Keruald start process

Application entry point walkthrough

This white paper is a walkthrough of application entry point: what happens when you load the index.php of a PHP application built on the top of Xen or Pluton?

2010

Sébastien Santoro aka Dereckson

Trantorium

2/28/2010

*To Jérôme F.,  
my first Pluton user.*



# Table of contents

[Table of contents 3](#_Toc255159067)

[Introduction 4](#_Toc255159068)

[Standard index. php operations 4](#_Toc255159069)

[Load libraries 4](#_Toc255159070)

[Session initialization 5](#_Toc255159071)

[Configuration settings 5](#_Toc255159072)

[Handle login or logout actions 6](#_Toc255159073)

[Login 6](#_Toc255159074)

[Logout 6](#_Toc255159075)

[Get logged in user information 7](#_Toc255159076)

[How to determine if the user is logged in? 7](#_Toc255159077)

[It’s now up to you. 8](#_Toc255159078)

[Some tips 8](#_Toc255159079)

[Pluton’s index.php extra operations 9](#_Toc255159080)

[Xen’s index.php extra operations 10](#_Toc255159081)

[Index 11](#_Toc255159082)

# Introduction

*This white paper is a walkthrough of application entry point: what happens when you load the index.php of a PHP application built on the top of Xen or Pluton?*

# Standard index. php operations

## Load libraries

The first operation is to load the Keruald libraries. Those libraries are commonly placed in includes folder.

*The following variables will be available at this stage:*

**$db** database connection

**$Config** configuration array

include('includes/core.php');

During this operation, some components will be initialized, like the database connection.

To control what libraries to load, edit the top of includes/core.php

The common scenario is to load:

1. config.php the site configuration file
2. error.php the error management functions (e.g. the message\_die method)
3. mysql.php the database layer (it’s here you can load other sql\_db classes)
4. session.php the session handler

Then, you can add components always mandatory in your application.

If a component requires database access, load it \*AFTER\* the database file.

Pluton loads document.php, the class determining the document to load and getting metadata information.

If you’ve miscellaneous utilities functions, you can append them to core.php in a starter kit scenario or in a separate file to simplify update process.

## Session initialization

Two sessions levels coexist, the database and the PHP one:

* at database level, the table sessions keeps track of online/offline status, time of the last session update and user logged in ;
* at PHP level, you’re free to use PHP $\_SESSION array to store and load session settings.

The session is initialized in the Session object.

$Session = Session::load();

This load method creates the unique Session object instance of this singleton class[[1]](#footnote-1), performing the following actions:

1. Sets the current PHP session ID to $\_SESSION[‘ID’]
2. Sets the following Session instance properties:
   1. **id** current PHP session ID
   2. **ip** remote client IP
3. Cleans up sessions in database:
   1. deletes expired sessions
   2. sets offline inactive sessions
4. Saves or updates current session in database

### Configuration settings

You can customize in includes/config.php the expiration and offline sessions time and resource identifier:

|  |  |  |
| --- | --- | --- |
| Parameter | Description | Default value |
| $Config['OnlineDuration'] | The duration in seconds a user is still considered online after the last hit. | 300[[2]](#footnote-2) |
| $Config['SessionDuration'] | The duration in seconds the session is kept alive after the last hit. | 7 200[[3]](#footnote-3) |
| $Config[‘ResourceID’] | Site identifier | 1[[4]](#footnote-4) |

## Handle login or logout actions

Once the libraries are loaded and the session initialized, it’s time to get current user information. But before, we need to login or logout user.

*OpenID*

If you want to enable OpenID on your Pluton/Xen/Keruald website, there is an OpenID-ready login.php available, which consumes the PHP OpenID library by JanRain.

A short generic file[[5]](#footnote-5) is provided to do those tasks:

include("includes/login.php");

### Login

#### Condition

POST request must contain a **LogIn** value.

Values **username** and **password** are expected too.

#### Actions

Checks if there is in users table a record matching the specified user.

If so, it compares the passwords[[6]](#footnote-6).

##### If all is correct

1. Session user\_login method is called with user logged in id in argument.  
   That will set the user ID in the sessions table user\_id field.
2. Global variable $LoginSuccessful is set to true;  
   This variable allows you to execute code on login.

##### If there is an error

Global variable $LoginError will contain error message, ready to be printed to user.

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Proin pharetra nonummy pede. Mauris et orci.

### Logout

#### Condition

Whether a POST request with a LogOut parameter,  
whether a GET request with an **action** parameter having *user.logout* as value.

#### Actions

Session user\_logout method is called.

That destroys any of your $\_SESSION table values, help the ID who is preserved.

