Conversion Tools - Feature #6256

improved profile support

04/06/2022 06:46 PM - Greg Shah

Status:	WIP	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Constantin Asofiei	**% Done:**	60%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**version:**	
vendor_id:	GCD		

Description

Related issues:

Related to Conversion Tools - Feature #6320: sharded schemata	**New**
Related to Conversion Tools - Feature #7169: drive conversion order using use...	**New**
Related to Conversion Tools - Feature #8525: eliminate the runtime dependency...	**New**

History

#1 - 04/06/2022 07:03 PM - Greg Shah

- Assignee set to Constantin Asofiei

For complex projects, the OpenEdge compilation will often be done in multiple different compile runs. Each one might have different databases connected, different propaths (and other compile settings) and of course, different file-sets.

Our current approach for this idea is very dependent upon overriding values in directory.hints, which is really not a good solution. It splits the configuration for the project up into little shards and spreads them around the file system. This makes it hard to reason about the project configuration. It is also more work to define and maintain.

This task will extend Ovidiu's profile approach (see #4105) to make it more generic. The idea is that anything that can be specified in the global section of p2j.cfg.xml will be allowed in the profile (including the file-set capability of #6253). Eventually, we will allow some or all of the things from hints files to be pushed up into p2j.cfg.xml (including in profiles), but for now we will just enable the current capabilities of the global section to be specified in any profile.

That means that the profile support is not just for namespaces and when these values are looked up, any specification in the current profile section should take precedence over anything in the global section. If not specified in the current profile section, the same value in the global section will be honored.

This will allow the propath, file-set, databases ... everything needed to parse/convert will be in each named profile.

The ConversionDriver already allows specification of a -P<profile>. It should be enhanced to run with any of these:

- a single profile name (as today)
- a comma-separated list of profile names (e.g. "p-one,p-two")
- the special profile name "all" which means that all profiles will be executed in the order they are defined

These last two cases will require reworking of the ConversionDriver/AstGenerator/ScanDriver to handle the swapping of configuration during a single run. In other words, when more than one profile is used, there are multiple file sets that must be combined into a single run while letting their configuration be different as needed. We cannot do this as multiple ConversionDriver runs because some of the downstream conversion processing needs to be aware of all files in one pass. I think this is the hardest part of the task.

I think we may have some OO management issues to handle here as well (e.g. different sets of class defs by propath?).

