Xaraya
Request for Comments: 0007
Category: Informational

G. Rothfuss
Project Management Core
J. Schrage
Xaraya Development Team
January 2002

Modularized Data

Status of this Memo

This memo provides information for the Xaraya community. It does not specify an Xaraya standard of any kind. Distribution of this memo is unlimited.

Copyright Notice

Abstract

This RFC proposes to have a unified mechanism to extend modules with custom data.

Table of Contents

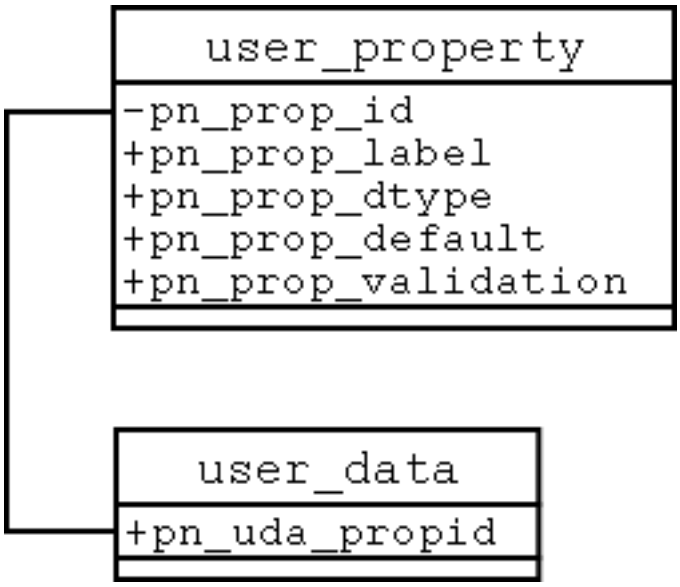
1. Introduction

2. General statements about Modularized Data

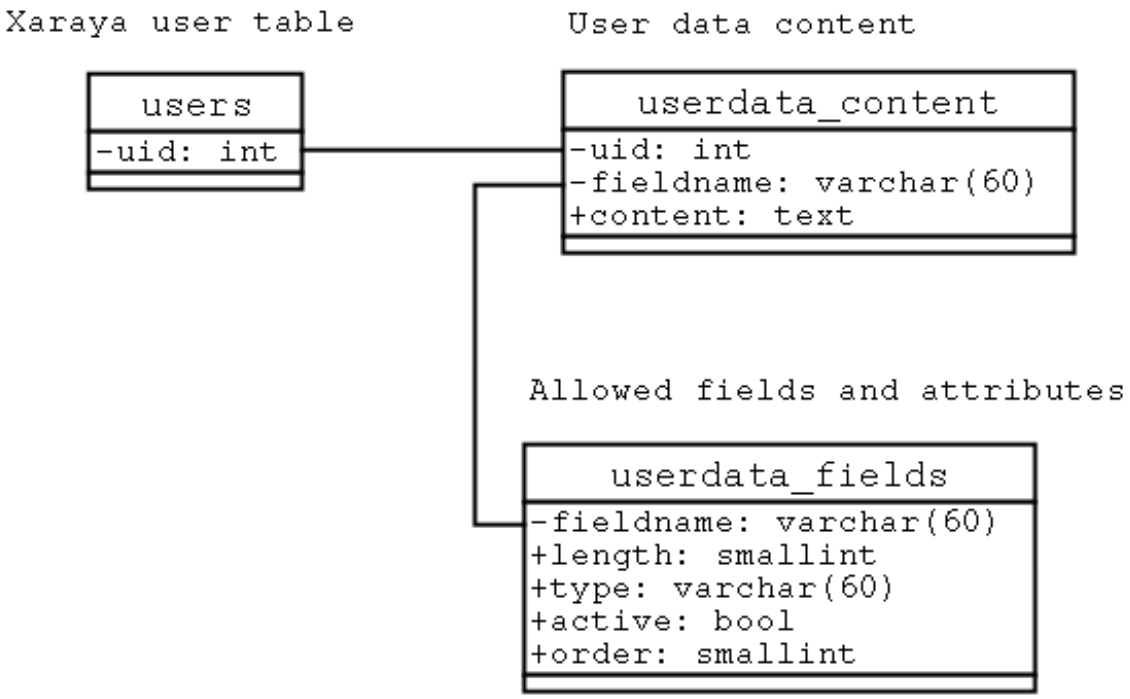
3. List of requirements for Modularized Data

4. Solution proposals - database tables

This is the DB setup as extracted from the admin API.



