Subject: Re: Fwd: Ongoing projects - status update as of 2023/09/25 / requests for clarification
From: Michal Siemaszko <mhs@into.software>
Date: 9/26/23, 18:37
To: Jürgen Albert <j.albert@data-in-motion.biz>

Juergen,

Likewise, see comments below -

On 9/26/23 17:37, Jürgen Albert wrote:

> Hi Michal,
>
> you can find my answers inline.
>
> Jürgen.
>
> Am 25/09/2023 um 23:27 schrieb Michal Siemaszko:
>
>> 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.
>
> As I stated in my last mail: I did not realize some things till now. This means, that I had a different understanding about some things. When we reach a consensus, we should change what needs changing, even if something is already merged.

This does not make things simpler, and affects entire design, pretty much from ground up. It was created based on requirements I received from you since beginning of this year, as well as discussions we had during calls / reviews. Large part of what is done currently done based on that code (hence extracted to 'Abstract' base class, i.e. first thing I did when you asked to implement additional exporters).

Solution implemented is very flexible, handles infinite levels of recursion, presents all information, including metadata and mapping tables, and is able to output same into multiple formats.

The purpose of having IDs where no IDs exist in model I explained multiple times - you agreed that in such case, such field should be marked in meta-data sheet / table as "transient" / "temporary" so user will be able to recognize such.

> > 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.
>
> Which we are discussing right now.
>
> > 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
>
> I thought as much. I just defined it again, to be consistent.

This is in no way consistent with any of the requirements / discussions so far - that is the problem. Please see above comment for more details. This is a fundamental change which affects entire design - that is definitely not consistency.

> > 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
>
> Some things I have learned after working with EMF for over 10 years: All this topics are

> already solved in EMF. It implements Standards, that are proven and work as they do for good reasons, including the limitations that come with it.
>
> The Model defines the truth and the only truth. Thus a Model must be created with deliberation. One of the choice is the question of containment/non-containment. This Addresses the point of recursion. References and their resolving are made possible via their URIs, which in EMF always will identify an Object.

Solution was tested across three different models, with both containment and non-containment references, and tenths of thousands of records. It works in all these cases.

What I encountered across these 3 models already is that models have errors which need to be corrected / accounted for, and I doubt such errors in model do not happen often / with other models.

> As I said, this also comes with a few limitations. One of them is that, a non-containment will never work, if the EObjects don't have a Resource with a proper URI. If you take one of our Family Objects and Write it out via a JsonResource or an XML Resource will always produce invalid references, when the Persons it references have no Resource. In Json it would work if you add the Person to the same resource. In XML it would not, as an XML document by definition can only have one Root Element and not many. So I conclude that certain sterilization formats will have certain limits, and that is okay.

I am not sure what you mean that it will never work, if it works already. That is the problem I have - I created a solution which works in all these cases, we discussed this multiple times since beginning of this year, reviewed it / supplemented requirements / and now you claim something that works does not work.

This works already, with infinite levels of recursion, both for non-flat and flat modes. It provides output where each of the records can be uniquely identified, and when metadata is exported, such "transient"/"pseudo ID" fields can be easily identified - they are needed for some of the other features which you asked me to implement and I did.

This is a data exporter which outputs information in multiple formats, but not in EMF format. EMF is input to this exporter, what is output is data and metadata contained in such model, but in different formats - why do you insist that output formats are in any way bound by how EMF presents information. These are different formats.

> Thus my requirement for the “export non-containment references” option in the sense that it exports the full Object, was actually ill conceived and we should modify it, that it controls if references will be exported or not.
>
> You may ask why, as you already have implemented it and it works. It works until it doesn't :-). The Model you test with is rather small and the amount of Objects is limited. Exporting all the non containment references will make the amount of exported uncontrollable in a large enough dataset, when your Model has too many connections. I've been there already in other Projects :-).

Do you mean for non-flat or flat mode?

For non-flat mode: would you please provide an example of model and data set which you suspect could not be handled by existing solution ? Like with trees model and data set - which it does handle with no problems.

For flat mode: there is already a built-in restriction regarding class hierarchy, and first object in list passed determines which class hierarchy will be used - 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

Likewise, would you please provide an example of model and data set which you suspect could not be handled by existing flat-mode solution?