#2 - 04/27/2022 10:55 AM - Greg Shah

- Assignee changed from Constantin Asofiei to Eric Faulhaber

#4 - 04/27/2022 04:59 PM - Eric Faulhaber

To translate the above configuration requirements into a sample p2j.cfg.xml, I understand it should look something like this:

```
<?xml version="1.0"?>

<!-- P2J main configuration -->
<cfg>

   <global>

      <!-- default conversion tools "internal" values -->
      <parameter name="P2J_RULES"        value="${P2J_HOME}/p2j/rules" />
      <parameter name="patpath"          value=".:${P2J_HOME}/pattern:${P2J_RULES}/include:${P2J_RULES}:" />
      <parameter name="registry"         value="./cfg/registry.xml" />
      <parameter name="rootlist"         value="./cfg/default_rootlist.xml" />
      <parameter name="matchlist"        value="./cfg/default_matchlist.xml" />
      <parameter name="datanames"        value="./cfg/default_datanames.xml" />

      <!-- default values from the original system on which Progress 4GL ran -->
      <parameter name="propath"          value="${P2J_HOME}:${P2J_HOME}/abl/default/path/1:${P2J_HOME}/abl/def
ault/path/2:${P2J_HOME}/abl/default/path/3:" />
      <parameter name="basepath"         value="./abl" />
      <parameter name="include-spec"     value="*.[fhiv]" />
      <parameter name="oo-skeleton-path" value="./skeleton" />

      <!-- conversion output -->
      <parameter name="output-root"      value="./src" />
      <parameter name="pkgroot"          value="com.acme" />
      <parameter name="merge-point"      value="/server/default/runtime/default/" />
      <parameter name="comments"         value="true" />
      <parameter name="foreign-keys"     value="false" />
      <parameter name="opsys"            value="UNIX" />

      <!-- default file-set -->
      <file-set>
         <directory recursive="true" path="./abl/" spec="(*.[pPwWtT]|*.cls)" />
         <exclude filter="./abl/possenet/*" />
      </file-set>

   </global>

   <schema>

      <namespace name="acme" default="true" />
      <namespace name="standard" default="true" />

      <metadata name="standard">
         <table name="_area" />
         <table name="_db" />
         <table name="_file" />
         <table name="_file-trig" />
         <table name="_field" />
         <table name="_field-trig" />
         <table name="_index" />
         <table name="_index-field" />
         <table name="_user" />
         <table name="_connect" />
         <table name="_myconnection" />
         <table name="_database-feature" />
         <table name="_filelist" />
         <table name="_startup" />
         <table name="_lock" />
         <table name="_usertablestat" />
         <table name="_sequence" />
      </metadata>
   </schema>

   <profile name="module1">

      <file-set>
```

```
         <directory recursive="true" path="./abl/common/" spec="(*.[pPwWtT]|*.cls)" />
         <directory recursive="true" path="./abl/module1/oo/" spec="*.cls" />
         <directory recursive="true" path="./abl/module1/legacy/" spec="*.[pPwW]" />
         <directory recursive="true" path="./abl/module1/trig/" spec="*.[tT]" />
         <exclude filter="./abl/module1/legacy/broken/*" />
      </file-set>

      <parameter name="merge-point"      value="/server/default/runtime/module1/" />
      <parameter name="propath"          value="${P2J_HOME}:${P2J_HOME}/abl/common/:${P2J_HOME}/abl/module1:${
P2J_HOME}/abl/some/other/path:" />

      <schema>
         <namespace name="module1" default="true" />
         <namespace name="standard" />
      </schema>

   </profile>

   <profile name="module2">

      <file-set>
         <directory recursive="true" path="./abl/common/" spec="(*.[pPwWtT]|*.cls)" />
         <directory recursive="true" path="./abl/module2/main/" spec="(*.[pPwWtT]|*.cls)" />
      </file-set>

      <parameter name="datanames"        value="./some/other/location/module2_datanames.xml" />
      <parameter name="propath"          value="${P2J_HOME}:${P2J_HOME}/abl/common/:${P2J_HOME}/abl/module2:${
P2J_HOME}/abl/module2/main:" />

      <schema>
         <namespace name="module2" default="true" />
         <namespace name="standard" />
      </schema>

   </profile>

   <profile name="module2-test">

      <file-set>
         <directory recursive="true" path="./abl/common/" spec="(*.[pPwWtT]|*.cls)" />
         <directory recursive="true" path="./abl/module2/main/" spec="(*.[pPwWtT]|*.cls)" />
         <directory recursive="true" path="./abl/module2/test/" spec="*.[pP]" />
      </file-set>

      <parameter name="propath"          value="${P2J_HOME}:${P2J_HOME}/abl/common/:${P2J_HOME}/abl/module2/ma
in:${P2J_HOME}/abl/module2/test:" />

      <schema>
         <namespace name="module2_unittest" default="true" />
         <namespace name="standard" />
      </schema>

   </profile>

</cfg>
```

Is this the right vision?

It seems like some of the global parameters don't make sense to override (e.g., registry, oo-skeleton-path, etc.).

I'm assuming (for now) that profile *.df files will be found in ./data/<profile-name>/.

#5 - 04/27/2022 06:07 PM - Ovidiu Maxiniuc

At this moment the structure of the configuration file is:

```
<cfg>
  <global>...</global>

  <schema>
    <!-- unnamed profile -->
    <namespace name="ns_0.1"/>
    <namespace name="ns_0.2"/>
    <!-- end of unnamed profile -->

    <profile name="p1">
       <namespace name="ns_1.1"/>
       <namespace name="ns_1.2"/>
    </profile>
    ...
    <profile  name="pk">
       <namespace name="ns_k.1"/>
       <namespace name="ns_k.2"/>
    </profile>

    <metadata name="standard" />
  </schema>
</cfg>
```

Each profile have a set of namespaces, but metadata is common. All of them are part of the schema node (SchemaConfig). There is an optional "unnamed" profile which allows namespaces to be present directly into schema node, which assures backward compatibility but whose usage I think we should discourage. I understand that from this point forward, we will switch the structure, in that the nesting node is a profile, instead, and will contain:

- a set of property-es specific to respective profile,
- the schema sub-node with a set of namespaces and
- the set of files to be processed.

