Datalus User Manual

Abbey Hawk Sparrow
Bryan Traywick

August 11, 2006

Contents

1 Introduction

1.1 Purpose

Datalus is a Web API which focuses on 'objects' as the source of most of the headache in writing a dynamic website or 'web application'. Much of the work in developing code to display form interfaces to the user and submitting that data to some kind of storage devices, with perhaps a few specialized scripts for extra special voodoo. Datalus removes all the tedium of form creation, submission, validation, and data retrieval. You concentrate on the logic of your web application, or simply use objects and scripts others have created. As a showcase for showing what can be done with Datalus, I've created a set of objects for a blog/news/community site, so with the exception of designing your page graphics/layout, it can just be a simple matter of editing a few properties files (2). It could just as easily be a shopping cart, or a personnel tracking interface or a CRM, or pretty much anything. And as more primitives make thier way into the API, possibilites will only expand.

1.2 History

Datalus began life as a templating engine I wrote in 1999 as my first foray into PHP called 'Catalyst', which used recursive HTML templates with an escape charater sequence and properties files to create a session with permissions and to generate a page, so I could spend my time tooling around with SQL, Forms and the wheel-reinvention normally associated with web development. The idea was I would be able to maintain a common code-base between all the sites I managed. The truth of it, was I maintained slightly different versions for each site, which helped, but didn't exactly solve the problem.

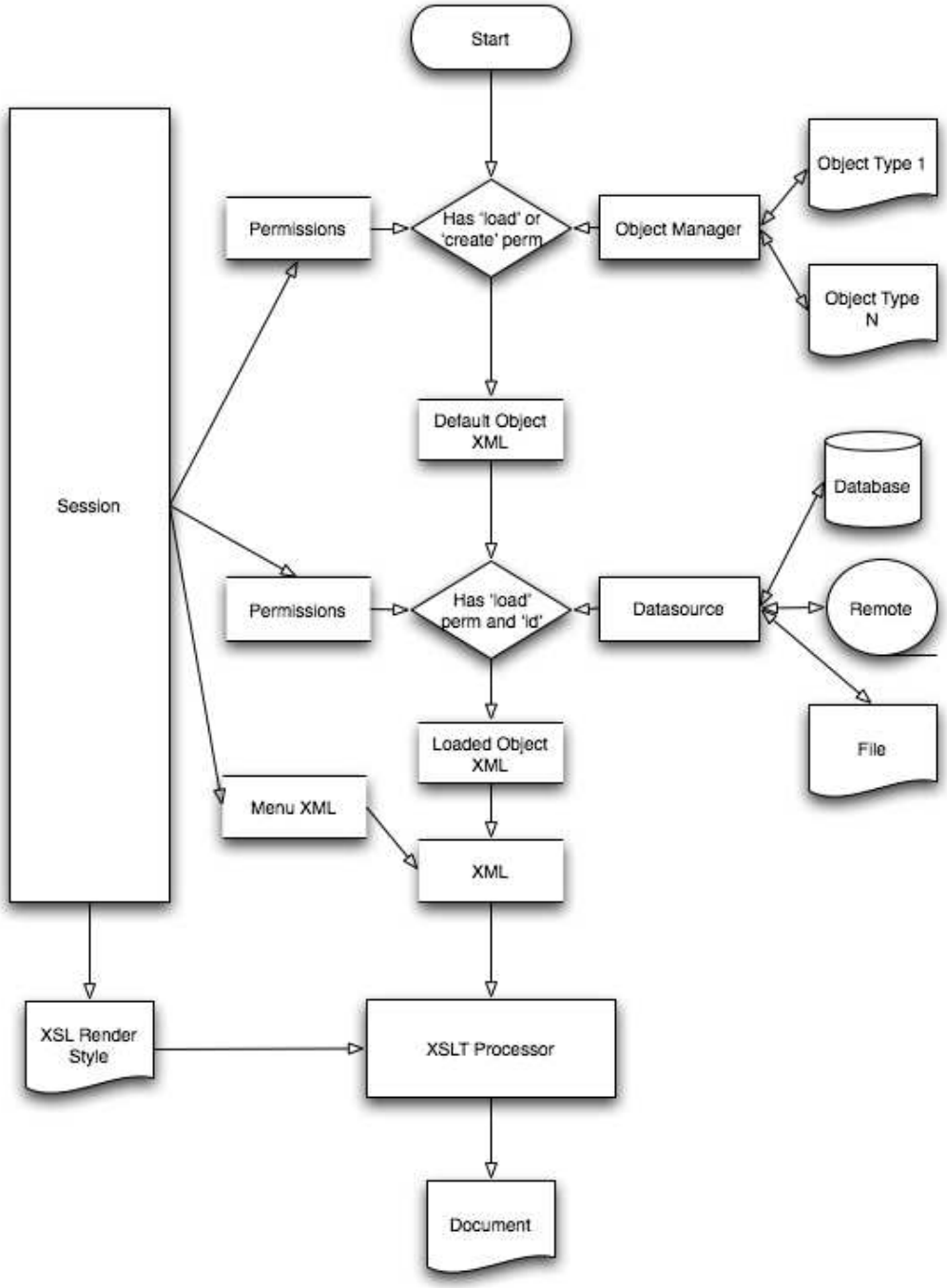
In late 2003 I decided to scrap the old idea in favor of a new design which would hide everything but getting, setting and the initiation of loading and storing. Any other interaction would have to either be automated or prompted to the user. During this time I basically kept a running sheaf of papers refining what I thought would work and doing various proof-of-concepts.

