GOOGLE APP ENGINE

CHAPTER 3

PHP: RUNTIME ENVIRONMENT

PHP Runtime Environment

Selecting the PHP runtime

App Engine knows to use the PHP runtime environment for your application code when you use the tool named`appcfg.py` from the PHP SDK with a configuration file named `app.yaml`.

You specify the `runtime` element in `app.yaml`:

```
runtime: php

api_version: 1

...
```

The first element, `runtime`, selects the PHP runtime environment.

The second element, `api_version`, selects which version of the PHP runtime environment to use. As of this writing, App Engine only has one version of the PHP environment, `1`. If the App Engine team ever needs to release changes to the environment that may not be compatible with existing code, they will do so with a new version identifier. Your app will continue to use the selected version until you change the `api_version` setting and upload your app.

For more information about `app.yaml` and `appcfg.py`, see PHP Application Configuration, and Uploading an App.

The sandbox

To allow App Engine to distribute requests for applications across multiple web servers, and to prevent one application from interfering with another, the application runs in a restricted "sandbox" environment. In this environment, the application can execute code, use the App Engine mail, URL fetch and users services, and examine the user's web request and prepare the response.

An App Engine application cannot:

- write to the filesystem. PHP applications can use Google Cloud Storage for storing persistent files. Reading from the filesystem is allowed, and all application files uploaded with the application are available.

- respond slowly. A web request to an application must be handled within a few

seconds. Processes that take a very long time to respond are terminated to avoid overloading the web server.

•make other kinds of system calls.

Automatic class loading

Both Standard PHP Library (SPL) [http://php.net/spl] classes and any classes that are part of the PHP SDK for App Engine are automatically loaded when needed. This means that you do not have to use `include` or `require`statements at the top of your PHP scripts.

By default, automatic class loading will occur only for classes defined in files that reside in the App Engine PHP SDK root (and, if it has been specified by `--php_executable_path`, your local PHP installation).

To add more paths to be searched for automatic class loading, use `set_include_path` in your PHP script.

```
set_include_path('my_additional_path' . PATH_SEPARATOR . get_include_path());
```

Enabled extensions

The following extensions have been enabled in the PHP runtime for App Engine:

•apc	•gd	•mysqli	•SimpleXML
•bcmath	•hash	•mysqlnd	•soap
•calendar	•iconv	•OAuth	•SPL
•Core	•json	•openssl	•standard
•ctype	•libxml	•pcre	•tokenizer
•date	•mbstring	•PDO	•xml
•dom	•mcrypt	•pdo_mysql	•xmlreader
•ereg	•memcache	•Reflection	•xmlwriter
•filter	•memcached	•session	•Zip
•FTP	•mysql	•shmop	•zlib

Sessions

Most web applications need a way to preserve user state information between requests.

PHP provides a convenient session management layer. Sessions in App Engine work much like sessions in any other PHP application.

Setting a variable in a user's session:

```
session_start();

$_SESSION['Foo'] = 'Bar';
```

On a subsequent request by the same user:

```
session_start();

print $_SESSION['Foo']; // prints Bar
```

By default the App Engine runtime will use memcache to store session information using the`MemcacheSessionHandler` class. You can adjust this behavior by specifying your own session handler using PHP's `session_set_save_handler()` method. Memcache allows session data to be saved and retrieved quickly, meaning the overhead on your request is minimal. However data in App Engine memcache may be flushed periodically, meaning any session information will be lost. For longer-lived sessions, it may be preferable to use an alternative storage service such as Cloud SQL.

Warning: `MemcacheSessionHandler` uses Memcache to store values, meaning that by default, session data will be transient and corruptable by concurring requests that have the same session token (which can occur when users hit "refresh" while waiting for a request to complete, etc). This issue can be solved by implementing your own session functionality using a data storage solution that supports transactions, such as Google Cloud SQL.