## Get logged in user information

To get a current user object, the session instance’s get\_logged\_user method is called:

$CurrentUser = $Session->get\_logged\_user();

Or, if you’ve deleted this $Session global variable assignment:

$CurrentUser = Session::load()->get\_logged\_user();

That will generate a User[[7]](#footnote-7) object’s instance, with an extra *session* property.

This *session* additional property is an associative array containing the current session row extracted from the table sessions.

### How to determine if the user is logged in?

By default, the anonymous user has the user id -1, even if you don’t have added an anonymous user row in your users table.

You can change the anonymous user id adding the following constant in includes/config.php:

|  |  |  |
| --- | --- | --- |
| Named constant to define | Description | Default value |
| ANONYMOUS\_USER | The duration in seconds a user is still considered online after the last hit. | -1[[8]](#footnote-8) |

Knowing this, the easy way to check is the following code:

if ($CurrentUser->id > ANONYMOUS\_USER) {  
 //User is logged in   
}

The Pluton convention starts real user id from 1001, id till 999 being reserved for dummy/bots/system accounts (and 1000 as separator). Numeric comparison is still pertinent:

if ($CurrentUser->id > 1000) {  
 //User is logged in   
}

Or:

if ($CurrentUser->id < 1000) {  
 //Anonymous user  
}

## It’s now up to you.

*Xen & Pluton note*

There are extra initialization steps in Xen and Pluton index.php:

* **Xen** loads Smarty, the template engine, and then determines the controller to load.
* **Pluton** prepares the document to serve and gets metadata

The following white paper sections describe those steps with more details.

Now the site resources are loaded, it’s time to start your application.

You can load your template engine, start procedural code to initialize resources.

It’s also time to think about internationalization and localization.

Then, you can write the code to serve the page.

## Some tips

1. Keep your application entry point code clean and *aéré*.

It will allow you a modular approach when your application grows.

1. You can search [TODO] expressions in source code. Those comments have been added for you as repairs where to do things.
2. The Xen installer is shipped with an index.php creator. If you create on a regular basis new sites , spend some time to personalize and transform it in your own site generator solution.
3. Applications in production could be a great source of inspiration. The Keruald website lists open source applications built on the top of Xen or Pluton.

# Pluton’s index.php extra operations

[This section is a placeholder. In a future release of this white paper, it will describe how Pluton determines the document to serve and loads metadata]

# Xen’s index.php extra operations

[This section is a placeholder. In a future release of this white paper, it will describe how Xen initialize template engine, parses URL determines the controller to call]

# Index

A

anonymous user 7

C

config.php 4, 5

core.php 4

D

database 4, 5

E

error.php 4

L

logged in 7

Login 6

Logout 6

M

mysql.php 4

S

session 4

Session 5

session.php 4

1. A singleton class is a oriented object pattern which applies to situations in which there needs to be a single instance of a class. Implementing this pattern allows a programmer to make this single instance easily accessible by many other objects ([PHP Manual, the PHP Documentation Group, Classes and Objects, Patterns](http://www.php.net/manual/en/language.oop5.patterns.php)). [↑](#footnote-ref-1)
2. You can edit this default value in include/session.php, clean\_old\_sessions session method (around line 93) [↑](#footnote-ref-2)
3. Idem (around line 94) [↑](#footnote-ref-3)
4. You can edit this default value in SQL structure, executing this query (replace 3 by your new default value) :  
   ALTER TABLE sessions CHANGE session\_resource session\_resource TINYINT(4) UNSIGNED NOT NULL DEFAULT '3' [↑](#footnote-ref-4)
5. [↑](#footnote-ref-5)
6. P Passwords are stored in MD5 to maximize interoperability with other applications: instead to implement a SSO (single sign-on) solution, you can mirror your Pluton/Xen/Keruald users table in a view matching third party application users table structure. Cf. [Howto use your login/pass system to Bugzilla (or other app) without edit any line of code ?](http://www.dereckson.be/Blog/2007/07/08/howto-use-your-login-pass-system-to-bugzilla-or-other-app-without-edit-any-line-of-code/) in Dck Area :: Dereckson Personal Space blog. [↑](#footnote-ref-6)
7. User is a model class shipped with Keruald. Even if you don’t use a MVC approach, models – classes representing objects and allowing you to interfere with – are useful to clean your code from the load/save object from a database table or the result of a posted form. You can see models like classes mapping database fields as properties. [↑](#footnote-ref-7)
8. To change the default default value, edit the bottom of includes/session.php:

   //The user\_id matching anonymous user  
   if (!defined('ANONYMOUS\_USER')) define('ANONYMOUS\_USER', -1); [↑](#footnote-ref-8)