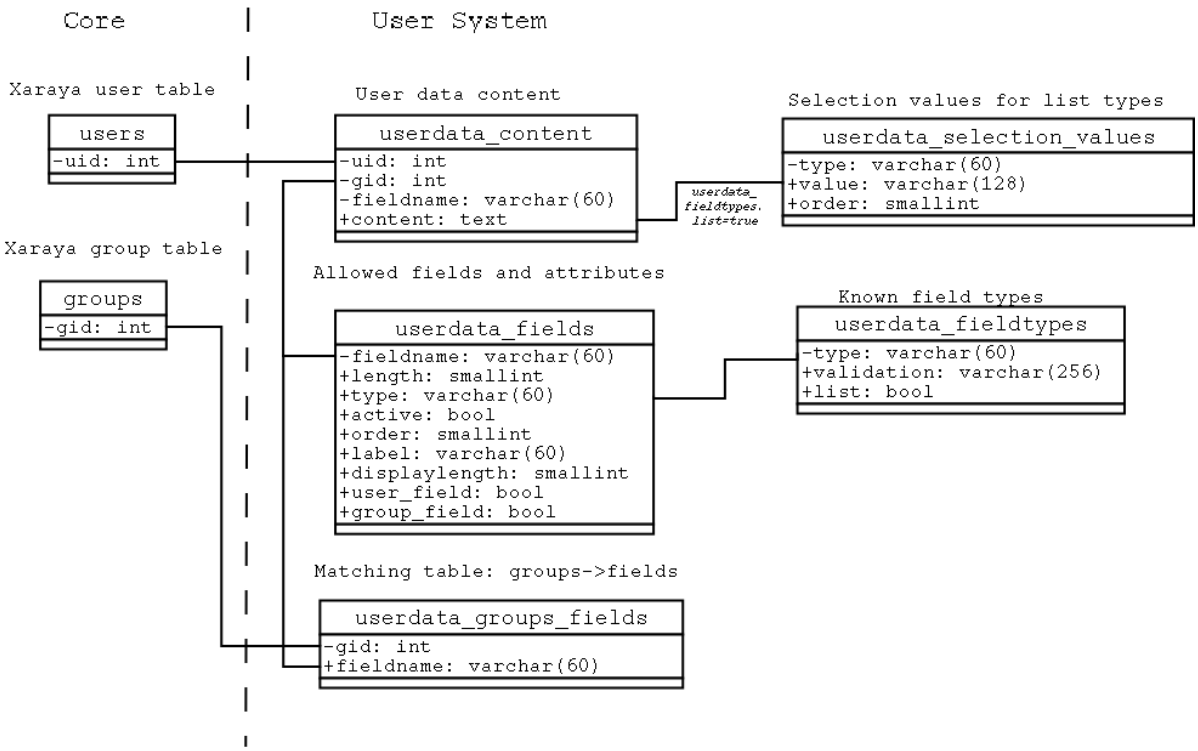
This design is for text-based dynamic data.