At some point I grew frustrated with my own templating language and discovered XSL, this in combination with ideas that come from a Java library I am also writing gave me my current model.

Datalus completely seperates Composition, Structure and Style. All core logic is performed on the 'Datalus object' making any script accesible through the core XML dialect. The structure is given to the data by passing this XML through an XSL transformation (based on write mode and output format) which outputs a browser format (in it's simplest case, xhtml). This markup is entirely semantic, however, allowing all positioning/orientation/style to be applied by the user in their CSS. This model alows alternative formats (for example, 4.0 generation browsers) to simply perform a different XSL transformation for display.

It does not require rendering trees, which must be kept up-to-date through administrative hoop-jumping. All Datalus' caching happens automatically and uses a temporal window to refresh new copies. Datalus, though it includes imperitive main scripts, is entirely Object Oriented and extensible.

The easiest way to understand how this works is to trace how an object is requested and delivered:



First the incoming type and id come in through the script (in 'type' and 'id', respectively), then an object is requested from the ObjectManager, which initializes a default instance of the object from it's XML description. If an id is present the object loads itself from the datasource defined in the object's XML type description. the object can then ouput itself as XML (using the type dialect). This is merged with any state XML data (persistant page elements, session state and menu elements, for example). The conglomeration of all this XML is the fed through an XSL transformation which corresponds to the target output format (for example: pure xhtml, xhtml + Javascript, xhtml + Ajax).

1.3 Future Direction

Datalus current has a low degree of optimization, while algorithmic complexity should be low, I'm quite sure many statements or functions could use some work in that area. Also, supporting searching and core objects in the file source will allow users to run without a database. Hardening the permissions and user input security is probably the last priority for a candidate I feel good about releasing... though I toy with the idea of adding Safe-Surf tags as well.

One of the main focuses for future development will be expansion: object primitives (3D, Audio, Video...), output formats(wml, etc), verifiers and a versitile set of objects for use 'out-of-the-box'. I'd also like to explore design concepts for internationalization.

1.4 3rd Party Libraries

The following Libraries were used for some of the output targets in Datalus:

- Dynamic Table Sorting: http://friedcellcollective.net/js/SortedTable/
- Rich Text Editor: http://tinymce.moxiecode.com/
- Date Selection Input: http://calendar.moonscript.com/dateinput.cfm (we use a heavily modified version that includes time)
- Location Batch Display: http://maps.google.com
- OpenID : xprofile
- Modal Display: ibox, Ahmed Farook, www.iBegin.com
- browser detection: class written by dragan@dinke.net (slightly modified)

My thanks to anyone who contributed to these libraries.

2 Configuration

2.1 Style

All layout should be done in style.css
Here we will later list id's and classes for CSS

2.2 Datasources

Datalus defines datasources in a generic way, it is a place where objects get mapped to and from some external storage system. Currently there is a MYSQL source (which uses mostly generic SQL) and a File Source. Inside Configuration/Datasources/ you describe them as ¡name¿.properties using the following parameters:
mode: the mode this datasource is (currently 'sql' or 'file')
name: the name of the database
host: the host URL for the database
user: database username
pass: database password