Special $_SERVER keys

PHP makes the special `$_SERVER[]` array available to in the request scope. In addition to the standard CGI paramaters, App Engine adds some additional useful keys.

- `APPLICATION_ID` - The app_id of the application set when the app was created. eg. my-wordpress
- `AUTH_DOMAIN` - The domain used for authenticating users with the Users API. Apps hosted on appspot.com have an AUTH_DOMAIN of gmail.com, and accept any Google account. Apps hosted on a custom domain using Google Apps have an AUTH_DOMAIN equal to the custom domain
- `CURRENT_VERSION_ID` - The major and minor version of the currently running application, as "X.Y". The major version number ("X") is specified in the app's

app.yaml file. The minor version number ("Y") is set automatically when each version of the app is uploaded to App Engine. On the development web server, the minor version is always "1".

•`DEFAULT_VERSION_HOSTNAME` - The hostname of the default version of this application, eg. my-php-app.appspot.com.

•`HTTP_X_APPENGINE_CITY` - Name of the city from which the request originated. For example, a request from the city of Mountain View might have the header value mountain view.

•`HTTP_X_APPENGINE_CITYLATLONG` - Latitude and longitude of the city from which the request originated. This string might look like "37.386051,-122.083851" for a request from Mountain View.

•`HTTP_X_APPENGINE_COUNTRY` - Country from which the request originated, as an ISO 3166-1 alpha-2 country code. App Engine determines this code from the client's IP address.

•`HTTP_X_APPENGINE_REGION` - Name of region from which the request originated. This value only makes sense in the context of the country in X-AppEngine?-Country. For example, if the country is "US" and the region is "ca", that "ca" means "California", not Canada.

•`USER_EMAIL` - Returns the email address of the user, if they have been authenticated using the Users API. If you use OpenID, you should not rely on this email address to be correct. Applications should use nickname for displayable names.

•`USER_ID` - If the email address is associated with a Google account, user_id returns the unique permanent ID of the user, a str. If they have been authenticated using the Users API. This ID is always the same for the user regardless of whether the user changes her email address.

•`USER_IS_ADMIN` - 1 if the logged in user is also an Adminstrator of the application, if they have been authenticated using the Users API. 0 otherwise.

•`USER_NICKNAME` - For Google Accounts users, the nickname is either the "name" portion of the user's email address if the address is in the same domain as the application, or the user's full email address otherwise. For OpenID users, the nickname is the OpenID identifier.

•`USER_ORGANIZATION` - An application using the Google Accounts setting can determine if the currently signed-in user is using a personal Google Account or an account which is managed by a Google Apps domain.

Important: In App Engine release 1.9.0, the behavior of the environment variables`$_SERVER['SCRIPT_NAME']` and `$_SERVER['PHP_SELF']` changed significantly, in order to be consistent with the Apache implementation generally expected by PHP applications. For details, see Updated `PHP_SELF`and `SCRIPT_NAME` behavior in 1.9.0

Directives with new initialization defaults

This table specifies directives whose initialization defaults differ from the defaults supplied with the standard PHP 5.4 interpreter available from php.net. You can override these default directives by including them in a`php.ini` file for your application.

Directive	Default Value in Google App Engine
`detect_unicode`	`false`
`session.gc_maxlifetime`	`600`
`session.cookie_secure`	`600`
`session.cookie_httponly`	`1`
`session.use_only_cookies`	`1`
`display_errors`	`0`
`display_startup_errors`	`0`
`html_errors`	`0`
`log_errors`	`1`
`file_uploads`	`0`
`upload_max_filesize`	`262144`
`max_file_uploads`	`0`
`date.timezone`	`UTC`
`sendmail_path`	`null`
`allow_url_fopen`	`1`
`allow_url_include`	`0`
`enable_dl`	`0`
`expose_php`	`Off`
`register_globals`	`Off`
`magic_quotes_gpc`	`0`
`mysqlnd.collect_statistics`	`0`
`mysql.allow_local_infile`	`0`

Directive	Default Value in Google App Engine
`mysqli.allow_local_infile`	`0`

Disabled Functions

Either for security reasons, or for compatibility with Google App Engine execution environment, some PHP functions have been disabled. Some of these functions can be explicitly re-enabled in the php.ini file for your application.