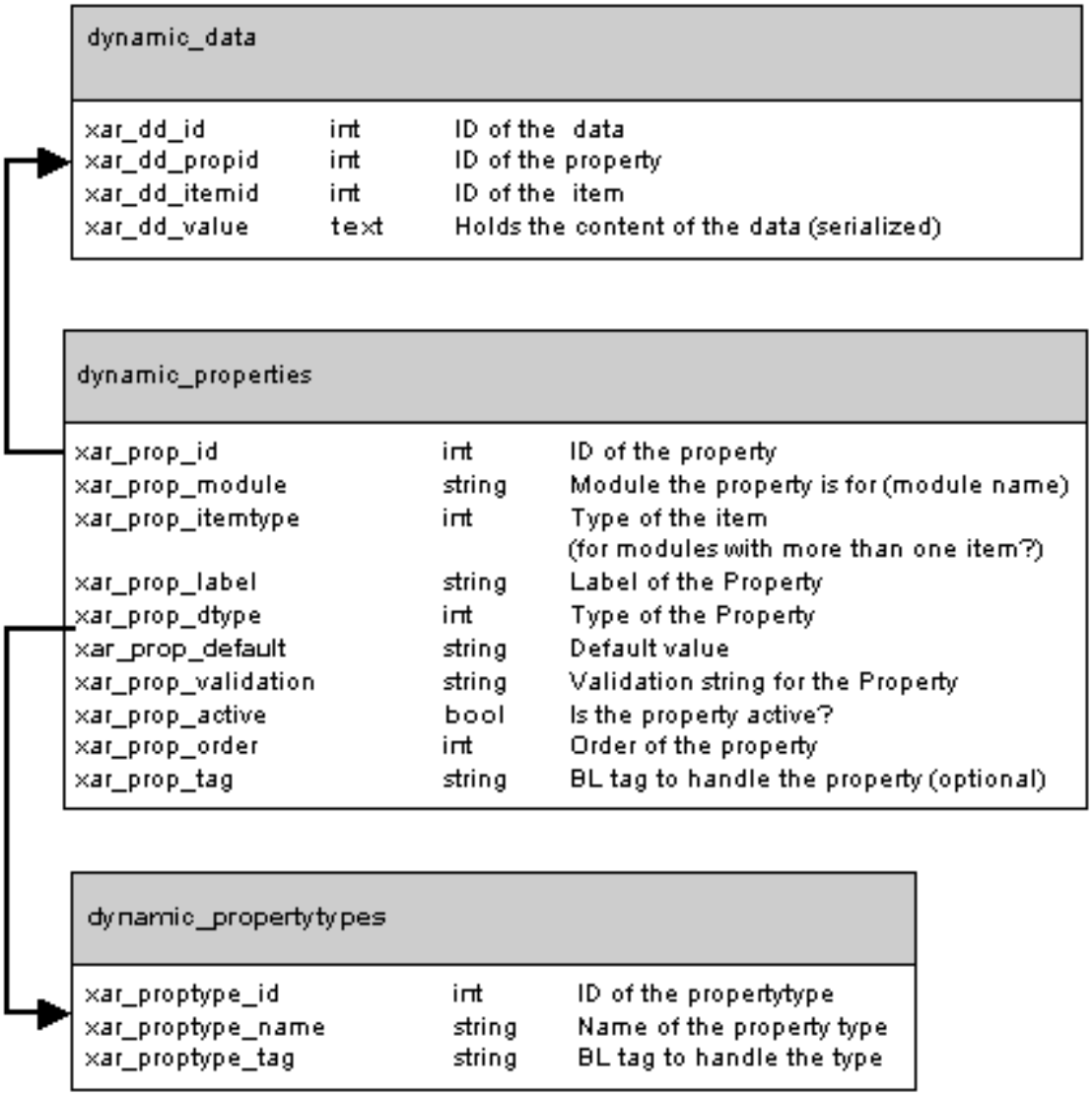
Q: Allow certain fields for certain groups only ?

This design is for user data that can be assigned to groups, with optional selection lists for fields, i.e. for

dropdowns.



This design attempts to merge Dynamic Data, Dynamic User Data and Module Variables. Values are serialized to support transparent storage of complex types (arrays etc). Each type has a default BL tag for rendering, which can be overwritten in the property. Properties are referenced by module name.