2.3 Properties

The main properties file centralizes most settings needed for the site. Hopefully this stuff is somewhat self-explainatory...
style : the default style delivered to any browser that doesn't fit one of the "special" definitions
nonstandard_IE_style : This is a css definition for IE versions on Windows that use various nonstandard style rules (borders and such).
mac_IE_style : This is a css definition for IE versions on Macintosh, which in addition to the windows peculiarities, has a few of it's own.
NS4_style : This is the style information for NS4 generation browsers, going light on sizing and positioning, sticking mainly to color, font and such.
transform_path : This is the directory which contains the XSL files catalyst uses to transform the core XML dialect to a target format.
wml_transform : This is the identifier used for delivery to mobile devices in the WML format.
mosaic_transform : This is Raw, unstyled HTML.
NS4_transform : An Identifier for late 90s era browsers, typified by Netscape 4.
ajax_transform : A target which enables Session management using AJAX.
javascript_transform : An output target which uses rich JS editing widgets for JS enabled browsers.
transform : This target uses semantic tags, relying on it's CSS for sizing and positioning
default_transform : This indicates which transformation Datalus should prefer
datasources : This folder indicates where Datalus should load Datasource configurations.
default_datasource : This is identifier of the database used as the default. Core data is stored here as well as any objects with no explicit datasource.
welcome_mail_body : This is the location of the activation and welcome mail sent to new users(More information on this format).
content_owner : The name of the site owner
domain : The root URL of the site
site_origin : The initial copyright year of the site's content.
information_contact, registration_contact, technical_contact : contact email addresses.
copyright_notice : this is the relative location of the HTML snippet used for the copyright notice.
privacy_policy : this is the relative location of the HTML snippet used for the privacy policy.
HTML_pages_directory : this is the directory Datalus looks in for HTML snippets
public_HTML : this is a comma seperated list of HTML snippet labels (filename ¡label¿.html) ***custom_code*** : this is a comma seperated list of user PHP scripts which are executed on each page
404_config_string : this is the string used to configure the 404 script.

cache_mode : cache control variable
clean_output_folder : clean files in the temp directory?
default_order : default ordering for batches
google_maps_key : the google maps key for the URL the site is hosted on
mcrypt_key : Your AES Encryption Key.
password_encoding : which codec the password is encoded in.