> You mentioned shifting of preparation to the user: Yes this is something that is totally acceptable for me. If you have an Object that should have an ID but it does not, than it is an invalid Object, if the ID is mandatory. In such a case, fail early is the better approach. If you try to correct it by assuming an ID, you kick the ball down the road until somebody else later discovers, that the ID is actually no valid ID in the System the data came from. The Exporter is also no tool for the direct enduser. The system around it must take care that the date it receives is valid, similar to any other System.
>
> > 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.

> A URI will always identify an EObject if it has a Resource (the Resource is the important part). This works for Objects that have no ID Attribute and I believe can also handle EObjects where the ID is empty. It also comes with a few other issues: You create IDs that actually don't exist. If I would ask a Rest endpoint for the same Data would result in Objects with no or different IDs (if somebody corrected the data and created IDs). For EClass that have no ID field, It furthermore would actually change the Model in the eye of the recipient of the export, by presenting and ID field that does not exist.
>
> BTW: a URI for an Object with no ID would looks as follows: "a443fe59-1f9b-46cd-8a5a-c895b45547b2@contact.2" which would mean: Object with id: a443fe59-1f9b-46cd-8a5a-c895b45547b2 and there the 3rd contact in the list. This is by the way also a standard.

We discussed this several times already and you agreed that in such cases such IDs should be marked in meta-data sheet / table as such. They are required by other features you asked me to implement.

> > 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.
>
> I hope I don't miss any of the Querstions:
> * flat mode for R/ODS/EXCEL -> no (may change if Stefan wants it later on)
> * metadata: Not for flat mode
> * limit: yes, we should follow that limit
> * log4j: you can ignore this. This something that goes away with the proper logger setup.

- **Flat mode - only yesterday you confirmed this logic should be extracted to base class (which is why I asked), and now you say something completely different; which one should it be then?**

- **Regarding metadata - I asked about type-level metadata documentation. Please see my summary for more details about distinction / what I'm referring to**

- **Regarding 1 million rows limit - I asked if more than one 1 million rows**

should be supported; so, should it be supported, or should it throw exception if that limit is reached, like I already implemented such for number of columns ? 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

- **Regarding log4j - that is exactly why I asked about this as I did not want to waste time if that problem is known to you; it is apparently known to you, so would you please share solution to this problem - purpose of my question.**

> > 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/testExportExampleModelBasicEObjectsToCsvFlatMode1369267426601493 7147.csv
>
> I can totally live with such a result.
>
> Thus to reiterate and to conclude what I would suggest:
>
> Flat mode:
> * no non containment Objects will be exported
> * the “export non-containment references” option, will control if refs will be written or ignored. The Family example would boil down to the resulting header: Family_id
>
> "normal" mode
> * only Objects that are part of the given List (Resources content list) will be exported
> * non containments that are not part of the list will be referenced via their URI

> My goal with this is not to invalidate the work you already have done. I want to simplify the solution a bit, as it will become less error prone. It additionally will become more "EMF" like in its behavior. What do you say?

What you are proposing almost entirely invalidates all the work done on so far. Solution I provided is very flexible, with all errors caught so far handled already. Please see my earlier comments in this response for more details. This is definitely not "simplifying". In addition, "less error prone" implies that you found errors with existing solution or expect errors, but none of information you presented gives any specific example where what is already done would not work in such and such case; therefore, I asked to please provide examples of models and data sets which you think would not be handled.

Perhaps you would like to have a call where I can present and provide you with an overview of which part is used for what ? We already did similar some time ago, and I explained briefly during our calls several times.

I feel very uncomfortable with artificially restricting what is very flexible already, including flat mode export being able to handle infinite recursion for non-containment references, and throwing out over 50% of code just because you are afraid of something non-specific.

So, since focus of current iteration was flat mode, and that works already for both containment and non-containment, and I would like to wrap up with remaining work left for this month (I was going to switch to "R" exporter and have it ready by end of this month, before this came up .. so unfortunately this will not be done this month because of this), would you be satisfied with:

a) having URIs for non-containment references only, i.e. if “OPTION_EXPORT_NONCONTAINMENT” export option is set to false, use URIs only for non-containment references, otherwise stick with IDs;

b) having flat mode work with both containment AND non-containment, as it works already;

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/
```

> > 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

> --
> 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