The metadata should be still common and defined only once.

#6 - 04/27/2022 07:27 PM - Greg Shah

> Is this the right vision?

Mostly, yes. The part I don't understand is why there is a schema node inside the profile. I would not change anything. Just allow the namespace elements to be directly in the profile element.

> It seems like some of the global parameters don't make sense to override (e.g., registry, oo-skeleton-path, etc.).

Agreed.

> I'm assuming (for now) that profile *.df files will be found in ./data/<profile-name>/.

No, I don't think this is the case. Ovidiu already added support for specifying a custom path for the .df. This should be used unchanged. In other words, we don't need any changes to how the namespace nodes are processed in a profile.

#7 - 04/28/2022 07:25 AM - Ovidiu Maxiniuc

Greg Shah wrote:

> > I'm assuming (for now) that profile *.df files will be found in ./data/<profile-name>/.
>
> No, I don't think this is the case. Ovidiu already added support for specifying a custom path for the .df. This should be used unchanged. In other words, we don't need any changes to how the namespace nodes are processed in a profile.

I think that is a good idea for the default location: if importFile is not specified then the default value should be ./data/<profile-name>/<namespace>.df. For the unnamed profile the .df will be located directly into ./data/, by default.

#8 - 04/28/2022 12:00 PM - Eric Faulhaber

- Status changed from New to WIP

Incorporating the above suggestions/requirements, I have updated the sample configuration as follows:

```
<?xml version="1.0"?>

<!-- P2J main configuration -->
<cfg>

   <global>

      <!-- default conversion tools "internal" values -->
      <parameter name="P2J_RULES"        value="${P2J_HOME}/p2j/rules" />
      <parameter name="patpath"          value=".:${P2J_HOME}/pattern:${P2J_RULES}/include:${P2J_RULES}:" />
      <parameter name="registry"         value="./cfg/registry.xml" />
      <parameter name="rootlist"         value="./cfg/default_rootlist.xml" />
      <parameter name="matchlist"        value="./cfg/default_matchlist.xml" />
      <parameter name="datanames"        value="./cfg/default_datanames.xml" />

      <!-- default values from the original system on which Progress 4GL ran -->
      <parameter name="propath"          value="${P2J_HOME}:${P2J_HOME}/abl/default/path/1:${P2J_HOME}/abl/def
ault/path/2:${P2J_HOME}/abl/default/path/3:" />
      <parameter name="basepath"         value="./abl" />
      <parameter name="include-spec"     value="*.[fhiv]" />
      <parameter name="oo-skeleton-path" value="./skeleton" />

      <!-- conversion output -->
      <parameter name="output-root"      value="./src" />
      <parameter name="pkgroot"          value="com.acme" />
      <parameter name="merge-point"      value="/server/default/runtime/default/" />
      <parameter name="comments"         value="true" />
      <parameter name="foreign-keys"     value="false" />
      <parameter name="opsys"            value="UNIX" />

      <!-- default file-set -->
      <file-set>
         <directory recursive="true" path="./abl/" spec="(*.[pPwWtT]|*.cls)" />
         <exclude filter="./abl/possenet/*" />
      </file-set>

   </global>

   <schema>

      <namespace name="acme" default="true" />
      <namespace name="standard" default="true" />

      <metadata name="standard">
         <table name="_area" />
         <table name="_db" />
         <table name="_file" />
         <table name="_file-trig" />
         <table name="_field" />
         <table name="_field-trig" />
         <table name="_index" />
         <table name="_index-field" />
         <table name="_user" />
         <table name="_connect" />
         <table name="_myconnection" />
         <table name="_database-feature" />
         <table name="_filelist" />
         <table name="_startup" />
         <table name="_lock" />
         <table name="_usertablestat" />
         <table name="_sequence" />
      </metadata>

   </schema>

   <profile name="module1">

      <file-set>
```

```
