**Subject:** Re: Fwd: Ongoing projects - status update as of 2023/09/25 / requests for clarification
**From:** Michal Siemaszko <mhs@into.software>
**Date:** 9/25/23, 23:27
**To:** Jürgen Albert <j.albert@data-in-motion.biz>

Hi Juergen,

Requirements for flat mode CSV exporter we discussed on at least 4 or 5 different occasions, including https://github.com/geckoprojects-org/org.gecko.emf.utils/issues/14, https://github.com/geckoprojects-org/org.gecko.emf.utils/pull/21, calls, and email exchanges. ODS exporter is ready for over 6 months already, merged to 'jakarta' branch even – and CSV exporter in large part is based on that same code.

Flat mode CSV exporter is implemented exactly as per requirements I received from you during those discussions. The only thing that differs is the URI instead of IDs – that was one of the questions I asked in summary I sent last night.

Class hierarchy is already checked for flat mode exporter – see commits from PR https://github.com/geckoprojects-org/org.gecko.emf.utils/pull/23, i.e. most recent from 23.09: https://github.com/geckoprojects-org/org.gecko.emf.utils/pull/23/commits/4a3b153e145d1d118fd3086d0a29bcff9ac930ad

What you are proposing here, as far I as I can understand, is:

- artificially restricting mode of operation, even though it is already written in a very flexible way, including infinite recursion / resolving references and references' attributes, etc. why should I artificially restrict what it can do if it already does all these things ?
- shifting preparation / handling of different scenarios onto user instead of the tool itself – i.e. your example re: addresses, etc.
- no way to correlate data if only URIs are used, instead of IDs, especially in flat mode

In addition, I explained some time ago that not all EObjects have IDs, and that is why "pseudo IDs" are generated – these are necessary for using any kind of correlation / referencing / linking.

I also did not receive questions to all answers I asked in summary I sent you; this is especially needed since, as you say, you will not be available second half of this week; please see summary I sent last night.

In any case, tool is already written, in a very flexible way, exactly as per requirements I received, with ample time for review on several occasions. I will of course supplement it if anything is missing – i.e. URIs instead of IDs – but I need you to confirm / clarify regarding above mentioned, plus need answers to questions I asked, so I can plan ahead.

For comparison, please see attached flat-mode CSV, which was exported having disabled "export non-containment references" export option, and using URIs instead of IDs. IMHO, it provides very little information / therefore value, as compared to flat-mode CSVs I shared two days ago ( see my comment https://github.com/geckoprojects-org/org.gecko.emf.utils/pull/23#issuecomment-1732401123, especially https://github.com/geckoprojects-org/org.gecko.emf.utils/files/12707227/testExportExampleModelBasicEObjectsToCsvFlatMode1369267426601493714 7.csv

BTW: the "Save resource" test cases are implemented for quite some already in:

- `org.gecko.emf.csv.tests.EMFCSVResourceTest`
- `org.gecko.emf.xlsx.tests.EMFXLSXResourceTest`
- `org.gecko.emf.ods.tests.EMFODSResourceTest`

Regards,

--
Michael H. Siemaszko
Telegram: mhsiemaszko
Email: mhs@into.software,
mhsiemaszko@7thraylabs.com
WWW: http://ideas.into.software/
GitHub: https://github.com/ideas-into-software/
LinkedIn: http://www.linkedin.com/in/mhsiemaszko/
Twitter: https://twitter.com/IntoSoftware/

On 9/25/23 21:08, Jürgen Albert wrote:

> Hi Michal,
>
> I'm sorry but I just have realized that I haven't looked deep enough into your test cases, as I would have realized a few things earlier on.
>
> EMF is intended to be used in a certain way and I like to explain that before I get into certain requirements that that I hope will make the Exporters a bit less complex. EMF in and of itself provides a bit better POJOs with their Model definition attached. What makes it powerful from our perspective is the attached concept of Resources, which come in handy for serialization. Thus any kind of read and write is always done through a Resource. Thus any kind of export will always happen as follows:
>
> ```
> Resource resource = set.createResource(URI.createFileURI("export.csv"));
>
> resource.getContents().add(simpsonFamily);
> resource.getContents().add(flintstonesFamily);
>
> resource.save(Collections.singletonMap("someOption", "optionValue"));
> ```
>
> This also means, that all EObjects that are touched during the Export are:
> a. Contained in the Resource to Export.
> b. Are Contained inside an EObject that is part of that Resource (EReferences with Containment true).
> c. EObjects References via EReferences with Containment false are:
> 1. Proxys, with a valid ProxyURI
> 2. EObjects that are in turn contained in a Resource
>
> EMF already has a quite powerful mechanism to address references, which comes for free and out if the box.
>
> I have modified one of your tests to showcase what I mean:
> (You have to add the org.gecko.emf.json bundle as well)
>
> ```
> @Test
> public void testExportExampleModelBasicInvalidClassHierarchyException(
> @InjectService(cardinality = 1, timeout = 4000, filter = "(component.name=EMFCSVExporter)") ServiceAware<EMFExporter> emfCsvExporterAware,
> @InjectService BasicFactory basicFactory, @InjectService BasicPackage basicPackage, @InjectService ResourceSet set) throws Exception {
>
> assertThat(emfCsvExporterAware.getServices()).hasSize(1);
> EMFExporter emfCsvExporterService = emfCsvExporterAware.getService();
> assertThat(emfCsvExporterService).isNotNull();
>
> Family simpsonFamily = createSimpsonFamily(basicFactory);
>
> Family flintstonesFamily = createFlintstonesFamily(basicFactory);
> ```

```
                BusinessPerson businessPerson = createBusinessPerson(basicFactory);

                Path filePath = Files.createTempFile("testExportExampleModelBasicInvalidClassHierarchyException", ".csv");

                OutputStream fileOutputStream = Files.newOutputStream(filePath);

                Resource resource = set.createResource(URI.createFileURI("export.json"));

                resource.getContents().add(simpsonFamily);
                resource.getContents().add(flintstonesFamily);
                resource.getContents().add(businessPerson);

                resource.getContents().addAll(simpsonFamily.eCrossReferences());
                resource.getContents().addAll(flintstonesFamily.eCrossReferences());
                resource.getContents().addAll(businessPerson.eCrossReferences());

                Resource addresses = set.createResource(URI.createFileURI("address.json"));

                simpsonFamily.eCrossReferences().stream()
                .filter(Person.class::isInstance)
                .map(Person.class::cast)
                .forEach(p -> {
                        addresses.getContents().add(p.getAddress());
                });

                resource.save(System.err, null);
}
```