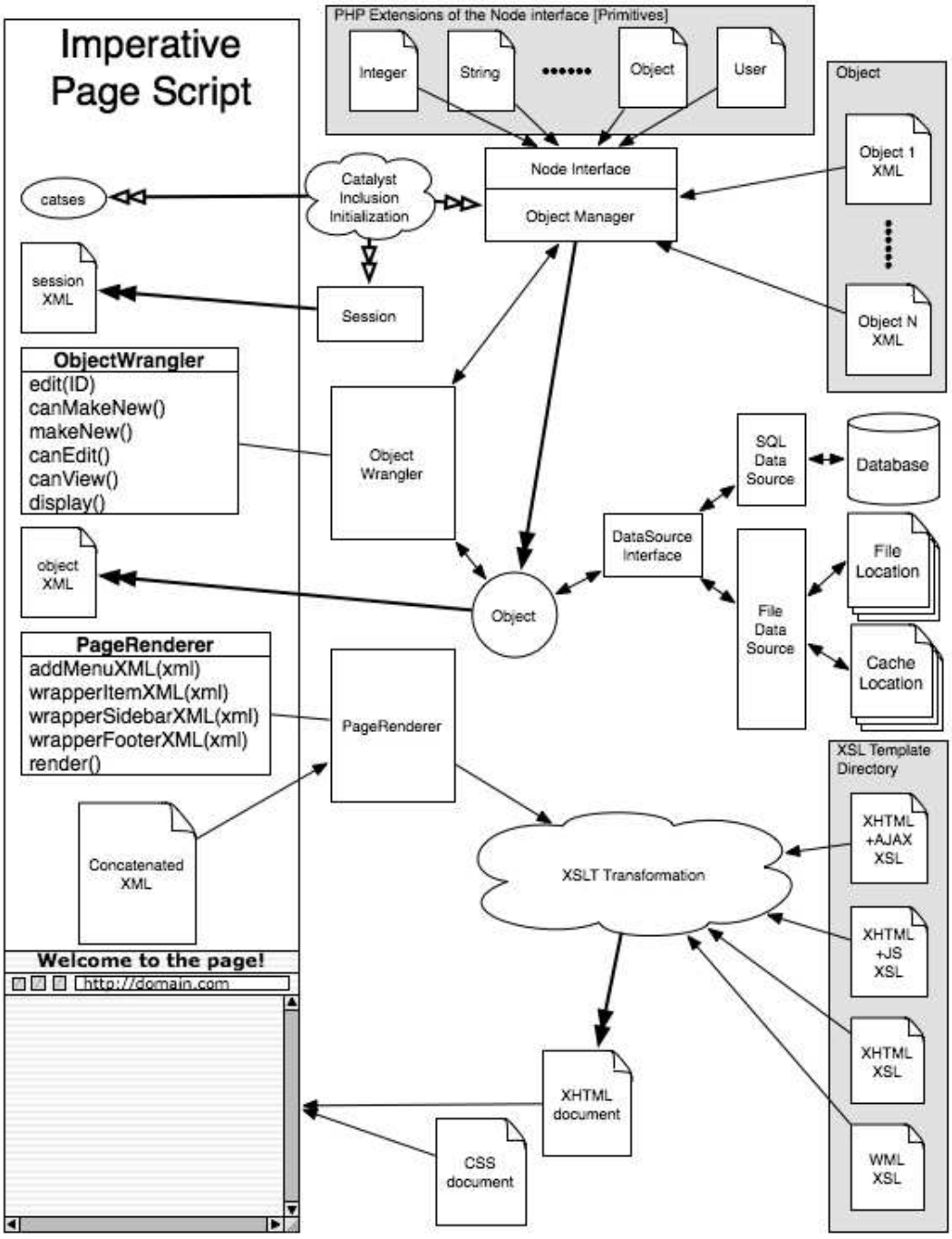
3 Objects

3.1 Overview

Objects are loaded by the 'Object Handler' so that when needed, new instances can be cloned off of the generic, handling of searches and batch retrieval are handled by the 'BatchWrangler' while the 'ObjectWrangler' fetches and manipulates individual objects. The very most simple use of this would be a 'news' blog as follows (fetches most recent entries up to the default threshold, in the order defined in 'datalus.properties'):

```
<?php
    require("Catalyst.php");

    $renderer = new PageRenderer();
    $wrangler = new BatchWrangler('news', getCatProp('default_order'));
    $xml = $wrangler->display();
    $renderer->addMenuXML($catses->makeCatalystMenu()->XML(""));
    $renderer->addMenuXML($catses->makeMenu("Navigation")->XML(""));
    $renderer->wrapperStaticXML($catses->getGreeting());
    $renderer->wrapperFooterXML($renderer->renderCopyright());
    $renderer->wrapperJavascriptXML($catses->getVerifierJS());
    $renderer->wrapperItemXML($xml);
    echo($renderer->render());
?>
```



3.2 Syntax

3.2.1 Global Properties

The following attributes are common to objects and primitives:

- name: The item's name
- label: The item's displayed name

The remainder are attributes for primitives only:

- obsfucate: Make this field hidden to the user

3.2.2 The Object Tag

- datasource: textual handle of a registered datasource
- createPermission: permission to create new instances
- editPermission: permission to edit existing entries
- deletePermission: permission to remove existing entries
- viewPermission: permission to view existing entries
- ownerAccesible: what can the owning user do? (read/write/none)
- groupAccesible: what can the owning group do? (read/write/none)
- axis: what field are groups categorized using by default? (note that instances will produce a calendar, location: a google map... provided you supply a key, and image will produce thumbnails)
- identifier: what field is used to summarize the object?

3.2.3 Primitives

block: a large chunk of text (multi-line)

- limit: The maximum number of characters

file: an unrecognized binary file
float: a floating point number
group: a linked group of objects of another type
hidden: a value to be passed back without edit(best to keep clear of this)
image: an image file, may be resized, may be copied from another source in this object (to create a thumbnail, for example)

- height: The maximum height (will scale in existing ratio)
- width: The maximum width (will scale in existing ratio)
- copy-from: duplicates the image from the primitve, who's name is provided

instant: a point in time denoted by MM-DD-YYYY HH:MM:SS, defaults to current if unset
integer: a whole number
location: a physical address
reference: a reference to an instance of the object type given

- type: the type of object this is a reference to
- label-key: which primitive is the label
- value-key: which primitive contains the value

string: a single line string (currently limited to 128 chars)
user: a local user accoutn
usergroup: a local user group

3.3 Examples

```
<object name="artist" axis="picturethumb" identifier="published_name" datasource="database"
 createPermission="administrator"  editPermission="administrator" viewPermission="none"
  deletePermission="administrator" ownerAccessible="write" groupAccessible="read">
    <string name="published_name">John Doe</string>
    <string name="last_name">Doe</string>
    <string name="first_name">John</string>
    <string name="email">john@johndoe.com</string>
    <string name="homepage">http://johndoe.com</string>
    <string name="phone_number">912-555-5555</string>
    <block name="bio" limit="1500">Text</block>
    <image name="picture" height="300" width="300"></image>
    <image name="picturethumb" copy-from="picture" height="100" width="100"></image>
    <image name="signature"></image>
</object>
```