         <directory recursive="true" path="./abl/common/" spec="(*.[pPwWtT]|*.cls)" />
         <directory recursive="true" path="./abl/module1/oo/" spec="*.cls" />
         <directory recursive="true" path="./abl/module1/legacy/" spec="*.[pPwW]" />
         <directory recursive="true" path="./abl/module1/trig/" spec="*.[tT]" />
         <exclude filter="./abl/module1/legacy/broken/*" />
      </file-set>

      <parameter name="merge-point"      value="/server/default/runtime/module1/" />
      <parameter name="propath"          value="${P2J_HOME}:${P2J_HOME}/abl/common/:${P2J_HOME}/abl/module1:${
P2J_HOME}/abl/some/other/path:" />

      <namespace name="module1"
                 default="true"
                 importFile="./custom/path/to/module1/schema" />

   </profile>

   <profile name="module2">

      <file-set>
         <directory recursive="true" path="./abl/common/" spec="(*.[pPwWtT]|*.cls)" />
         <directory recursive="true" path="./abl/module2/main/" spec="(*.[pPwWtT]|*.cls)" />
      </file-set>

      <parameter name="datanames"        value="./some/other/location/module2_datanames.xml" />
      <parameter name="propath"          value="${P2J_HOME}:${P2J_HOME}/abl/common/:${P2J_HOME}/abl/module2:${
P2J_HOME}/abl/module2/main:" />

      <namespace name="module2" default="true" />

   </profile>

   <profile name="module2-test">

      <file-set>
         <directory recursive="true" path="./abl/common/" spec="(*.[pPwWtT]|*.cls)" />
         <directory recursive="true" path="./abl/module2/main/" spec="(*.[pPwWtT]|*.cls)" />
         <directory recursive="true" path="./abl/module2/test/" spec="*.[pP]" />
      </file-set>

      <parameter name="propath"          value="${P2J_HOME}:${P2J_HOME}/abl/common/:${P2J_HOME}/abl/module2/ma
in:${P2J_HOME}/abl/module2/test:" />

      <namespace name="module2" default="true" />
      <namespace name="module2_unittest" default="true">
         <parameter name="cpstream" value="1252" />
         <parameter name="byte-mapping" value="81:3F  90:3f  9D:3f" />
         <parameter name="ddl-dialects" value="h2" />
         <dialect-specific name="h2">
            <parameter name="collation" value="en_US@cp1252_fwd_basic" />
         </dialect-specific>
      </namespace>

   </profile>

</cfg>
```

Does that work for everyone?

Ovidiu, do I understand correctly that when the importFile attribute of namespace is specified, schema conversion artifacts (e.g., the *.dict, *.schema, and *.p2o files) now will all be stored alongside the *.df file found at that same importFile path?

#9 - 04/28/2022 12:10 PM - Greg Shah

Yes, the syntax is OK.

#10 - 04/30/2022 09:35 AM - Constantin Asofiei

Greg/Eric: my thought would be the profiles to work like this:

- there is the 'global' profile which is used implicitly
- the 'global' profile also sets defaults for any explicit profile
- each profile has the same syntax as the 'global' profile (so the namespace nodes would be added in a schema node)
- when reading a profile, first the 'global' configuration is created and after that the profile is just merged in the 'global' configuration

This way, we can work with the same Configuration instance, and other FWD conversion code will require no changes.

If we will want to run multiple profiles at the same time, this can work in two modes:

- 'merge' mode, where the profiles are merged and everything is converted together.
- individual mode, where FWD runs conversion for each specified profile, sequentially (same way as you would run individual ConversionDriver commands in a script, to run each profile).

#11 - 04/30/2022 09:39 AM - Constantin Asofiei

When running in single profile mode, the profile will need to overwrite default parameter and file-set nodes, and merge namespace nodes to the global schema, correct?

#12 - 04/30/2022 09:48 AM - Greg Shah

> each profile has the same syntax as the 'global' profile (so the namespace nodes would be added in a schema node)

This is a change from the current approach. I guess the benefit of consistency may outweigh the short term interest in less change. Let's go ahead.

