Xaraya
Request for Comments: 0013
Category: Informational

A Varganov
Xaraya Development Group
January 2002

Module Classification System

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

The contents of this RFC contain the literal content of the old plain text version of RFC-0013

When time is a less scarcer good, someone might convert the plain text into structured XML so we can benefit from it.

Table of Contents

1 Introduction 3

2 How to identify a module? 4

3 Module Classes 5

4 Module Categories 6

5 Module Classification IDs 7

6 Module Unique ID 8

Author's Address 9

A Module dependencies 10

B Required PHP extensions 12

Intellectual Property and Copyright Statements 13

1. Introduction

The Xaraya modules are there to extend the functionality of this content management system (we assume that you are already familiar with the Xaraya Module Developers Guide by Jim McDonald - link).

Traditionally, the modules have been added to /modules directory and activated/deactivated via the admin interface. However, they didn't have any particular weight, class or identity, which could have helped to quickly and unmistakably determine (both programmatically and visually) the module's purpose, role and hierarchy within the xaraya system.

The aim of this document is to set module classification standards for ALL xaraya modules, including old style (pre pn 0.7), new style and future versions.

2. How to identify a module?

A module can be identified by way it interacts with the core and other modules. Additionally, we need to identify modules by their virtual categories within the xaraya administration system. In this document, we are going to use and combine both approaches to form a generic set of classes and categories for all the modules.

3. Module Classes

We suggest the following list of possible module classes: Format of presentation: long name | short identifier | description | notes

Core Classes: these modules should stay with the core to provide basic functionality

- core admin modules | 'Core Admin' | admin system functions
- core user modules | 'Core User' | user system functions
- complete core modules | 'Core Complete' | user and admin functions
- core utility modules | 'Core Utility' | cross-module functions
- other core modules | 'Core Miscellaneous' | functions other than above

General Classes: these modules can be added and removed as necessary

- general admin modules | 'Admin' | admin functions only
- general user modules | 'User' | user functions only
- general complete modules | 'Complete' | user and admin functions
- general utility modules | 'Utility' | cross-module functions
- general other modules | 'Miscellaneous' | functions other than above

Misc Classes: all other modules which don't fit into the first two groups of classes

- miscellaneous other modules | 'Miscellaneous' | all other modules

Note: Each group of the above module classes can be modified, edited or removed, subject to approval by the xaraya-dev group.

4. Module Categories

Within each of the module classes, we suggest a detailed categorization in the form of 'category' system variable, which would determine the module's place and hierarchy at the Xaraya administration system level.

Examples:

Category: 'Global'

- Settings
- Banners
- Languages
- Modules
- Blocks

Category: 'Users & Groups'

- Permissions
- User Manager
- Groups

Category: 'Content'

- Add Story
- Admin Messages
- FAQ
- Polls
- Reviews
- Sections
- Topics

Category: 'Miscellaneous'

Note: If a module is not classified and categorized yet, it is auto-assigned the class 'Unclassified' and the category 'Unclassified'

Note: Each group of the above categories can be modified, edited or removed, subject to approval by the xaraya-dev group.

5. Module Classification IDs

A module's IDs within this classification system is a combination of its class and category short identifiers accessible by all core and non-core functions.

Example:

Module 'Modules' of the class 'Core Admin' and category 'Global' would have its IDs encoded into xarversion.php (or Version.php) file as follows:

```
$modversion['class'] = 'Core Admin';

$modversion['category'] = 'Global';
```

6. Module Unique ID

It has also been suggested that each module should be registered with its unique ID. When designing the module registration system, we suggest that this module classification system is taken into consideration.

Author's Address

Andy Varganov
Xaraya Development Group
EMail: andyv@msph.com
URI: http://www.xaraya.com

A. Module dependencies

In order to allow the modules system identify module dependencies and deal with them, every module must have the following definitions in it's xarversion.php file:

if module is self-sufficient (only depends on the core and core mods):

```
$modversion['dependency'] = array(); // empty array
```

if $modversion['dependency'] is not defined the system will assume that your module is not dependent on any other non-core modules

NOTE: all definitions below contain extra variables; this is done for sole purpose of simplified formatting and to avoid complexity of writing multi-dimensional arrays. However it's up to the module developer how to format the definition, as long as the meaning remains the same.

