Magazine System

Version 1.0

Technical Reference

Table of Contents

Introduction

About Neredataltics UG

The Neredataltics UG is devoted to collect scholars, in a number of fields, as well as research librarians, it is evaluates the social, economic, and technical issues in the use of online infrastructure and knowledge management strategies to improve both the scholarly quality and public accessibility in a sustainable and globally accessible form. This company tries to merge emerging standards for digital library access.

About Magazine System (MS)

Magazine System (MS) is a publishing system that has been developed by the Neredataltics UG. MS assists with every stage of the refereed publishing process, from submissions to online publication. MS can improve both the scholarly and public quality of referred research. MS's purpose is to make open access publishing a possible option for more magazines, as open access can increase a magazine readership as well as its contribution to the public knowledge.

About This Document

Conventions

- Filenames, URLs, and class names indicated in a `courier` typeface;
- Square braces are used in code samples, filenames, URLs, and class names to show a same value: for example, `[anything]Addresser.inc.php` can be interpreted as any file name ending in `Addresser.inc.php`
- The URL http://www.neredatatlics.org used anywhere is intended as a fictional url.

Technologies

MS 1.0 is written in object oriented PHP (http://www.php.net) uses the Smarty template system (http://smarty.php.net).

Recommended server configurations:

- **PHP** (4.4.x or later)
- **MySQL** (4.2353 or later) or **PostgreSQL** (7.1 or later)
- **Apache** (2.3.2 or later) or **Apache 2** (2.5.4 or later) or
- **Linux**, **Mac OS X, Windows** operating systems

Design Overview

Conventions

General

Directories named by lowerTowerCase capitalization convention;

User Interface

Layout should be separated from content using Cascading Style Sheets (CSS);

PHP Code

- Global variables and functions outside of classes must be avoided;
- Symbolic constants, mapped to integers using the PHP `define` function, should be numeric string constants;
- Filenames must match class names; e.g, the `SectionAuthorAction` class is in the file

`SectionAuthorAction.inc.php`;

- Class names and variables must be capitalized as follows: Class names use CamelCase, and instances use lowerCamelCase. For example, instances of a class Your`Class` could be called `$yourClass`;
- The variable name should match the class name: For instance, `$yourClass` is better than an arbitrary name like `$x`;
- Class names and source code filenames should be informative and unique;
- To enhance performance and reduce server load, `import(...)` calls should be kept as localized as possible;

Database

- SQL tables are named in the plural (e.g. `users`, `magazines`) and table names are lower case;
- SQL databases must be kept minimal to keep wide compatibility. For example, since databases handle date arithmetic incompatibly, it is implemented in the PHP code not at the database level.

Security

- The validity of user requests is checked at the User Interface level and in also the associated Page class. For instance, when a user is not allowed to click on a button, it will be disabled in HTML by the Smarty template. If the user attempts to jump this and submits the button click anyway, the Page class receiving the form or request will ensure that it is not allowed.
- Use the Smarty template engine's string escape to ensure that HTML exploits and bugs are not allowed and also special characters displayed correctly.

Introduction

The design of MS. 10 is heavily designed for maintainability, flexibility. For this reason it may seem complicated at first glance.
As in a Model View Controller (MVC) architecture, data storage and representation, user interface presentation, and control are isolated. The main categories are as follow:

- **Smarty templates**, which are for representing HTML pages.
- **Page classes**, which accept requests from user via web browsers, redirects any required processing to various other classes, and call the respective Smarty template to display a response;
- **Action classes**, which are used by the Page classes to process user requests;
- **Model classes**, which execute PHP objects representing the system's various entities, such as Users, Papers, and Magazines;
- **Data Objects**, which update, create, and delete functions for their

associated Model classes, are responsible for all sending or receiving data from/to database;

As the MS uses inheritance and has consistent class naming conventions, it is easy to find out a particular class belongs to which category. For instance, a Data Object class always inherits from the `DO` class, has a class name of the form `[sth]DO`, and has a filename of the form `[sth]DO.inc.php`.

Request Handling

How the system handles a request from a browser is confusing if the code is examined directly, thus the use of stub files whose only purpose is to call the correct PHP class. For instance, the standard `index.php` file is in many locations, but it never performs any actual work except delegation.

Instead, work is delegated to the respective Page classes, each of which is a subclass of the `Handler` class and is in the `pages` directory.

A Note on URLs

URLs into MS use the `PATH_INFO` variable. For instance, examine the following (fictional) URL:

`http://www.ms.com/ms/index.php/magazine/user/profile`

The PHP code call to handle this request, `index.php`, seen halfway through the URL. The portion of the URL seen after this is passed to `index.php` via `PATH_INFO`.

Locating Request Handling Code

To find the code handling a particular request, follow below steps:

- Find the name of the Page class in the request URL. This is the second field after `index.php`; for example, in the following URL:

  `http://www.neredataltics.org/index.php/magazine/user/author`

  the name of the Page class is `UserHandler`. (Page classes always end with `Handler`. Note the differences in capitalization; in the URL, lowerTowerCase is used; class names are always TowerCase.)
- Find the source code for this Page class in the `pages` directory of the source tree. In the above example, the source code is in `pages/user/UserHandler.inc.php`.
- See which function is called by examining the URL. This is the third field after

`index.php`, or, in this case, `author`.