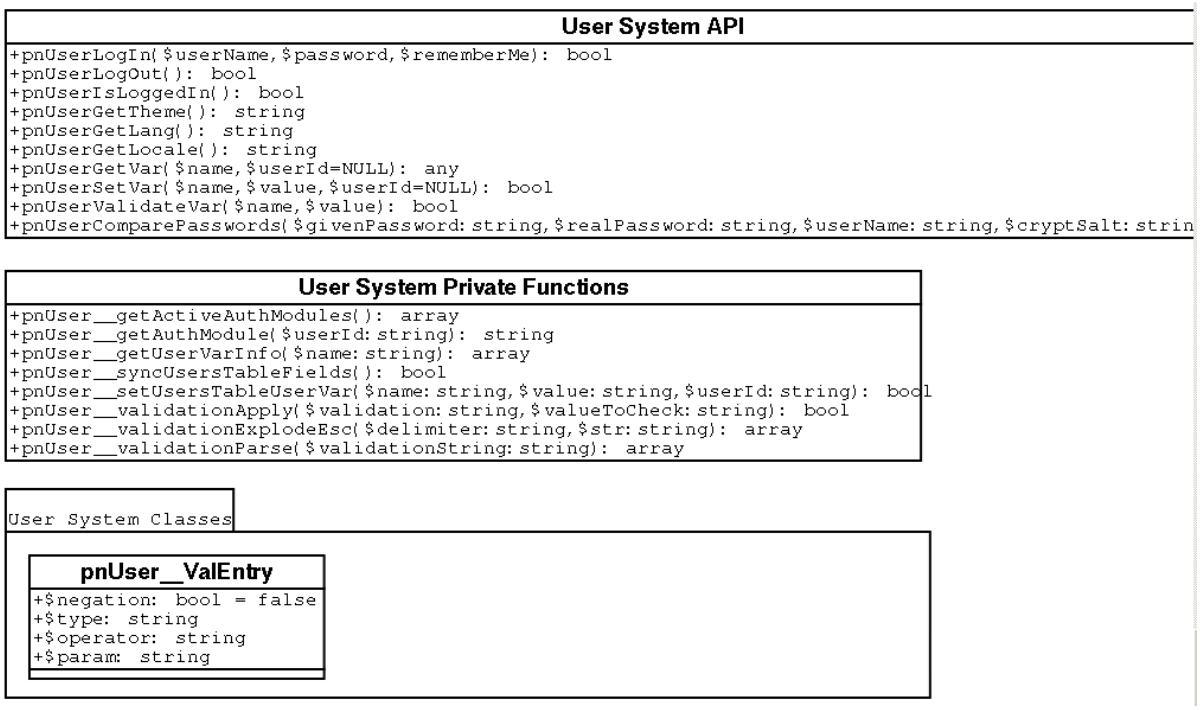


Proposed change: Remove field "validation": Validation aspects should be handled by the data dictionary.

5. Table 'Content'

6. Solution proposals - APIs

Dynamic Data will use two sets of APIs. Creation and Deletion of of variables is done within a module, while reading and writing variable contents is done in the core. There are currently discussions whether to combine Dynamic User Data with generalized Dynamic Data, in essence abstracting away the relationship between users and dynamic data, and allowing every module to define its own dynamic data. This brings up the question if module variables are being reinvented, and what should happen with the functionality to assign responsibility for storing user variables to authentication modules. The current core API functions are:



6.1 Review of the current DD API with extension proposals

function dynamicdata_adminapi_createprop($args)

modid - module id of the item field to create

itemtype - item type of the item field to create

label - name of the field to create

type - type of the field to create

default - default of the field to create

validation - validation of the field to create

Proposed changes: Add

active - boolean: is this property active, i.e. usable?

order - int: sort order for display

tag - string: BL tag that handles the output

Delete

validation: handled by data dictionary

function dynamicdata_adminapi_updateprop($args)

prop_id - property id of the item field to update

modid - module id of the item field to update (optional)

itemtype - item type of the item field to update (optional)

label - name of the field to update

type - type of the field to update

default - default of the field to update (optional)

validation - validation of the field to update (optional)

Proposed changes: Add:

active - boolean: is this property active, i.e. usable?

order - int: sort order for display

tag - string: BL tag that handles the output

Delete:

modid: A property that belongs to a specific module should remain there

itemtype: Update of item types may break the module logic. Should be handled by deletion and new creation

type: Likewise.

validation: handled by data dictionary

function dynamicdata_adminapi_deleteprop($args)

prop_id - property id of the item field to delete

modid - module id of the item field to delete

itemtype - item type of the item field to delete

label - name of the field to delete

type - type of the field to delete

default - default of the field to delete

validation - validation of the field to delete

Proposed changes:

- Delete all parameters except prop_id and modid: For security check ensure a module can delete only its own properties. It should, however, be allowed to delete any of them.

Proposed new functions:

function getAllProperties(modid): get a list of all properties for module $modid and their basic type and BL tags. returns array(propid,type,tag)

7. Relationship to other areas

It will need to be determined if it makes sense to allow storage of certain properties via the modular authentication system. For instance, it could be beneficial to store dynamic properties for the users module in LDAP. Also, it needs to be determined how and if dynamic properties integrate with the Multilangue system. Marco has some thoughts about this, i hope he will share them soon. Property validation should be done by using the facilties provided by xarVarValidate.

