starfish project documentation

# Table of Contents

* **Objects**
  + **Core**
  + **Private**
  + **Public**
  + **Ajax**
  + **Libraries**
* **API / Services**
  + **Naming Conventions**
  + **Services**
* **Object Relation Model**
  + **Inheritance Model**

# Objects

## Core

Core objects are the base of the starfish ORM. They operate as fundamental operations upon which other objects are developed. This allows child objects to easily inherit all the functionality they need or simply spawn a new instance of a needed object for the duration of the decedents needs. Core functionality should see the least amount of API change through the duration of the project.

* **Core Objects**
  + **obj.core.php**
    - Classes: stCore
    - Sets Error Reporting.
    - Sets Timezone.
    - Place Holder for core functionality.
  + **obj.debug.php**
    - Classes: debug
    - Used to debug variables / states of the application
    - Toggled via config.
  + **obj.exception.php**
    - Classes: stException, stFatalError, stWarningError, stPermissionError, stBasicError.
    - Application error handling.
    - Auto loaded by core
  + **obj.filesystem.php**
    - Classes: stFilepath, stWebpath
    - Software file pathing
    - Auto loaded by core
  + **obj.global.php**
    - Classes: stGlobal
    - global post, get, file data
    - Auto loaded by core
  + **obj.includer.php**
    - Classes: stIncluder
    - Pathing and details for commonly included files/object etc.
    - Auto loaded by core
  + **obj.mysql.php**
    - Classes: stMysql
    - mySQL abstraction layer.
    - Auto loaded by core.
    - Configured by config.
    - Returns objects as results or arrays.
  + **obj.proxy.php**
    - Classes: stProxy
    - Uses curl to access web content and return results.
    - Transparent Request results are transferred and buffered.
  + **obj.security.php**
    - Classes: stSecure
    - Congifures InputFilter library object for inheritance.
  + **obj.session.php**
    - Classes: stSessionData, stSession, stVisitorSession, stMemberSession, stAdminSession
    - Holds session data.
    - Authenticates users.
    - Maintains sessions.
    - Revalidates user credentials.
  + **obj.smartfields.php**
    - Classes: stSmartFields
    - Smart data input fields for forms etc.
    - Maintains state across page states.
  + **obj.template.php**
    - Classes: stTemplate
    - Templating Engine.
    - Allows object to template segment matching.
    - Allows associative array to template matching.
  + **obj.validate.php**
    - Classes: stValidate
    - Validator Object
  + **obj.xml.php**
    - Classes: stXML
    - XML interpreter / parser.
    - simpleXML controller.

## Private

Private objects are protected objects that can only be executed by authorized users. This is generally the backend administration of a website or the administrative functions of an application. These are objects that will never be executed or included within the public realm.

By default there are no private classes or objects at this time.

## Public

Public objects are objects that are used in the public domain of a project. These objects are generally used to read data and occasionally write data or contact a service. They generally have no functionality that can be harmful to the application. This allows for safer code separation to happen between the public and private domains of the project.

* **Public Objects**
  + **obj.gdlib.php**
    - Classes: stGDlib
    - Public Image Handler and GD functionality.
  + **obj.logger.php**
    - Classes: stLogData, stLogger
    - Log Data and Log Handler
    - Allows quick and easy application logging.
    - Allows searching of logs for checking user actions.

## Ajax

The AJAX portion of the application is a collection of objects, controllers, dispatchers, widgets, and widget generators. The focus of this portion of the application is to easily apply functionality and AJAX compatibility within the starfish framework. The controllers and dispatchers allow for calls back into the Starfish system. The objects and generators allow control and generation of common AJAX UI functionality such as dialogs, modals, and enhanced input fields etc.

* **Public Objects**
  + **Controllers**
    - **obj.dialogs.php**
      * Classes: stDialogGenerator
      * Used for generating dialogs with content on the fly.
    - **obj.windows.php**
      * Classes: stWindowGenerator
      * Used for generating modal windows with content on the fly.
  + **obj.ajax.php**
    - Classes: stAjax
    - AJAX dispatcher and controller for AJAX objects.
  + **obj.macro.php**
    - Classes: stAjaxMacro
    - Handles stored macro procedures for AJAX services.

## Libraries

Starfish uses a number of open source and freely available libraries to extend its functionality. These libraries are stored so that they are easily updated. No code stored within a library is ever altered by the Starfish developers. These libraries are meant to be maintained unaltered so that they are easily patched and updated in the future by their developers.

* **Libraries**
  + **inputfilter**
    - PHP Library Object
    - Used for data sanitization.
  + **phpThumb**
    - Used for on demand thumbnails and thumbnail caching.
  + **YUI (Yahoo User Interface)**
    - Used for JavaScript standardization.
    - Used for base widgets of the Starfish framework.
  + **HTML Mime Mail 5**
    - Used as the sendmail / smtp layer of the framework.

# API / Services

## Naming Conventions

The Starfish framework follows a simple naming convention that prepends the type of code contained within the file followed by the purpose of the code followed by the file extension for this file type.

For instance – an object that controls the display of a news article would be named obj.news.php and a collection of objects that describe a database would be named col.database.php for instance.

* **Naming Conventions**
  + Objects: obj.{type}.php
  + Collections: col.{type}.php
  + Javascript Widget: widget.{type}.php
  + Logic: logic.{type}.php
  + Macros: macro.{type}.php

You will also notice that almost all code developed within Starfish uses camel casing and standard PHP5 markup for statements. Objects within the framework also follow a naming convention that appends “st” to the objects purpose. For instance an xml data object would be named stXMLData and a news object would be named stPublicNews etc.