It adds the Objects you export to one Resource and all the Addresses of the Simpons to the address json. The Flintstones Address is not attached anywhere (They are called dangling references). This will print out the attached json.

EcoreUtil.getURI is used here to produce all the values for the $ref fields. What do they mean:
"$ref" : "2134629d-cc6c-42ad-970f-b91365b20b67" -> An object in the Same document with the stated ID
"$ref" : "address.json#0c5c8a13-8f5b-4144-a1df-2cb44e210cf5" -> the References Object can be found in the document address.json and the Fragment (the element behind the #) is the ID of the Object in that document.
"$ref" : "#da7ed7bc-975b-4273-b26f-976f8bed40dc" -> The Referenced Object has an ID, but is not contained anywhere

If an EObjects EClass has no Attribute marked as ID  attribute, EMF will be able to provide a reference as well. The result could looks like: "something.json//@orders.0/@items.4".

With this in mind I'd like to define some assumption under which you can work:
* Garbage in, Garbage out. If an Object has a defined ID attribute, but has no Attribute then you can take what EcoreUtil.getURI provides. If the ID is technically necessary, you can fail with an Exception.
* Not sure if you will ever get null as an URI, but you can always rely on the isCurrentDocumentReference method on the URI to find if you have a valid reference inside your resource.
* We will only export Objects that are part of the Resource

Flat Mode:
* The first Object in the Resources Content List defines the Kind of EClass exported
  - EObjects in the Resources Content List must be "related" to the first EClass (All need must share a common Class hierarchy)
  - EObjects in the List that don't match the above -> Exception
* References to non containment Objects that are not part of the Resources will be written as follows: family.children.0.ref, family.children.1.ref and will have their respective URIs
* We only export Containments. This Means that exporting the Simpons Family will only result in Id,father.ref,mother.ref,children.0.ref, children.1.ref, children.2.ref
* The rest of the Headers can stay as they are.

NonFlat Mode:
* All Objecs in the Resources will be exported.
* Non Containment References where EcoreUtil.getUri().isCurrentDocumentReference() == false -> write the URI as ref.
* I believe the current Version is solid, If I haven't missed something.

I hope this makes it a bit more clear.If you still have questions and/or I have overlooked any cases please don't hesitate to ask.

PLEASE NOTE: I will only be available this week till 1430 on Wednesday. After that I'm only sparingly reachable for the rest of the Week.

Jürgen.

Am 25/09/2023 um 16:15 schrieb Michal Siemaszko:

> Hi Juergen,
>
> As a follow up to our call today, please see examples I shared via https://github.com/geckoprojects-org/org.gecko.emf.utils/pull/23#issuecomment-1732401123 (CSV flat and zip mode) and let me know regarding questions from document attached. Please clarify also in which cases / modes should URIs be output instead of IDs - keeping in mind that in flat mode, if references / attributes are not unpacked, this would become unreadable (as you have only one file / flat CSV representation format).
>
> For comparison, you can see XLSX documents shared via https://github.com/geckoprojects-org/org.gecko.emf.utils/pull/23#issuecomment-1713014605 - where for complex models, you have one document (XLSX document) with different sheets.
>
> Regards,
>
> -------- Forwarded Message --------
> **Subject:**Re: Ongoing projects - status update as of 2023/09/25 / requests for clarification
> **Date:**Mon, 25 Sep 2023 00:02:38 +0200
> **From:**Michal Siemaszko <mhs@into.software>
> **To:**Jürgen Albert <j.albert@data-in-motion.biz>, Mark Hoffmann <m.hoffmann@data-in-motion.biz>
>
> Hi,
>
> Attached please find PDF with status update as of 25.09.2023.
>
> Regards,
>
> --

> Michael H. Siemaszko
> Telegram: mhsiemaszko
> Email: mhs@into.software,
> mhsiemaszko@7thraylabs.com
> WWW: http://ideas.into.software/
> GitHub: https://github.com/ideas-into-software/
> LinkedIn: http://www.linkedin.com/in/mhsiemaszko/
> Twitter: https://twitter.com/IntoSoftware/

--
Jürgen Albert
CEO
Chair Eclipse OSGi Working Group Steering Committee

Data In Motion Consulting GmbH

Kahlaische Str. 4
07745 Jena

Mobil: +49 157-72521634
E-Mail: j.albert@datainmotion.de
Web: www.datainmotion.de

XING: https://www.xing.com/profile/Juergen_Albert5
LinkedIn: https://www.linkedin.com/in/juergen-albert-6a1796/

Rechtliches

Jena HBR 513025

Attachments:

testExportExampleModelBasicResourceToCsvFlatMode1611272311863004363 8.csv 422 bytes