I'm good with the other ideas. I think the multi-profile individual mode will not be used too often. I only see 2 scenarios:

- Can't run the entire project at once because of some kind of memory constraint. This would be a kind of bug that needs to be fixed anyway, so I don't see this as a long term need.
- Early runs of the conversion where we want to allow a wide range of the project to be checked without aborting on each problem. I think we are going to make conversion more tolerant of failures, but still this could have some benefit.

Is there some other use case I'm missing?

#13 - 04/30/2022 09:49 AM - Greg Shah

> When running in single profile mode, the profile will need to overwrite default parameter and file-set nodes,

Correct.

> and merge namespace nodes to the global schema, correct?

We already handle the namespace parts today for this mode. Do we need to change it?

#14 - 04/30/2022 09:56 AM - Constantin Asofiei

- Assignee changed from Eric Faulhaber to Constantin Asofiei

Greg Shah wrote:

> Is there some other use case I'm missing?

I don't think so.

> We already handle the namespace parts today for this mode. Do we need to change it?

At least the syntax in note 5 is no longer valid, right?

Otherwise, I haven't looked at the code yet, I'll see how this can be used.

#15 - 04/30/2022 10:03 AM - Greg Shah

> At least the syntax in note 5 is no longer valid, right?

Right.

#16 - 04/30/2022 11:12 AM - Constantin Asofiei

The profile node has only name attribute in #6256-8.

The SchemaConfig$Profile has a isDefault, which suggests that there can be a isDefault attribute at the profile node. Do we need this? I can't find any reference in redmine related to it, but there is SchemaConfig.getDefaultProfile code which uses this to determine the default profile.

#17 - 04/30/2022 11:24 AM - Greg Shah

The namespace should assume default="true" so that it doesn't have to be duplicated. But it should be possible to define default="false" to establish a logial database but only load it for specific files (via directory or file-level hints). We support this today. Initially, Ovidiu's changes required default="true" to be explicit but he changed that to be assumed since it is the common case.

#18 - 04/30/2022 11:26 AM - Constantin Asofiei

Greg Shah wrote:

> The namespace should assume default="true" so that it doesn't have to be duplicated. But it should be possible to define default="false" to establish a logial database but only load it for specific files (via directory or file-level hints). We support this today. Initially, Ovidiu's changes required default="true" to be explicit but he changed that to be assumed since it is the common case.

I'm not talking about the namespace. I'm talking about the schema/profile node. Current code assumes there can be a <profile default = "true"/> attribute.

#19 - 04/30/2022 11:26 AM - Greg Shah

> The SchemaConfig$Profile has a isDefault, which suggests that there can be a isDefault attribute at the profile node. Do we need this? I can't find any reference in redmine related to it, but there is SchemaConfig.getDefaultProfile code which uses this to determine the default profile.

Sorry, I was thinking about the namespace. I don't recall the reason for this.

#20 - 04/30/2022 11:27 AM - Greg Shah

I don't think there is a hard requirement for that at the profile level.

#21 - 04/30/2022 12:10 PM - Constantin Asofiei

Do you see a need to make the profile/schema node optional? In this case, it would be inherited from the global config.

PS: I'm not working on the multi-profile mode (either merge or list), this was just an idea.

#22 - 04/30/2022 12:12 PM - Greg Shah

> Do you see a need to make the profile/schema node optional? In this case, it would be inherited from the global config.

Yes, this makes sense.

> PS: I'm not working on the multi-profile mode (either merge or list), this was just an idea.

The multi-profile merge mode is needed as part of this task. The multi-profile individual mode is a useful addition, but is not required.

#23 - 04/30/2022 12:13 PM - Constantin Asofiei

Greg Shah wrote:

> The multi-profile merge mode is needed as part of this task.

Understood.

#24 - 04/30/2022 12:20 PM - Constantin Asofiei

Greg Shah wrote:

> > The SchemaConfig$Profile has a isDefault, which suggests that there can be a isDefault attribute at the profile node. Do we need this? I can't find any reference in redmine related to it, but there is SchemaConfig.getDefaultProfile code which uses this to determine the default profile.

> Sorry, I was thinking about the namespace. I don't recall the reason for this.