8. Code that will need to be rewritten

xarUserGetVar, xarUserSetVar and xarUserValidateVar will go away, to be replaced by xarModGetVar, xarModSetVar, xarVarValidate respectively.

9. Tools that need to be created from scratch

10. Basic documents for this RFC 7

11. Retractions

12. Author contact

13. Changelog

Authors' Addresses

Gregor J. Rothfuss
Project Management Core
EMail: gregor@xaraya.com
URI: http://www.xaraya.com

Jan Schrage
Xaraya Development Team
EMail: jan@xaraya.com
URI: http://www.xaraya.com

A. Additional comments

A.1 Introduction

Dynamic data (or any data in fact) generally goes through several steps :

1. definition : the admin (or the system) defines some dynamic property for a module item type
2. input : the user/admin can enter or modify the data in a form
3. validation : the system verifies that it has an acceptable value
4. storage : the data is saved/updated in the database, or perhaps it is passed along to some external store (e.g. other DB, LDAP, web service, ...)
5. retrieval : the data is retrieved from the database, or from some other external store
6. display : the data is made available for display by the module

Xaraya should facilitate *all* these steps, and it seems we're still missing a few pieces of the puzzle here.

I'll start with what we should be able to do from a user/admin point of view, and then go back to the definition part, to see what kind of information we need in order to define all this.

A.2 User input

This implies showing some kind of form field of a certain type (or some equivalent like a date selector or image picker), with a particular field name (or names), some initial content, possibly some list of acceptable values, and so on.

One approach is to let BL do all the work, and use one common tag that handles all the proper formatting for you. An example of that is the (current) articles-field tag, that receives a name, property type and value from the module, and handles all the rest automatically. That includes providing the right formatting, retrieving the list of acceptable values based on the property type ("field format"), or in some cases even retrieving the value itself (e.g. for username or date).

For the moment, there is no explicit support for retrieving a list of acceptable values by individual property, rather than by property type, but this is certainly going to be necessary if you want to provide even simple things like dropdown lists etc.

And going one step further (cfr. FormExpress), you could imagine defining a function when creating your property, and let BL invoke that function to retrieve the list for you.

The BL template is reduced to a simple loop :

```
            <xar:loop name="$fields">
            <tr align="left" valign="middle">
            <td>&xar-var-label;</td>
            <td><xar:articles-field definition="$definition" /></td>
            </tr>
            </xar:loop>
```

and with a better definition of properties, you could even replace that by a single tag like

```
<xar:articles-form module="articles" itemtype="1" itemid="1234">
```

That's one extreme, and I'm not far from being able to do just that ... for articles. You could imagine doing the same thing for any module, with the right support of property definitions by Xaraya.

Advantage : you define your properties, and Xaraya does the rest.

Disadvantage : no fine control over how the form is created, except by modifying how the individual fields are generated by that tag.

But by allowing admins to modify the formatting of individual properties (and/or making use of widgets in the future), non-designer people [like me :)] can let Xaraya create the whole input interface.

The other extreme -the "classic" approach I came from- is of course to build your whole form yourself in the module template, retrieving the data, acceptable values etc. in your module. This works fine as long as you know beforehand which fields you're going to have, but once you start dealing with dynamic data, this may no longer be valid.

That means that you have another choice to make :

a. either those dynamic fields should appear "automagically" in the input form via hooks, and the creation of the fields should be done via some mechanism similar to what I described above for articles,

b. or the module should be able to retrieve the dynamic data itself and the admin will then manually add the additional form fields in the template himself

For the moment, option a. is used in the dynamic data module, but option b. should be made available as well. And of course, we could imagine having some generator build a sample template based on the dynamic properties defined for that module...

A.3 Validation

The xarVarFetch() and xarVarValidate() functions already provide some basic validation based on data type, and also some verification of allowed HTML tags (cfr. other on-going thread), but it doesn't come anywhere near the kind of validations we'll need for dynamic data yet.

Some sample validations I'm thinking of here :

- is it a existing user ?
- is this a valid date (year, month, day, hour, min, sec) ?
- does this belong to that list of acceptable values ?
- is this an acceptable URL ?
- is this a local image from that directory where you're allowed to pick some image from ?
- is this text not too long ?
- is that a valid status change ?
- is that really a webpage that you're trying to retrieve ?
- is this a valid file upload, of those acceptable file types, size, ... ?
- ...

