Base Language - Feature #4762

localize system error messages

07/15/2020 12:48 PM - Greg Shah

Status:	New	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**version:**	
vendor_id:	GCD		

Description

Related issues:

Related to Base Language - Feature #3817: create resource bundles from string...	**Closed**

History

#1 - 07/15/2020 01:24 PM - Greg Shah

We know that 4GL system error messages are localized. They have different message source files that are somehow selected at runtime.

We need to find out if this varies by the current setting (at error time) of SESSION:CURRENT-LANGUAGE, by SESSION:CURRENT-LANGUAGE at the time the failing code was loaded/RUN or if it is global to the 4GL process (based on locale). As far as I understand it, the value does not change based on CURRENT-LANGUAGE but instead based on the PROMSGS "function" and "language statement". Let's confirm this.

The solution to be implemented:

- The error messages (which are currently hard coded into our source files) should be moved into resource bundles.
- Implement the mechanism for switching message bundles based on whatever condition has been found above.
- Review all current locations where any hard coded text is output to a user or log, including where errors or warnings are raised. Based on the existing use cases:
  - Create a set of helper functions to raise the errors or warnings.
  - Instead of hard coding the error text we will pass in an ID that maps to the resource bundles.
  - Instead of the caller handling the string formatting, the caller must pass in any substitution values and the helpers must handle the substitutions using the localized message text. Varargs can be used here to handle the substitution lists.
  - The helper will then raise the error or warning with the same features as the original ErrorManager usage.
  - We can discuss whether these helpers should be in ErrorManager or in a new class.

As part of this task we should also plan to implement:

- PROMSGS builtin function (e.g. message promsgs.)
- PROMSGS assignment-style language statement (e.g. promsgs = "prolang/promsgs.eng".)

We can split off the non-legacy text (things that aren't error or warning messages controled by PROMSGS) but we probably want to have them in the same resource bundles. We also need to plan for use in both Java and Javascript.

#2 - 07/15/2020 01:35 PM - Marian Edu

Greg Shah wrote:

> We know that 4GL system error messages are localized. They have different message source files that are somehow selected at runtime.
>
> We need to find out if this varies by the current setting (at error time) of SESSION:CURRENT-LANGUAGE

Greg, this has nothing to do with CURRENT-LANGUAGE (used by tranman) but with the PROMSGS - [[https://knowledgebase.progress.com/articles/Article/P68929]].

#3 - 04/08/2021 10:00 AM - Ovidiu Maxiniuc

Starting with my work for datasets I added some new APIs in ErrorManager which allows to raise (throw/display) errors without hardcoding the error message locally. The current methods are:

```
public static void recordOrShowError(int id, String... params)
public static void recordOrShowError(int id, boolean isError, String... params)
public static void throwError(int id, String... params)
public static void recordOrThrowError(int id, String... params)
```

Notice the last variable length parameter. It is used for passing a set of String parameters which the private method replaceTokens() will use to replace the tokens for the specific message (identified by the id received as parameter).

At this moment there are about 250 error messages located in a big switch in ErrorManager, all were captured directly from 4GL user interfaces, only the tokens were replaced, instead of bracketed name, an index was used. I am sure they are not using them in that format because some of the errors extracted have the token completely missing. Also, the indexed approach allow the message to easily swap order of tokens in different languages. It is not mandatory to have the token in same order in the printed message as they were passed, but they are uniquely identified.

This is just the first step toward fully I18n of the error messages. The big switch can be now easily extracted to:

- a xml file (this may be easier to develop as it can be altered externally and reload on live server);
- a table in a database (that would be nice, but it adds some extra management and database trips with the only advantage of index access) or
- even simpler, kept in a language class in same format as it is now (no extra effort and the access is faster than internal database index because of the switch) the only limitation here might be the bytecode/class/switch size in Java.
  I would prefer the latter because it has the fastest access and I am sure some attributes or code specific to each language will be needed.

These resource can be dynamically configured, even for each user context. Additional API will have to be added. Certainly some client-side parameters will be, and they must be relayed to server-side during the initial handshake protocol.

At this moment there are a lot of APIs and parameters which include: prefixing with **, adding a final .. They should be dropped. These 'decorations' are part of the error message and a specific error message always has the prefix (only first 500 errors or so) or final dot (somebody at OE who wrote the error message simply forgot to put it there). Adding parameters for these only complicates the interface and adds extra CPU clocks for processing them.

One final issue. I chose to overload the existing methods so for a specific number of (variable) arguments the compiler will prefer the old API. This is the cause in some places I added additional arguments, even if the respective error message has fewer parameters/tokens. However, once we get of the old API this will not be an issue any more.

#4 - 04/08/2021 10:07 AM - Greg Shah

> This is just the first step toward fully I18n of the error messages. The big switch can be now easily extracted to:

I want to move to a resource bundle approach. This will be compatible with our efforts in #3817.

> At this moment there are a lot of APIs and parameters which include: prefixing with **, adding a final .. They should be dropped.

Agreed.

> One final issue. I chose to overload the existing methods so for a specific number of (variable) arguments the compiler will prefer the old API. This is the cause in some places I added additional arguments, even if the respective error message has fewer parameters/tokens. However, once we get of the old API this will not be an issue any more.

I understand and agree.

#5 - 08/20/2021 07:50 AM - Greg Shah

- Related to Feature #3817: create resource bundles from string literals and implement optional support for setting values from the translation manager database added