This is used by an existing customer project, so I'll keep a 'default' at the profile. It will work like this:

- if a profile is set as default, then automatically load the default profile (be it runtime or conversion).
- if multiple profiles are set as default, abend
- if no profile is set as default, let the global configuration be the default, unless one is specified at command line. Currently it uses the first 'schema/profile', a 'random' choice, if no default profile is set. I don't like this, it makes more sense to default to the global config. Let the user explicitly choose what needs to run.

#25 - 04/30/2022 12:22 PM - Constantin Asofiei

Constantin Asofiei wrote:

> - if no profile is set as default, let the global configuration be the default, unless one is specified at command line. Currently it uses the first 'schema/profile', a 'random' choice, if no default profile is set. I don't like this, it makes more sense to default to the global config. Let the user explicitly choose what needs to run.

Another addition: if a profile is set as default, command line can still override it.

#26 - 04/30/2022 01:00 PM - Greg Shah

It is a good plan.

#27 - 04/30/2022 01:03 PM - Constantin Asofiei

Greg Shah wrote:

> The multi-profile merge mode is needed as part of this task.

This will be only for conversion. Runtime part is tricky, especially the schema part.

#28 - 04/30/2022 01:59 PM - Eric Faulhaber

A useful addition (medium term, not immediately) would be to be able to load multiple DF files into a single schema namespace. Rather than concatenating the DF files together manually or maintaining a script to do so, the SchemaLoader could be configured to parse multiple DF files into a single, logical schema. This could be represented as one or more import-file elements as children of the namespace element, rather than the current importFile attribute of namespace. We could leave the importFile attribute intact for backward compatibility or deprecate it, but the preferred way to define a schema namespace would be with the new import-file element. If no importFile attribute or import-file child element is defined, the namespace would default to the current approach of just assuming the name attribute of the namespace element indicates the root name of the DF file, located in the data subdirectory of the project.

For example:

```
<namespace name="federated">
   <import-file path="[optional/relative/location/of/]shard1.df" />
   <import-file path="[optional/different/relative/location/of/]shard2.df" />
</namespace>
```

#29 - 04/30/2022 02:04 PM - Greg Shah

Eric: I think the shard idea makes good sense. We have multiple customers using this approach in OE today. The idea should probably be in its own task.

#30 - 04/30/2022 02:07 PM - Constantin Asofiei

So, the approach is this:

- multiple profiles are allowed (only for conversion time), and they can be specified via -pprofile1 -pprofile2 at command line or as default=true at the p2j.cfg.xml profile
- if no command-line -p argument, then p2j.cfg.xml is checked for one or more default profiles. If none set, the global config is used.
- if one or more profiles are determined as default, start with the pristine global config, and merge all the profiles into this:
  - parameters are overwritten, in the order of how the profiles are defined in p2j.cfg.xml
  - the file-set used is the one computed from the profiles - each profile computes its files individually, and an explicit list of files is returned. The global file-set is not used!
- schema namespaces are merged from all profiles, and are used instead of the global schema. WARNING: if there is a namespace clash and the schema is **not the same** for all namespaces sharing this name (over all used profiles), then ... weird things will happen.
- if a profile has no schema defined, then there will be no schema used for that profile (I do not default to the global schema)

#31 - 04/30/2022 02:15 PM - Greg Shah

When I said the multiple profile merge was needed, I meant a different idea.

Each profile can be run standalone. But when you run multuple profiles as a batch, we need the settings to remain intact. The files specified for the profile should be parsed/converted with the same settings as if the profile was run standalone. But the overall list of files processed will be the aggregate of all the profiles selected.

> parameters are overwritten, in the order of how the profiles are defined in p2j.cfg.xml

No, we don't want this. The settings for a given file will depend on the specific profile it is in + the global settings, that is all.

> schema namespaces are merged from all profiles,

No, this won't work. The idea is the same as for the rest of the parameters. The files specified for that profile must get just the schemata that are specified in that profile.

#32 - 04/30/2022 02:26 PM - Constantin Asofiei

Greg Shah wrote:

> The files specified for the profile should be parsed/converted with the same settings as if the profile was run standalone.