4 Reference

4.1 Package Structure(A current snapshot)

4.1.1 BatchWrangler.php
20 functions/classes, 307 lines, 11313 bytes

- 1: class BatchWrangler;
- 16: function *BatchWrangler* (type, order);
- 27: function *addDiscriminator* (operand1,operator, operand2);
- 33: function *limitBatchSize* (size);
- 37: function *setStart* (position);
- 41: function *setToSummary* ();
- 46: function *setAxis* (axis);
- 50: function *createFooter* (currentPage);
- 136: function *getURLForPage* (page);
- 156: function *getTotalNumber* ();
- 166: function *assemblePredicate* ();
- 186: function *wrangle* ();
- 194: function *displayPage* (page);
- 235: function *canEdit* ();
- 242: function *canView* ();
- 194: function *display* ();
- 255: function *renderSummaryNames* ();
- 266: function *selectSummaryFields* ();
- 282: function *getTotalNumberWithoutRange* ();
- 290: function *selectAllFields* ();

4.1.2 Encoding.php
5 functions/classes, 25 lines, 369 bytes

- 2: class Encoding;
- 5: function *Encoding* (name);
- 9: function *getName* ();
- 13: function *encode* (value);
- 18: function *decode* (value);

4.1.3 MCryptEncoding.php
5 functions/classes, 35 lines, 866 bytes

- 4: class MCryptEncoding extends Encoding;
- 8: function *MCryptEncoding* ();
- 15: function *initKey* ();
- 22: function *encode* (value);
- 27: function *decode* (value);

4.1.4 MD5Encoding.php
3 functions/classes, 17 lines, 233 bytes

- 4: class MD5Encoding extends Encoding;
- 6: function *MD5Encoding* ();
- 10: function *encode* (value);

4.1.5 SHA1Encoding.php
3 functions/classes, 17 lines, 239 bytes

- 4: class SHA1Encoding extends Encoding;
- 6: function *SHA1Encoding* ();
- 10: function *encode* (value);

4.1.6 FileDataSource.php
17 functions/classes, 410 lines, 16975 bytes

- 1: class FileDataSource;
- 7: function *FileDataSource* (path, acting_as_cache=false);
- 12: function *linkObjects* (source, destination);
- 39: function *getUniqueKeysFor* (dataName, objectName);
- 67: function *getUniqueKeysForUser* (dataName, objectName, idCol, userID);
- 116: function *exists* (objectName);
- 121: function *load* (object);
- 193: function *store* (object);
- 236: function *getMaxID* (objectName);
- 252: function *saveUploadedFile* (fieldName);
- 256: function *initializeDataSource* (object);
- 287: function *renderDatasourceFields* (object);
- 330: function *getSummaryData* (IDs, objectType, editable, showHeaders);
- 353: function *getSummaryHeaders* (objectType);
- 364: function *isSimpleField* (type);
- 382: function *getSQLTypeFromType* (type);
- 402: function *wrapperValue* (value, type);

4.1.7 MenuItem.php
14 functions/classes, 121 lines, 3550 bytes

- 1: class MenuItem;
- 9: function *MenuItem* (name, location);
- 14: function *setID* (val);
- 18: function *getParent* ();
- 22: function *setParent* (parent);
- 26: function *setLink* (link);
- 30: function *getName* ();
- 34: function *getChild* (name);
- 38: function *getLink* ();
- 42: function *setMenuType* (type);
- 51: function *XML* (indent);
- 69: function *HTML* (indent);
- 89: function *addChild* (child);
- 94: function *buildItem* (item, link, id = 0);

4.1.8 Node.php
22 functions/classes, 223 lines, 7666 bytes

- 2: class Node;
- 15: function *Node* (name);
- 20: function *getName* ();
- 24: function *getType* ();
- 28: function *getNumKids* ();
- 32: function *hasKids* ();
- 37: function *getNumAttributes* ();
- 41: function *addChild* (child);
- 47: function *getChild* (name);
- 51: function *setParent* (node);
- 55: function *getParent* ();
- 59: function *getValue* ();
- 63: function *setValue* (newval, refObject = null);
- 106: function *getAttribute* (index);
- 110: function *setAttribute* (id,value);
- 120: function *setAttributes* (array);

- 131: function *getChildrenNames* ();
- 140: function *setChildValue* (childName , newValue);
- 154: function *summary* (indent);
- 164: function *setError* (error);
- 168: function *XML* (indent);
- 184: function *replicate* ();

4.1.9 Block.php
2 functions/classes, 10 lines, 105 bytes

- 2: class Block extends Node;
- 4: function *Block* ();

