PHP Persistent Data

Overview

The PHP persistent data module, as outlined in this document, is intended to fill a growing need of web applications to have persistent, constant data and library functions available for every request made of the application. While there are mechanisms that currently exist for filling this need, these methods introduce overhead that we are trying to avoid through the implementation of this extension.

Persistent Data

Objective

The goal of having persistent data is to allow the developer to access specific data structures from any page within the application. We would like to do this in a way that would allow seamless integration of the data into the environment for purposes of data access, and use a limited number of functions for the registration/removal of the data structures. By making the data persistent, we hope to only have the overhead of creating the data once, and limit the new overhead involved with maintaining the data and inserting it into a request's environment.

Functionality

We propose that persistent data elements be accessed as if they were constants created via the existing define() function. In this manner, a developer can access the data via existing and known features within PHP.

While the access of these date elements will be familiar, we will need to create functions to allow for the definition and removal of the structured that belong in the persistent data space. These functions are proposed as follows:

```
pd_define($key, $value, $replace=false);
pd_remove($key);
```

Implementation Details

Use of pd_define() will insert a copy of the data into the persistent data space. This data space will be implemented internally as a zval hash. Prior to script execution, the module will add the elements from the persistent data space into the constant data table, making these elements available to future requests.

Possible Issues

The Zend Engine 2 will treat objects in a new way. From conversations to date, if you reference an object through a constant, you will actually get a reference to the object as opposed to a copy. Unless this functionality changes, there will need to be a copy created for each request prior to merging into the constant data area.

Persistent Functions & Classes

Objective

In developing web applications, it is common for the developer to create and uses various function and class libraries. These libraries are usually loaded, complied, and executed with each request that is made. Some attempt to reduce the overhead of repeatedly perfoming the loading and compiling by using some form of caching mechanism. While these mechanisms do work, we are seeking a more efficient manner by which to maintain these functions - by simply keeping them in memory.

In practice this would allow the developer to create extensions to the PHP language in PHP as opposed to C. These extensions would be loaded once and added to the function table of each request with minimal overhead.

Functionality

The functionality of this part of the extension would involve loading and compiling a set of PHP files at startup. The function and class tables would then be moved into a persistent function area. Functions and classes in this area would be merged into the request's function and class tables prior to script execution.

Implementation

A set if php.ini directives would determine the files to load and compile. None of the complied files would themselves be executed, which would disallow the possibility of conditional functions. It may be necessary to add the filenames to the included file list for the request.

Possible Issues

Error handling for persistent functions/classes can become an issue. How we handle parse errors will be key.

Proof of Concept

A proof of concept is current ly functioning with base function features. We are able to load a library, maintain the functions created in normal (as opposed to shared) memory, and add the functions into the request's function table, without copying the opcode array itself. A similar process should be able to be used for classes and persistent data.