $*$ vs $\%$

- Use `Repository.RepositoryType` method to get the DB name e.g. POSTGRES

Use EA SQL Scratch pad

- To build queries (IntelliSense available: Ctrl+Space)
- To test and fix queries from the DEBUG System Output
EA SCRIPTS OVERVIEW

Object Model API: see EA User Guide


EA DB schema: generate a reverse (98 tables)
CONTENT

01. INTRODUCTION

02. EA SCRIPTS OVERVIEW

03. SCRIPTS HELPERS

04. EA-MATIC TRIGGERED SCRIPTS

05. DEBUG & HINTS

06. DEMONSTRATION
For the developer

Scripting variables used

- `bShowOutput = false`
- `'bShowOutput = true`
- `sOutputPath = ""
  `'sOutputPath = "C:\Users\GFI\Desktop"
- `dim outHeader`
- `outHeader = " | XLS_1-Excel Export Class Details from Branch"

`Debug Mode`

Prompt for a Folder?

Script name for the System Output

```
if (bShowOutput) then
  Session.Output SQL
end if
```

```
Session.Output Now & outHeader & " - Starting"
```
Publication process:

- Export reference data
- Import reference data
- Delete temporary script groups
- Disable the scripting Debug Mode
- Export reference data

Updates all scripts
```
'bShowOutput = true
'ssOutputPath = "C:\Users\GFI\Desktop"
```

Published scripts available to install

All scripts are exported (no selection available)

Deletes reserved scripts
SCRIPTS HELPERS

For the users

- Store the last selected folder for the input or output file(s)
- Save time with the option to reuse the previous file location
- Apply scripts filters
  - Workaround for the lack of sub menus with 15+ scripts
- Built-in help

DEMO
EA-MATIC ADD-IN

Automatically triggered scripts for users

ea-Matic is a free add-in by Geert Bellekens.  
bellekens.com/ea-matic

EA broadcast events are sent to all add-ins.

e.g. EAOOnPostNewElement is received when a new element is created

ea-Matic can trigger a script with an associated event.

Replaces users repetitive tasks.

Add-in installed only for the target users.

"Magic keyword"

'EA-Matic
'a new connector has been created.
function EA_OnPostNewConnector(Info)
  'get the connector id
  dim connectorID, newConnector
  connectorID = Info.Get("ConnectorID")
  set newConnector = Repository.GetConnectorByID(connectorID)
  MsgBox "New connector type: " & newConnector.Type
end function

Associated event
Script debugging aids in the development and maintenance of scripts.

Set Breakpoints to step over/into/out.

View local variables values at the time of execution.
HINTS

Code editor key shortcuts

- Ctrl+F2 to set bookmarks, F2/Shift+F2 go to next or previous

- Ctrl+Shift+C = Toggle line comment on selection

- Ctrl+Space = Intellisense

- Shift+Alt+Return = Fullscreen

- Ctrl+U = to lowercase, Ctrl+Shift+U = to uppercase

- Ctrl+keypad(-) Zoom out, Ctrl+keypad(+) Zoom in, Ctrl+keypad(/) restore
Search in Scripts

Use Sparx EA built-in libraries
| 01. | INTRODUCTION |
| 02. | EA SCRIPTS OVERVIEW |
| 03. | SCRIPTS HELPERS |
| 04. | EA-MATIC TRIGGERED SCRIPTS |
| 05. | DEBUG & HINTS |
| 06. | DEMONSTRATION |
**Main topics**

**Outside applications & people**
- Manage EA user permissions
- Attributes mapping
- UML to JDL Generator (JHipster Entities)

**Third party tools**
- Stakeholders, client...

**Internal use**
- Lock/unlock Branch
- Move classes/tables with criteria (prefix, attr count)
- Compare model with template
- Migrate Diagrams
- Version

**Method**
- EA users

**Admin**

**Standards**

**User Helper**

**QA**
Main topics

**DEMONSTRATION**

**ENTERPRISE ARCHITECT USER GROUP**

**EA User Group London 2018**

Extending EA with custom scripts


**SCRIPTS DEMONSTRATION**

- **Lock/unlock branch**
  - User read only mode (not available via the UI)
  - Example:
    - Schema.org Data Model

- **Migrate diagrams between versions**
  - Reverse on new releases with the need to maintain custom diagrams before deleting old versions
  - Example:
    - Migrate Diagrams

- **Attributes mapping**
  - Excel import/export on attributes to map Data Model (e.g., Application compliance with a standard)
  - Example:
    - UML/Schema.org mapping

- **Isolate classes based on a prefix/suffix or number of attributes in a package**
  - Useful for large DB Reverse (>100 tables + >50 columns on tables)
  - Example:
    - DBReverse1

- **Compare a model with its original template**
  - Mark populated / unused models with *
  - Example:
    - Template Compare

- **EA Security management from Excel**
  - Context: with large number of EA projects to maintain, this Excel file lets the Admin update the group permissions and assigned users in a consistent manner.
  - Example:
    - EA Security & user management

- **Sparx UML to JHipster JDL integration**
  - Example:
    - JHipster 3 EA integration

**DEMO**
CONTACT

Guillaume FINANCE

Blog:  www.umlchannel.com
Add-in:  www.eautils.com
Twitter:  @umlchannel
LinkedIn:  fr.linkedin.com/in/guillaumefinance