So far, most of the validations in articles are pretty simplistic too, but that's the kind of input validation we will need if we want to provide more than really basic dynamic data.

And as mentioned before, some validations may be on the level of property types, others may require specific rules per individual property (for a particular module + item type), and others might be "filled in" dynamically by the module for each item we're dealing with. So Xaraya should support validation rules on all three levels.

Another aspect is *what* data to validate. Each module knows what particular data it has to receive on input/modification, but of course this list is dynamic for ... dynamic data. This is (again) handled via hooks at the moment, but in some cases, there might be properties that *may not* be modified, either because it is not a valid input field (e.g. the current user is fixed), or because the field (e.g. the article status) may not be modified by this particular user (permissions).

So another aspect of property definition is that the admin (or the property type) should be able to decide whether this is a valid input at all, or who is allowed to input/modify it - something I didn't cover in 1) User Input, but has an impact there as well, obviously.

And since we're pushing the envelope anyway, why not consider the situation where input validation for a module could be done without requiring any coding by the module developer as well ? After all, if Xaraya is (someday) able to display a complete input form based on property definitions, it should be able to handle the results as well, no ? [you're probably way ahead of me by now :)]

Again, this is not something that should be mandatory (forced by the core), but a convenience...

A.4 Storage and retrieval of data

Okay, so we have our dedicated tables for each module, and we have some ways to retrieve (user) data via LDAP or other means, and now we have some generic dynamic data table where people can store additional stuff if they want to extend a module.

And in the future, it would be nice if we could get data from other places as well, like some other external databases, or information made available via web services, in files, whatever.

Now unlike some other projects I won't mention :-), I don't think that putting *everything* in some generic meta-table is the best approach. Dedicated tables have their use, and we shouldn't get rid of them just for the sake of "ultra-flexibility".

But one thing that would really allow us to go beyond the "my module, my tables" stage would be to improve and extend the way user data can now be retrieved from different "sources" to [at least] allow you to do the same for dynamic data in general.

Basically, what we need then is some "mapping" mechanism between the logical property definitions used by Xaraya, and the physical access to that data in local or remote databases, via LDAP, and so on. And of course, "connectors" providing the corresponding access mechanisms to each of those "data stores".

Putting this into place will probably not be done overnight, but right now, there are a few blocking points in the core that make this nearly impossible to even start implementing.

And the most important one (in my point of view) is the correct handling of multiple database connections - I'm by no means an expert on PHP ADODB, but is it really so hard to make sure that module developers can (if they want to) open up another database connection without Xaraya loosing its own database connection along the way ? I haven't tried it myself with Xaraya, but this seems to be a very common complaint for PostNuke module developers, and somehow I don't think we've solved that particular issue yet - or have we ?

Anyway, besides some of the more fancy stuff we might implement someday [like importing meta definitions from existing sources and automatically making them available as property definitions for use in dynamic data], we could also imagine automating the select, insert, update, delete methods - not only for dynamic data itself, but also for dedicated module tables.

Again, not something mandatory, but another feature that Xaraya would provide for module developers and/or admins, so that once they've defined which properties their data should have, they could access it transparently via the core. And a step further of course, that they could do it for any data, regardless of where it comes from...

In the short term (for RFC 7), this means that dynamic properties may not always have their data (per item) stored in the dynamic data table, but the data might be located elsewhere - so it should be possible to say where it's coming from, and where it's going to be stored. We can then work on providing different "connectors" as we move along, making use of what's already been done in the context of the auth* modules - and then turn the situation around and let some of the auth* modules make use of the generic connectors.

That's it for data storage and retrieval - now let's finally do something with that data :)

A.5 Displaying data

One thing that's seriously lacking in the current dynamic data implementation is how to handle the actual

display of that data. So far, I'm just creating a name - value list and passing it along via the display hook, which obviously isn't good enough.

Now, remember those different templating options mentioned for User Input. Assuming we had a "display field" tag for each property, as well as an "input field" tag, we could (theoretically) provide a similar approach like :

```
            <xar:loop name="$fields">
            <xar:articles-displayfield definition="$definition" />
            </xar:loop>
```

or even

```
<xar:articles-display module="articles" itemtype="1" itemid="123" />
```

Again, this could be built up automatically by Xaraya based on the property definitions, by providing some default display for each property type. Of course, people aren't likely to be happy with the way things would be displayed "out of the box", but this would at least provide some basic configurable output, and a stepping stone towards full customisation.