4.1.10 Encrypted.php
7 functions/classes, 35 lines, 883 bytes

- 2: class Encrypted extends Node;
- 6: function *Encrypted* ();
- 12: function *decrypt* ();
- 17: function *getValue* ();
- 21: function *getEncrypted* ();
- 25: function *setEncrypted* (value);
- 29: function *setValue* (newValue);

4.1.11 File.php
7 functions/classes, 48 lines, 1325 bytes

- 2: class File extends Node;
- 5: function *File* ();
- 10: function *getFilePath* ();
- 14: function *getNextFileName* ();
- 18: function *saveFile* ();
- 31: function *getTypeFromFileName* (name);
- 41: function *XML* (indent);

4.1.12 Float.php
2 functions/classes, 9 lines, 104 bytes

- 2: class Float extends Node;
- 4: function *Float* ();

4.1.13 Group.php
5 functions/classes, 47 lines, 1446 bytes

- 2: class Group extends Node;
- 4: function *Group* ();
- 8: function *addNew* ();
- 17: function *linkExisting* (id);
- 27: function *XML* (indent);

4.1.14 Hidden.php
2 functions/classes, 9 lines, 107 bytes

- 2: class Hidden extends Node;
- 4: function *Hidden* ();

4.1.15 Image.php
9 functions/classes, 143 lines, 5731 bytes

- 2: class Image extends Node;
- 5: function *Image* ();
- 10: function *getFilePath* ();
- 14: function *getNextFileName* ();
- 18: function *saveImageAs* (image, saveName, heightMax = 0, widthMax = 0);
- 28: function *saveFile* ();
- 99: function *getTypeFromFileName* (name);
- 109: function *makeResizedJPEG* (name, imagePath, maximumX, maximumY);
- 136: function *XML* (indent);

4.1.16 Instant.php
4 functions/classes, 22 lines, 400 bytes

- 2: class Instant extends Node;
- 4: function *Instant* ();
- 8: function *setValue* (newval);
- 17: function *currentDate* ();

4.1.17 Integer.php
2 functions/classes, 9 lines, 110 bytes

- 2: class Integer extends Node;
- 4: function *Integer* ();

4.1.18 Location.php
2 functions/classes, 10 lines, 118 bytes

- 2: class Location extends Node;
- 4: function *Location* ();

4.1.19 Object.php
16 functions/classes, 113 lines, 3529 bytes

- 2: class Object extends Node;
- 7: function *Object* (datasource);
- 12: function *setSummary* (value);
- 16: function *setID* (value);
- 20: function *addToGroup* (name);
- 25: function *linkToGroup* (name, id);
- 30: function *getGroupType* (name);
- 35: function *getID* ();
- 39: function *load* ();
- 44: function *save* ();
- 48: function *canEdit* ();
- 58: function *canView* ();
- 65: function *canDelete* ();
- 72: function *canCache* ();
- 77: function *setEditable* ();
- 81: function *XML* (indent);

4.1.20 Reference.php
3 functions/classes, 29 lines, 867 bytes

- 2: class Reference extends Node;
- 4: function *Reference* ();
- 8: function *XML* (indent);

4.1.21 String.php
2 functions/classes, 9 lines, 107 bytes

- 2: class String extends Node;
- 4: function *String* ();

4.1.22 User.php
3 functions/classes, 33 lines, 976 bytes

- 2: class User extends Node;
- 4: function *User* ();
- 8: function *XML* (indent);

4.1.23 UserGroup.php
3 functions/classes, 34 lines, 1035 bytes

- 2: class UserGroup extends Node;
- 4: function *UserGroup* ();
- 8: function *XML* (indent);

4.1.24 ObjectManager.php
17 functions/classes, 217 lines, 8479 bytes

- 4: class ObjectManager;
- 12: function *ObjectManager* ();
- 29: function *printRegistered* ();
- 42: function *checkLoad* (type);
- 48: function *getCreationPermissionForObject* (objectType);
- 55: function *getEditorPermissionForObject* (objectType);
- 62: function *getViewPermissionForObject* (objectType);
- 68: function *getDeletePermissionForObject* (objectType);
- 75: function *makeObject* (type);
- 96: function *loadTypes* ();
- 114: function *getObjectNames* ();
- 122: function *getDataSourceForType* (type);
- 133: function *getDefaultDataSource* ();
- 137: function *registerDataSourcesInFolder* (folderPath);
- 159: function *isCached* (type, id);
- 168: function *makeDatasourceFromProperties* (path, name);
- 192: function *readObjectProfile* (objectName);