What if a file is part of multiple profiles? I think we see this in a current project. Also, if there is any namespace (or schema file) conflict, it will not work.

What you describe can't be achieved easily... it means switching the conversion context for each file, to set the schema, propath settings, and more that I can't think about without digging into the code.

#33 - 04/30/2022 02:30 PM - Greg Shah

> What if a file is part of multiple profiles?

The profiles are intended to be mutually-exclusive. We can define a rule that the first profile wins (and print a warning), if a file is specified more than once.

> What you describe can't be achieved easily... it means switching the conversion context for each file, to set the schema, propath settings,

Yes, I understand. That is the intention of this task.

#34 - 04/30/2022 02:32 PM - Greg Shah

My idea was to add an extra layer at the ConversionDriver level, which would associate the profile with some sub-set of total file list. As each file is processed, the proper profile would be used and all schemata/configuration would come from that profile.

#35 - 04/30/2022 02:49 PM - Constantin Asofiei

Greg Shah wrote:

> My idea was to add an extra layer at the ConversionDriver level, which would associate the profile with some sub-set of total file list. As each file is processed, the proper profile would be used and all schemata/configuration would come from that profile.

Thanks, I think I may be able to make it work for the front phase. But the conversion part may be tricky.

#36 - 04/30/2022 02:53 PM - Greg Shah

We can defer any conversion issues until a second phase of work. The front end part is what is needed this week.

#37 - 04/30/2022 05:20 PM - Constantin Asofiei

Greg Shah wrote:

> We can defer any conversion issues until a second phase of work. The front end part is what is needed this week.

I think it works, kind of ugly IMO, but with minimal code for the front phase. I don't see a way to switch profiles without explicitly 'injecting' code like 'cfg.withFileProfile(file)', which does all the underlying work of switching the Configuration and SchemaConfig.

I'll test and commit tomorrow.

#38 - 05/01/2022 11:43 AM - Constantin Asofiei

- % Done changed from 0 to 60

The current changes are in 6129a/13823.

#39 - 05/02/2022 10:24 AM - Ovidiu Maxiniuc

- Assignee changed from Constantin Asofiei to Eric Faulhaber

- vendor_id deleted (GCD)

Eric Faulhaber wrote:

> Ovidiu, do I understand correctly that when the importFile attribute of namespace is specified, schema conversion artifacts (e.g., the *.dict, *.schema, and *.p2o files) now will all be stored alongside the *.df file found at that same importFile path?

No. This would make the data source directory dirty. The artifacts are save in $cvtpath in using a path similar to original .df file. For example, if we have:

```
<namespace name="primaryDatabase" importFile="data/some/strange/location/my-database.df">
```

and the $cvtpath variable was not set, then the *.dict, *.schema, and *.p2o will be found as:

```
cvt/data/some/strange/location/my-database.dict
cvt/data/some/strange/location/my-database.schema
cvt/data/some/strange/location/my-database.p2o
```

The schema hints used for this is a input resource so it should be located right beside the .df as:

```
data/some/strange/location/my-database.df.hints
```

It will also work (temporarily) as:

```
data/some/strange/location/my-database.schema.hints
```

Greg Shah wrote:

> I don't think there is a hard requirement for that at the profile level.

The default profile is used in case none is specified in command-line arguments. Of course, it makes sense when there are multiple profiles defined.

The implementation idea before 6129a was that a combination of schema is desired, a specific profile would be written, this way avoiding conflicts (like adding same namespace from these new profiles).

#40 - 05/02/2022 10:25 AM - Ovidiu Maxiniuc

- Assignee changed from Eric Faulhaber to Constantin Asofiei

- vendor_id set to GCD

Sorry, failed refresh.

#41 - 05/02/2022 11:02 AM - Roger Borrello

If there were a need for 2 main database configured:

1. a **merged** configuration with all schemas in one database, named main_db
2. a **multi** configuration with all schemas spread into multiple databases, named main_db (which is a different schema than the above), multi_1, multi_2, and multi_3

And there is a "common" database between the 2 named common_db.