if dependent on a module (SINGLE), the short version definition:

```
$mastermodules = array(333);
$modversion['dependency'] = $mastermodules;
```

where 333 is the registered ID of the 'master' module upon which your module's life depends

if dependent on a number of modules (SEVERAL), the short version definition:

```
$mastermodules = array(333, 444, 555);
$modversion['dependency'] = $mastermodules;
```

where 333, 444, 555 are the registered IDs of the 'master' modules upon which your module's life depends

if dependent on a module (SINGLE), the expanded version definition: NOTE: we add this dimension to the definition so that we could facilitate users with extended warnings via the module system

```
$conditions = array(    'minversion' => '1.25',
                        'maxversion' => '2.1',
                        'effect'     => 'max',
                        'activate'   => 'before');

$mastermodules = array(333 => $conditions);
$modversion['dependency'] = $mastermodules;
```

where 333 is the registered ID of the 'master' module $conditions is an (optional) array of parameters which indicate degree of dependency of your module

```
  PARAMETERS (all optional):

   'minversion'
   // minimum version number of the 'master' module
   // with which your module is guaranteed to work (was tested)
   // default is '' (empty string)

   'maxversion'
   // maximum version number of the 'master' module
   // with which your module is guaranteed to work (was tested)
   // default is '' (empty string)

   'effect'
   // values can be 'max', 'min' or 'much'
   // default is 'max'

   // 'max' == your module cannot work without the 'master'
   // so the system will not allow user to activate your module
   // if the 'master' is not present/activated

   // 'min' == your module can be activated without 'master'
   // but then it will work with limited functionality
   // in other words the 'master' doesn't have much effect on yours
   // so the system will present a 'mild suggestion' to a user
   // with an 'option' to activate the 'master' module

   // 'much' == your module can be activated without 'master'
   // but you would prefer if user opt to activate 'master' as well
   // in other words the 'master' has much effect on yours
   // so the system will present a 'strong suggestion' to a user
   // with an 'option' to activate the 'master' module

   'activate'
   // values can be 'before', 'after' or 'anytime'
   // order in which the 'master' should be activated
   // in relation to your module
   // default is 'before'

    1c). if dependent on other modules (SEVERAL), the expanded definition:

$mod_A = 333;
$mod_B = 444;
$mod_C = 555;

$conditions_mod_A = array(  'minversion' => '1.25',
                            'maxversion' => '2.1',
                            'effect'     => 'max',
                            'activate'   => 'before');

$conditions_mod_B = array(  'minversion' => '1.0',
                            'maxversion' => '2.0',
                            'effect'     => 'min',
                            'activate'   => 'after');

$conditions_mod_C = array(  'minversion' => '0.1',
                            'maxversion' => '0.2',
                            'effect'     => 'much',
                            'activate'   => 'anytime');

$mastermodules = array( $mod_A => $conditions_mod_A,
                        $mod_B => $conditions_mod_B,
                        $mod_C => $conditions_mod_C);

   // and finally define the total dependency scheme
$modversion['dependency'] = $mastermodules;

    2) the module system scans thru the above array at the initialization time
    and ensures that ALL listed 'masters' loaded in defined order. It also puts
    forward 'warnings' if the 'master' mod's parameters are outside the ones
    stipulated in $modversion['dependency'] as above.

    3) at the deactivation/uninstallation time the modules system checks for
    presence of dependent 'children' mods and provides appropriate warnings if
    any found

    4) it is responsibility of the module developer to make sure that the
    supplied information is correct. The module system itself does not (and
    cannot) care which module depends on which another and how such dependency
    occurs. Rather the module system simply processes the supplied information
    and acts accordingly.
```

B. Required PHP extensions

Some modules may require specific PHP extensions that aren't included in standard PHP installations. In order to allow the modules system to identify those extensions, a module with such requirements must have the following definition in its xarversion.php file:

if the module requires a single extension:

```
$modversion['extensions'] = array('ldap');
```

if the module requires several extensions:

```
$modversion['extensions'] = array('ldap','xslt');
```

if $modversion['extensions'] is not defined the system will assume that your module does not require any specific PHP extensions

Intellectual Property Statement

Acknowledgement

Funding for the RFC Editor function is provided by the DDF