- Therefore, the handling code for this request is in the file `pages/user/UserHandler.inc.php`, in the function `author`.

Database Design

The Magazine System 1.0. database design is flexible and consistent.

Table Name	***Primary Key***	***Description***
`access_keyword`	`access_keyword_id`	Saves keys for reviewer access
`paper_authors`	`author_id`	Saves paper authors
`paper_comments`	`comment_id`	Saves comments about specific paper
`paper_email_log`	`log_id`	Saves log describing emails associated to a paper
`paper_event_log`	`log_id`	Saves log describing events happened to a paper
`paper_files`	`file_id, revision`	Saves information of files related to a paper
`paper_galleys`	`galley_id`	Saves information of a published paper
`paper_notes`	`note_id`	Saves notes of editor a out specific paper

Table Name	***Primary Key***	***Description***
`paper_search_keyword_list`	`keyword_id`	Saves all keywords appearing in items the system
`paper_supplementary_files`	`supp_id`	Saves information of supplementary files associated to an specific paper
`papers`	`paper_id`	Saves information of each newly submitted paper in the system
`comments`	`comment_id`	Saves reader comments about papers
`assignments`	`assignment_id`	Saves information of assignments

`currencies`	`currency_id`	Saves information of currencies for the subscription
`edit_paper`	`edit_id`	Saves information of editing papers
`decisions`	`decision_id`	Saves decisions for a specific paper
`email_templates`	`email_id`	Saves a list of email templates
`authors`	`author_id`	Saves information about authors
`users`	`user_id`	Saves information of users

Class Reference

Class Hierarchy

All classes and subclasses of the main MS 1.0 objects are given below. Indent shows inheritance; for instance, `EvaluatorAction` inherits from `Action`.

```
DataKeyManager Action
       PaperAction AuthoreditorAction LayoutEditorAction ReaderAction
       EvaluatorAction SubevaluatorAction
              LeaderAction
PaperLog PaperSearch PaperSearchIndex Configuration ConfigurationParser Main
       EntryKeyDO PaperCommentDO PaperDO PaperEmailLogDO PaperEventLogDO
       PaperFileDO PaperGalleyDO PaperNoteDO PaperSearchDO AuthorDAO
       AuthorSubmittedPaperDO CommentDO AssignmentDO
       CopySubmissionDO CountryDO
       CurrencyDO ModifyAssignmentDO EvaluatorSubmissionDO EmailTemplateDO
```

```
IssueDO MagazineDO MagazineSettingsDO
MagazineStatisticsDO LayoutDO LayoutEvaluatorSubmissionDO
NotificationDO
ProofPaperDO ProofreaderDO PublishedPaperDO
EvaluateAssignmentDO ReviewerPaperDO RoleDO SectionDO
SectionEvaluatorSubmissionDO SectionEvaluatorDO SessionDO
SiteDO SubscriptionDO SubscriptionKindDO SuppDO TempFileDO UserDO
VersionDO DORegistry DBConnection
AccessKey Paper
       AuthorPaper CopyEvaluatorSubmission LayoutEvaluatorSubmission
       ProofEvaluatorSubmission PublishedPaper PreEvaluatorSubmission
       SectionEvaluatorSubmission
              EvaluatorSubmission
PaperComment PaperEmailLogEntry PaperEventLogEntry PaperFile
       PaperGalley
              PaperHTMLGalley PaperNote
       SuppFile AuthLead Author
PreEmailTemplate
       AuthorEmailTemplate LanguageEmailTemplate
Comment CopyPaper Currency EditPaper Group GroupMembership HelpTopic
Issue
Magazine FormatAssignment Mail
       MailTemplate
              PaperMailTemplate EvaluationAssignment
CheckAssignment Role
Section Site
Subscription SubscriptionType TempFile User
       ImportedUser
Version FileManager
PaperFileManager FileManager TemporaryFileManager
```

Page Classes

Introduction

Pages classes accept requests from users' browsers, redirect any needed processing to other classes, and call the respective Smarty template to display a response. All page classes reside in the `pages` directory, and each one should extend the `Handler` class.

Also, page classes duty is to ensure user requests are valid and authentication requirements are satisfied. Submitted form parameters and URL parameters should be processed in Page classes and not anywhere else.

Each Page class contains some functions that can be called by the user by addressing the respective Page class and function in the request URL.

The number of tasks a Page handler must perform is significant. For instance, if all requests for Section Author functions were handled directly by the `SectionAuthorHandler` class, it would be significantly large and hard to maintain. Thus, functions are divided into other classes (such as `PaperEditHandler` and `PaperCommentsHandler`), and `SectionAuthorHandler would` only call the specific subclass.

Model Classes

The Model classes and their duty is only to represent database entities in memory. For instance, the `papers` table stores article information in the database; there is a corresponding Model class called `Papers` (see `classes/paper/Paper.inc.php`) and a DO class called `PaperDO`.

