Subject: Embedded methods in objects
Posted by Claus Feyling on Mon, 04 Feb 2019 18:27:23 GMT
View Forum Message <> Reply to Message

Dear all!

This Forum post is just the starting point for a discussion on the scope of railML – is the railML infrastructure schema just enoding a “static picture” of an infrastructure, or should we allow it to contain computer code meant for evaluation within a railML evaluation system?

We suggest that a new element called ‘formula’ is added to every object in railML. The cardinality is 0..*, but just one such element will be relevant for each attribute within the object. Deleting the object from a model will also delete its formulas. The immediate purpose of such an element is to allow standardisation of a versatile data structure based on railML, where the object’s behaviours are modelled along with their static values. In this way, a railML model may be loaded into a host system and then modified partly by entering new data values directly in the object’s attributes, partly by letting the objects calculate their own values.

Formulas are associated with attributes, using the mandatory attribute ‘name’.

The railML writing system decides whether to store attribute values only, or attribute values as well as their formulas.

We suggest that the open source scripting language Lua is defined as the default scripting language within railML.

We suggest that the default evaluation depth is 1, see more about this below.

The purpose of including such definitions in railML is to allow a railML writing system to store a model with formulas that can later be picked up and used or further refined by another railML reading system.

As an example, the ‘code’ attribute of a switch may be associated with a formula as follows in order to receive a computed value based on its ‘seq’ attribute:

```
<formula name=”code” encoding=”lua” depth=”1”>
 if dir == ‘up’ then
  return 500 + (2*seq+1)
 elseif dir == ‘down’ then
  return 500 + (2*seq+0)
 end
</formula>
```

We here assume that there is consensus on how to name relevant scripting languages. “Lua” is an open scripting automation language defined originally by Petrobras, and later picked up by the gaming industry. Other scripting languages may be defined.

Suggested guidelines for interpreting formulas inside the railML evaluation system:

Every railML attribute may have zero or one associated formula (a small program returning an adequate data structure). If an object has no formula, then the action of reading that value will return its last stored value. If an attribute has a formula, then this formula can be triggered for evaluation in order to update the attribute’s value before it is returned to its reader. The reader decides whether it wants to receive the existing value or whether the formula shall be evaluated first. The action of reading a value can trigger a chain of value reading events which can nest very deep until values are read for which there are just empty formulas. The reader may pass on an evaluation depth, zero or a positive integer, default 1, along with the reading request, defining how deep the evaluation of formulas shall reach.

Best regards,
--
Claus Feyling
Daglig leder
CEO

Railcomplete AS
Kontor: Vestfjordgaten 4, N-1338 Sandvika
Firmapost: Brageveien 4a, N-0358 OSLO
+47 908 24 018
www.railcomplete.com

Subject: Re: Embedded methods in objects
Posted by christian.rahmig on Mon, 11 Feb 2019 14:58:55 GMT
View Forum Message <> Reply to Message

Dear Claus,

Am 04.02.2019 um 19:27 schrieb Claus Feyling:
> [...] We suggest that a new element called ‘formula’ is added to every object
> in railML. The cardinality is 0..*, but just one such element will be
> relevant for each attribute within the object. Deleting the object from
> a model will also delete its formulas. The immediate purpose of such an
> element is to allow standardisation of a versatile data structure based
> on railML, where the object’s behaviours are modelled along with their

> static values. In this way, a railML model may be loaded into a host
> system and then modified partly by entering new data values directly in
> the object’s attributes, partly by letting the objects calculate their
> own values.

thank you very much for your ideas and input! The approach sounds
interesting.

> Formulas are associated with attributes, using the mandatory attribute
> ‘name’.
>
> The railML writing system decides whether to store attribute values
> only, or attribute values as well as their formulas.
>
> We suggest that the open source scripting language Lua is defined as the
> default scripting language within railML.
>
> We suggest that the default evaluation depth is 1, see more about this
> below.
>
> The purpose of including such definitions in railML is to allow a railML
> writing system to store a model with formulas that can later be picked
> up and used or further refined by another railML reading system.
>
> As an example, the ‘code’ attribute of a switch may be associated with a
> formula as follows in order to receive a computed value based on its
> ‘seq’ attribute:
> <formula name=”code” encoding=”lua” depth=”1”>
> if dir == ‘up’ then
> return 500 + (2*seq+1)
> elseif dir == ‘down’ then
> return 500 + (2*seq+0)
> end
> </formula>

To be honest, I have not heard of such "formulas" before and
consequently my know-how is very limited in this field. But how about
the other railML users with export and import interfaces? Do you
consider formulas as essential keystone for the railML standard and do
you have experiences with "Lua" or similar approaches?

Any feedback is highly appreciated...

Best regards
Christian

--
Christian Rahmig - Infrastructure scheme coordinator

railML.org (Registry of Associations: VR 5750)
Phone Coordinator: +49 173 2714509; railML.org: +49 351 47582911
Altplauen 19h; 01187 Dresden; Germany www.railml.org