4.1.25 ObjectWrangler.php
12 functions/classes, 186 lines, 6729 bytes

- 2: class ObjectWrangler;
- 11: function *ObjectWrangler* (type);
- 60: function *grabIncomingFormVariables* (children);
- 119: function *makeNew* ();
- 129: function *canMakeNew* ();
- 137: function *edit* (id);
- 151: function *display* ();
- 158: function *isEditing* ();
- 162: function *setEditable* (editing);

- 166: function *canEdit* ();
- 173: function *canView* ();
- 180: function *outputError* (text);

4.1.26 PageRenderer.php
18 functions/classes, 196 lines, 6477 bytes

- 2: class PageRenderer;
- 17: function *PageRenderer* (xsl = "display_ajax.xsl");
- 25: function *getCSS* ();
- 45: function *getXSL* ();
- 67: function *setEditable* (value);
- 73: function *addMenuXML* (text);
- 77: function *wrapperItemXML* (text);
- 81: function *wrapperAlertXML* (text);
- 85: function *wrapperStaticXML* (text);
- 89: function *wrapperStaticContentXML* (text);
- 93: function *wrapperFooterXML* (text);
- 97: function *wrapperJavascriptXML* (text);
- 101: function *getPageXML* ();
- 117: function *getItemText* ();
- 124: function *renderCopyright* ();
- 130: function *transform* ();
- 136: function *transformXML* (xml, xmlName);
- 124: function *render* ();

4.1.27 Presence.php
21 functions/classes, 177 lines, 5721 bytes

- 1: class Presence;
- 15: function *Presence* (session);
- 33: function *getLegalReferencesXML* (label, table, valueColumn, labelColumn, value);
- 37: function *getSession* ();
- 41: function *initializeCoreTables* ();
- 45: function *login* (username, password);
- 73: function *logout* ();
- 79: function *isLoggedIn* ();
- 83: function *update* ();

- 87: function *reset* ();
- 91: function *setSessionValues* (id, handle);
- 97: function *isInactive* ();
- 104: function *isOutdated* ();
- 111: function *loadPermissions* ();
- 118: function *loadGroupPermissions* (groupid);
- 121: function *hasPermission* (name);
- 136: function *getPermissionList* ();
- 143: function *renderPermissionListAsXML* ();
- 151: function *getSetDate* ();
- 155: function *modifyDate* (month, monthAdd, day, dayAdd, year, yearAdd);
- 163: function *modifyFullDate* (year, yearAdd, month, monthAdd, day, dayAdd, hour, hourAdd, min, minAdd, sec, secAdd);

4.1.28 PropertiesFile.php
8 functions/classes, 67 lines, 1955 bytes

- 2: class PropertiesFile;
- 6: function *PropertiesFile* (fileName, platform);
- 32: function *getProperty* (name);
- 37: function *setProperty* (name, value);
- 41: function *write* ();
- 44: function *getData* ();
- 48: function *printDirectory* (name);
- 57: function *printKeys* ();

4.1.29 RssRenderer.php
8 functions/classes, 144 lines, 5849 bytes

- 2: class RssRenderer;
- 16: function *RssRenderer* (type);
- 33: function *__destruct* ();
- 37: function *render* ();
- 53: function *rssStartElement* (parser, name, attrib);
- 76: function *rssEndElement* (parser, name);
- 96: function *rssCharacterData* (parser, data);
- 109: function *mysql2date* (dateformatstring, mysqlstring, translate = true);

4.1.30 Session.php
20 functions/classes, 316 lines, 11309 bytes

- 2: class Session;
- 9: function *Session* (session, database, name, password);
- 40: function *canDisplay* (pageName);
- 48: function *isAdmin* ();
- 52: function *isEditing* ();
- 56: function *setEditing* (editing);
- 62: function *makeMenuItem* (menu, menuName, menu, target);
- 62: function *makeMenu* (menuName);
- 81: function *makeCatalystMenu* ();
- 143: function *getMenus* ();
- 150: function *getLegalReferencesXML* (label, table, valueColumn, labelColumn, value);
- 214: function *hasPermission* (name);
- 255: function *getUserNameForID* (id);
- 280: function *login* (name, password);
- 285: function *logout* ();
- 289: function *loggedIn* ();
- 293: function *userName* ();
- 297: function *userID* ();
- 301: function *getGreeting* ();
- 306: function *getVerifierJS* ();