Methods of Model classes are get/set methods to fetch and store information, such as the `getTitle()` and `setTitle($title)` methods of the `Paper` class. Model classes duty is not database storage or updates; this is performed by the respective DO class. Model classes extend the `DataObject` class.

Data Objects

Data Objects used to retrieve data from the database in the form of Model classes, to update the database given a modified Model class, or to delete rows from the database.

Each Model class has a respective Data Access Object. For instance, the `Paper` class (`classes/paper/Paper.inc.php`) has an associated DO called `PaperDO` (`classes/paper/PaperDO.inc.php`) its duty is to implement interactions between the Model class and its database data

All DOs extend the `DO` class (`classes/db/DO.inc.php`). All communication between the codes and the database backend is implemented in DO classes.

When DOs are used they are not instantiated directly. But, they are retrieved by name using the `DORegistry` class, which maintains instances of the system's DOs. For example, to retrieve an paper DO:

```
$paperDo = &DORegistry::getDO('PaperDO');
```

Then, to use it to retrieve an article with the ID stored in `$paperId`:

```
$paper = &$paperDo>getPaper($paperId);
```

Configuration

MS 1.0 settings are saved in the database, for example magazine settings in the `magazine_settings` table, and can be accessed via respective DOs and Model classes. Some system-wide settings are saved in a file named `configuration.inc.php`. This configuration file is parsed by the

ConfigurationParser class and saved in an instance of the Configuration class located in the classes/configuration directory.

Main Classes

The Main classes (in the classes/main directory) provide basic functions.

- Main.inc.php: has systemwide functions
- DataObject.inc.php: Model classes extend this class
- Handler.inc.php: Page classes extend this class
- Registry.inc.php: has a systemwide facility for global values, for example system startup time, to be saved and retrieved

The Request class (defined in Request.inc.php in classes/core directory) has some functions to get information about the remote user and create responses. All URLs built by MS to link into itself are built using the Request::url function; similarly, all redirects into MS are formed using the Request::redirect function.

Security

The MS 1.0 security model on the basis of the concept of roles. The system's roles are (e.g. author, reader, editor, etc) and users are assigned to roles. Roles managed via the Role model class and respected RoleDO, which manages the roles database table and allows security checking.

The Validation class is defined in Validation.inc.php located in classes/security directory. Its duty is ensuring security in interactions between the client and web server. It adresse login and logout requests, creates password hashes, and facilitates many useful shortcut functions for security and validation issues.

Session Management

Session management is provided by the Session model class, SessionDO, and the SessionAdminister class (classes/session/SessionAdminister.inc.php).

Session and SessionDO manage database consistent sessions for every user, but SessionAdminister is about the technical aspects of sessions.

Template Support

Smarty templates (http://smarty.php.net) are accessed and managed via the `TemplateManager` class (`classes/template/TemplateManager.inc.php`), which performs numerous common tasks such as registering additional Smarty functions such as `{translate ...}`, which is used for localization, and setting up frequently used template variables such as URLs and date formats.

Database Interaction with DOs

The following code snippet fetches a paper object using the paper ID supplied in the `$paperId` variable, alters the title, updates the database to the new values.

```
// Fetch the paper object using the paper DO.
$papeeDo = &DORegistry::getDO('PaperDO');
$paper = &$paperDo>getPaper($paperId);

$paper->setTitle('This is the new paper title.');

// Update the database with the modified information.
$paperDo>updatePaper($paper);
```

Also the following snippet deletes an article from the database.

```
// Fetch the paper object using the paper DO.
$paperDo = &DORegistry::getDO('PaperDO');
$paper = &$paperDo>getPaper($paperId);

// Delete the paper from the database.
$paperDo>deletePaper($paper);
```

Basically, DOs duty is deleting dependent database entries. For instance, deleting a paper will delete that paper's authors from the database.

User Interface

The User Interface is a large set of Smarty templates, called from the different Page classes.

These templates duty is the HTML representation of each page; but all content comes from template variables (e.g. paper titles). One should be familiar with Smarty templates to work with MS 1.0 templates. Smarty information is available in http://smarty.php.net.

Variables

Template variables initialized in the Page calls the template. Also many variables are initialized by the `TemplateManager` class accessed by all templates:

- `siteTitle`: If the user is currently browsing a page of a magazine, this is the magazine title; in other cases the site title from Site Configuration
- `pagePath`: Path of the requested page
- `currentUrl`: The complete URL of the current page
- `currentMagazine`: The currently applicable magazine object
- `headerTitle`: Used by `templates/common/header.tpl` to display magazines specific information
- `headerLogo`: Used by `templates/common/header.tpl` to display magazines specific information
- `stylesheets`: An array of stylesheets to include with the template
- `pageFooter`: Custom footer to be displayed at the bottom of the page

Obtaining More Information

For more information, see the Neredataltics UG web site at www.neredataltics.org. There is an MS support available at www.neredataltics.org; this is the preferred method of contacting the MS team.

The team can be reached by email at info@neredatatlics.org