It would make sense to me to keep the DF files in a structure in the project such as:

- **data**/**merged** which contains just the main_db
- **data**/**multi** which contains the main_db, multi_1, multi_2, and multi_3
- **data**/**merged_multi_common** which contains the common_db

What would the p2j.cfg.xml configuration look like? Right now it looks like:

```
   <schema>
      <profile name="merged" default="true">
         <namespace name="standard" />
```

```
         <namespace
            name="common_db"
            default="false" >
            <parameter name="ddl-dialects" value="postgresql" />
         </namespace>
         <namespace
            name="main_db"
            default="true" >
            <parameter name="ddl-dialects" value="postgresql" />
         </namespace>
      </profile>

      <profile name="multi" default="false">
         <namespace name="standard" />

         <namespace
            name="common_db"
            default="false" >
            <parameter name="ddl-dialects" value="postgresql" />
         </namespace>
         <namespace
            name="main_db"
            default="true" >
            <parameter name="ddl-dialects" value="postgresql" />
         </namespace>
         <namespace
            name="multi_1"
            default="true" >
            <parameter name="ddl-dialects" value="postgresql" />
         </namespace>
         <namespace
            name="multi_2"
            default="true" >
            <parameter name="ddl-dialects" value="postgresql" />
         </namespace>
         <namespace
            name="multi_3"
            default="true" >
            <parameter name="ddl-dialects" value="postgresql" />
         </namespace>
      </profile>
```

My build.xml goes through the effort of copying the correct set of DF files from either data/merged or data/multi to the data/ directory to be alongside the common_db.df file, which is kept in that location.

I'd like to just place the DF files as mentioned, and leave them be during the conversion process. What should my configuration look like?

As far as the dump files, there is a singular collection of all the dump files, and there are symbolic links to the same collection for dump/multi_1/, dump/multi_2/, and dump/multi_3/ so I don't believe that aspect of configuration should change.

#42 - 05/04/2022 10:56 AM - Greg Shah

- Related to Feature #6320: sharded schemata added

#45 - 06/25/2022 11:09 AM - Greg Shah

Our initial support for multi-profile conversion needs to be enhanced. FWD supports conversion of separate profiles, but it does not allow merging all these converted profiles (with one or more of them representing an application module) and run them together, in the same FWD server.

- #6407 is part of the solution. (name_map.xml in 'extension jars')
- We need to handle the fact that per profile state like registry.xml and all the .p2o/.schema needs to be merged in some way.
- Different profiles may reference overlapping sets of OO 4GL classes. We probably need some concept of which profile is authoritative for these classes so that they can be only converted once. Doing so is a problem because the only mapping we have from 4GL OO features to Java OO features is calculated during parsing/conversion itself but the dependent projects must have access to this in order to convert. The current approach converts overlapping OO 4GL classes multiple times and has no way to merge them into a single result.

The objective here is to have two or more completely different converted applications which may or may not use the same schema/database, and we want to:

- Convert and compile each one independently, with the minimum dependencies on the others.
- Run them in the same FWD server at runtime with no duplication of classes or conversion artifacts/state.

As one customer notes: in Java this can be accomplished easily by just referencing a jar in your development environment. For that to work, we would have to package up enough of the conversion artifacts/state to allow it to be referenced in another "application". Doing this would make very large organizations able to split their development across tens or even hundreds of different development groups without each one having to convert the entire set of all dependent applications. This is a critical objective.

What would be required is that the dependencies would be converted first. Then the jar(s) (which have the converted code) would need to be available. Crucially, some amount of conversion knowledge about how the 4GL dependencies converted would be required. For example, we would need to know how each externally visible method and data member mapped to the converted version. Some amount of this (maybe all?) could be read from our class and method annotations (e.g. LegacySignature).

#46 - 03/03/2023 02:30 PM - Greg Shah

- Related to Feature #7169: drive conversion order using user-specified dependencies added

#47 - 03/30/2023 06:51 AM - Constantin Asofiei

There is a problem when the 'exclude' filter is used to not add files to conversion. During pre-scan phase, the filter is not checked, but instead a physical file on disk (based on the configured PROPATH). And, that file is used (if found), even if is excluded via the profile 'exclude' filter.

#48 - 03/26/2024 10:56 AM - Greg Shah

- Related to Feature #8525: eliminate the runtime dependency on registry.xml added