4.1.31 SQLDataSource.php
40 functions/classes, 594 lines, 27020 bytes

- 1: class SQLDataSource;
- 10: function *SQLDataSource* (database, user, password, hostname);
- 29: function *getLegalReferencesXML* (label, table, valueColumn, labelColumn, value);
- 44: function *query* (query);
- 52: function *linkObjects* (source, destination);
- 56: function *getUniqueKeysFor* (dataName, objectName);
- 72: function *getUniqueKeysForUser* (dataName, objectName, idCol, userID);
- 93: function *exists* (objectName);
- 102: function *loadSQL* (object);
- 102: function *load* (object);

- 147: function *store* (object);
- 154: function *storeSQL* (object);
- 226: function *saveUploadedFile* (fieldName);
- 230: function *coreObjects* (mode, table);
- 237: function *getSession* (presence);
- 249: function *checkLogin* (name, password);
- 259: function *newSession* (userID);
- 269: function *purgeSessions* (user);
- 276: function *resetSession* (session);
- 282: function *updateSession* (session);
- 286: function *getSessionYoungerThan* (session, years, months, days);
- 292: function *getUserName* (id);
- 299: function *addNewGroupWithOwnerAsMember* (vars);
- 309: function *saveNewUserGroup* (vars);
- 315: function *getGroup* ();
- 337: function *getGroupList* ();
- 353: function *getRecentUsers* (numMins);
- 361: function *initializeDataSource* (object);
- 396: function *modifyFullDate* (year, yearAdd, month, monthAdd, day, dayAdd, hour, hourAdd, min, minAdd, sec, secAdd);
- 407: function *modifyDate* (month, monthAdd, day, dayAdd, year, yearAdd);
- 415: function *initDatasource* ();
- 432: function *renderDatasourceFields* (object);
- 475: function *renderComments* (object);
- 488: function *renderRating* (object);
- 503: function *getSummaryData* (IDs, objectType, editable, showHeaders);
- 526: function *getSummaryHeaders* (objectType);
- 537: function *isSimpleField* (type);
- 555: function *getSQLTypeFromType* (type);
- 577: function *wrapperValue* (value, type);
- 584: function *cache* (object);

4.1.32 Uploader.php
13 functions/classes, 123 lines, 4438 bytes

- 2: class Uploader;
- 11: function *Uploader* (file);
- 11: function *upload* (path, name, destinationName);
- 46: function *saveAs* (filename, directory, field, overwrite,mode=0777);
- 77: function *getType* ();
- 88: function *getName* ();
- 92: function *getMIME* ();
- 96: function *getSize* ();
- 100: function *getError* ();
- 104: function *getFileError* (field);
- 108: function *getFileName* (field);
- 112: function *getFileMimeType* (field);
- 116: function *getFileSize* (field);

4.1.33 Verifier.php
6 functions/classes, 26 lines, 381 bytes

- 2: class Verifier;
- 5: function *Verifier* (name);
- 9: function *getName* ();
- 13: function *verify* (string);
- 16: function *getJavascriptFile* ();
- 20: function *getAjaxUrl* ();

4.1.34 EmailVerifier.php
3 functions/classes, 23 lines, 520 bytes

- 4: class EmailVerifier extends Verifier;
- 6: function *EmailVerifier* ();
- 10: function *verify* (string);

4.1.35 FileVerifier.php
3 functions/classes, 22 lines, 402 bytes

- 4: class FileVerifier extends Verifier;
- 6: function *FileVerifier* ();
- 10: function *verify* (string);

4.1.36 HtmlVerifier.php
6 functions/classes, 59 lines, 1396 bytes

- 4: class HtmlVerifier extends Verifier;
- 9: function *HtmlVerifier* ();
- 16: function *__destruct* ();
- 20: function *verify* (string);
- 37: function *htmlVerifierStartElement* (parser, name, attrib);
- 44: function *htmlVerifierEndElement* (parser, name);

4.1.37 TextVerifier.php
3 functions/classes, 22 lines, 362 bytes

- 4: class TextVerifier extends Verifier;
- 6: function *TextVerifier* ();
- 10: function *verify* (string);

4.1.38 UrlVerifier.php
3 functions/classes, 22 lines, 404 bytes

- 4: class UrlVerifier extends Verifier;
- 6: function *UrlVerifier* ();
- 10: function *verify* (string);

4.1.39 XHtmlVerifier.php
6 functions/classes, 63 lines, 1507 bytes

- 4: class XHtmlVerifier extends Verifier;
- 9: function *XHtmlVerifier* ();
- 16: function *__destruct* ();
- 20: function *verify* (string);
- 37: function *xhtmlVerifierStartElement* (parser, name, attrib);
- 47: function *xhtmlVerifierEndElement* (parser, name);