Permanently disabled functions

The following functions have been permanently disabled in Google App Engine:

- `disk_free_space()`
- `disk_total_space()`
- `diskfreespace()`
- `escapeshellarg()` and `escapeshellcmd()`
- `exec()`
- `highlight_file()`
- `lchgrp()`, `lchown()`, `link()`, and `symlink()`
- `passthru()`
- `pclose()` and `popen()`
- `proc_close()`, `prog_get_status()`, `proc_nice()`, `proc_open()`, and `proc_terminate()`
- `set_time_limit()`
- `shell_exec()`
- `show_source()`
- `system()`
- `tempnam()`

Google App Engine does not include the pcntl module, and thus the functions provided by `pcntl` are not available to PHP programs running in Google App Engine.

Partially restricted functions

The Google App Engine for PHP runtime does not support the `/e` pattern modifier of the `preg_replace()` and`mg_ereg_replace()` functions. See the `PREG_REPLACE_EVAL` documentation for the deprecation notice and an example of how to update your code to use `preg_replace_callback()` instead.

Functions that may be manually enabled

This list specifies the PHP function that must be manually enabled by using the`google_app_engine.enable_functions` directive in the `php.ini` file for your application.

- `gc_collect_cycles()`, `gc_enable()`, `gc_disable()` and `gc_enabled()`
- `getmypid()`
- `getmyuid()` and `getmygid()`
- `getrusage()`
- `getmyinode()`
- `get_current_user()`
- `libxml_disable_entity_loader()`*
- `parse_str()`
- `phpinfo()`
- `phpversion()`
- `php_uname()`
- `php_sapi_name()`

You can also manually disable functions by using the `disable_functions` directive in the `php.ini` file for your application.

Note: You must also call `libxml_disable_entity_loader(false)` somewhere in your application's startup script to enable loading of external XML entities.

Stream support

Stream wrappers

Many functions in PHP such as `fopen()` or `file_get_contents()` take advantage of PHP's streams interface to support different protocols.

The following is a list of built-in stream wrappers that are automatically registered and available in the App Engine runtime.

- `file://`
- `glob://`
- `http://` (This behaves like PHP's built-in http stream handler, but uses the App Engine URLfetch service)
- `https://` (This uses the App Engine URLfetch service)
- `gs://` (The stream handler for Google Cloud Storage)
- `php://`
- `zlib://`

The following is a list of built-in stream handlers that are not supported in Google App Engine and have have been unregistered.

- `ftp://`
- `data://`
- `phar://`
- `ssh2://`
- `rar://`
- `ogg://`
- `expect://`

Disabled stream transports

The following stream transports have been disabled.

- `ssl`
- `sslv2`
- `sslv3`
- `tcp`
- `tls`
- `udg`
- `udp`
- `unix`

Pure PHP

All code for the PHP runtime environment must be pure PHP. App Engine does not allow you to upload your own C extensions.

The environment includes the PHP 5.4 standard library. Some modules have been disabled because their core functions are not supported by App Engine, such as networking or writing to the filesystem.

You can include other pure PHP libraries with your application by putting the code in your application directory. If you make a symbolic link to a module's directory in your application directory, appcfg.py will follow the link and include the module in your app.

The PHP module include path includes your application's root directory (the directory containing the `app.yaml`file). Modules you create in your application's root directory are available using a path from the root.

The PHP SDK and tools

The App Engine PHP SDK includes tools for testing your application and uploading application files.

PHP interpreter source code

You can download the source code for App Engine's PHP interpreter here at this GitHub repository:https://github.com/GoogleCloudPlatform/appengine-php