For item lists and displays, module developers, designers and admins are much more likely to adapt templates to suit their needs, so here it's essential that they can handle the display of dynamic data directly, which means we need to :

- have some BL tags that allow you to get some default display for each dynamic property, or
- have some BL tags that allow you to get the data itself for each property in the template, or
- be able to automatically retrieve the dynamic data in the module and make it available for the template like any other variable, or
- any combination of the above ;)

Frankly, I'm not sure which of the above should be our first priority, and which one(s) should be supported in the end.

Ideally, the fact that there is dynamic data available or not should be made transparent if the admin is happy with some "standard" presentation, but he should be able to fully customise how and where the dynamic data will appear otherwise...

If we used transparent data wrappers that can hide where the data is coming from (dedicated table, dynamic data, ... see part 3), this would be less of a problem, but I don't think we can rely on that in any reasonably short period of time.

Either way, there's nothing implemented in the dynamic data module for this beyond some rudimentary display via hooks, so I'm certainly open for suggestions or ideas on how to proceed on the display part at this point !!! :-)

That's about all I can think of about using dynamic data from the point of view of users/admin - now let's see what all this means in terms of property definitions, and the design of the DD module in general. Which leads us to the last part of this RFC-about-an-RFC :)

A.6 Putting it all together

Based on all of the above, here's how I see us moving forward with property definitions and dynamic data :

1. in core (some new 'datadict' module, perhaps ?), define standard property types (in the sense of articles, not in the sense of xarVar or users module) and allow the creation of new property types.

This includes support by a BL tag to generate a standard input field (or equivalent), advanced validation rules (implemented with extended xarFetch and xarVarValidate), and support by some other BL tag to generate a standard display as output.

Ideally, there should be 1 common BL input tag and 1 common BL display tag, possibly supported by a bunch of widgets (Xaraya 1.1) or whatever for easier customisation afterwards.

2. in core ('dynamicdata' module or other), modules/admins can register properties for a particular module + item type. In the short term, this covers only dynamic data, and this module is optional. In the longer term, this also covers the "static" properties (=dedicated tables) of each module, and it will no longer be optional.

For each property, modules can use the common BL input and display tags, or specify some specific input and display tags (to be defined how).

For each property, modules can also "fill in" the generic validation rule of the corresponding property type, where necessary. Example : the acceptable values for a list, the base directory to look for images, a callback function, or whatever is relevant for that property type.

In the short term, the 'dynamicdata' module will create corresponding records in a dynamic data table for each property. In the longer term, modules will be able to specify where/how data should be stored/retrieved for each property.

3. the 'dynamicdata' module will provide functions to store/retrieve data for each of the properties defined for a module. In the short term, only with the dynamic data table - in the longer term, also with other "data stores" like dedicated module tables, external databases, LDAP, etc.

Also relevant in this context is nuncanada's proposal for new hook functions (cfr. mail of 1/1/2003 "Proposing new hooks functions").

In my opinion, the functions he proposed should *not* be seen as a replacement for the current hooks, but as data access functions to be provided by the core via the (extended) 'dynamicdata' module.

If/when each module registers its "static properties" to the core, the dynamicdata module can then create the appropriate joint statements transparently. But this would not be for Xaraya 1.0 (IMHO).

4. the 'dynamicdata' module will provide a BL form tag to show the input fields for all properties of a module+item type, and a BL output tag to show the display for all properties of a module+item type, making use of the standard BL input and display tags, or the individual tags defined for each property.
5. the 'dynamicdata' module will be able to validate input for all properties defined for that module (in the short term, only the dynamic properties, and in the longer term, every property defined for that module).
6. for more customized display of dynamic data by the modules, we need to rapidly select one (or more) of the options discussed in the previous part.
7. did I miss anything ?

Much of the short-term stuff is already available in one place or another. Missing is the central 'datadict' functionality and its corresponding BL tags and validations, and the display part. So do we go ahead with this or not ? :-)

Intellectual Property Statement

The DDF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the DDF's procedures with respect to rights in standards-track and standards-related documentation can be found in RFC-0.

The DDF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to practice this standard. Please address the information to the DDF Board of Directors.

Acknowledgement

Funding for the RFC Editor function is provided by the DDF