GOOGLE APP ENGINE

CHAPTER 5

PHP: TUTORIAL

PHP Tutorial

Introduction

The PHP runtime is available as an experimental Preview feature. Please be aware that the APIs and service may change before the service becomes Generally Available.

Welcome to Google App Engine! Creating an App Engine application is easy, and only takes a few minutes. And it's free to start: upload your app and share it with users right away, at no charge and with no commitment required.

Google App Engine applications can be written in the PHP, Java, Python or Go programming languages. This tutorial covers **PHP**. If you would prefer to use Java, Python or Go to build your applications, see the Java, Python 2.7 or Go guides.

In this tutorial, you will learn how to:

- build an App Engine application using PHP
- integrate an App Engine application with Google Accounts for user authentication
- use Google Cloud SQL to store your data
- upload your app to App Engine

By the end of the tutorial, you will have implemented a working application, a simple guest book that lets users post messages to a public message board.

Get set up

Before we continue, you will need to download the Google App Engine PHP SDK, which includes a web server application that simulates the App Engine environment, and tools to deploy your application to the App Engine production environment. Follow the directions for your operating system, then come back here so we can get going!

Hello, World!

Let's begin by implementing a tiny application that displays a short message.

Creating a Simple Script

Create a directory named `helloworld`. All files for this application reside in this directory.

Inside the `helloworld` directory, create a file named `helloworld.php`, and give it the following contents:

```
<?php
  echo 'Hello, World!';
```

This PHP script responds to all requests with the message `Hello, world!`.

Creating the Configuration File

An App Engine application has a configuration file called `app.yaml`. Among other things, this file describes which handler scripts should be used for which URLs.

Inside the `helloworld` directory, create a file named `app.yaml` with the following contents:

```
application: helloworld
version: 1
runtime: php
api_version: 1

handlers:
- url: /.*
  script: helloworld.php
```

- The application identifier is `helloworld`. Every new application on App Engine has a unique application identifier. You'll choose the identifier for your application when you register it in the next step. Until then you can just leave the value here set to `helloworld` because this value is not important when developing locally.
- This is version number `1` of this application's code. If you adjust this before uploading new versions of your application software, App Engine will retain previous versions,

and let you roll back to a previous version using the administrative console.

- This code runs in the `php` runtime environment, version "1". Additional runtime environments and languages may be supported in the future.
- Every request to a URL whose path matches the regular expression `/.*` (all URLs) should be handled by the `helloworld.php` script.

The syntax of this file is YAML. For a complete list of configuration options, see the app.yaml reference.

Testing the Application

With a handler script and configuration file mapping every URL to the handler, the application is complete. You can now test it with the web server included with the App Engine SDK.

If you're using the Google App Engine Launcher, you can set up the application by selecting the **File** menu, **Add Existing Application...**, then selecting the `helloworld` directory. Select the application in the app list, click theRun button to start the application, then click the Browse button to view it. Clicking Browse simply loads (or reloads) http://localhost:8080/ in your default web browser.

If you're not using Google App Engine Launcher, start the web server with the following command, giving it the path to the `helloworld` directory:

```
google_appengine/dev_appserver.py helloworld/
```

The web server is now running, listening for requests on port 8080. You can test the application by visiting the following URL in your web browser:

- http://localhost:8080/

For more information about running the development web server, including how to change which port it uses, seethe Dev Web Server reference, or run the command with the option `--help`.

Next...

You now have a complete App Engine application! You could deploy this simple greeting right now and share it with users worldwide. But before we deploy it, let's take a closer look some more interesting App Engine features.

Using the Users Service

Google App Engine provides several useful services based on Google infrastructure, accessible by applications using libraries included with the SDK. One such service is the Users service, which lets your application integrate with Google user accounts. With the Users service, your users can use the Google accounts they already have to sign in to your application.

Let's use the Users service to personalize this application's greeting.

Using Users

Edit `helloworld/helloworld.php` again, and replace its contents with the following:

View File | View Project | Download Project ZIP

```
<?php

use google\appengine\api\users\User;
use google\appengine\api\users\UserService;
# Looks for current Google account session
$user = UserService::getCurrentUser();
if ($user) {
  echo 'Hello, ' . htmlspecialchars($user->getNickname());
}
else {
  header('Location: ' . UserService::createLoginURL($_SERVER['REQUEST_URI']));
}
```

Reload the page in your browser. Your application redirects you to the local version of the Google sign-in page suitable for testing your application. You can enter any username you'd like in this screen, and your application will see a fake `User` object based on that username.

When your application is running on App Engine, users will be directed to the Google Accounts sign-in page, then redirected back to your application after successfully signing in or creating an account.

The Users API

Let's take a closer look at the new pieces:

```
# Looks for current Google account session
$user = UserService::getCurrentUser();
```

View File | View Project | Download Project ZIP

If the user is already signed in to your application, `getCurrentUser()` returns

the `User` object for the user. Otherwise, it returns `null`.

```
if ($user) {
  echo 'Hello, ' . htmlspecialchars($user->getNickname());
}
```

View File | View Project | Download Project ZIP

If the user has signed in, display a personalized message, using the nickname associated with the user's account.

```
else {
  header('Location: ' . UserService::createLoginURL($_SERVER['REQUEST_URI']));
}
```

View File | View Project | Download Project ZIP

If the user has not signed in, redirect the user's browser to the Google account sign-in screen. The redirect includes the URL to this page (through the inclusion of `$_SERVER['REQUEST_URI']`) so the Google account sign-in mechanism will send the user back here after the user has signed in or registered for a new account.

For more information about the Users API, see the Users reference.

Handling Forms

If we want users to be able to post their own greetings, we need a way to process information submitted by the user with a web form. PHP makes processing form data easy.

Handling Web Forms

Replace the contents of `helloworld/helloworld.php` with the following:

```
<html>
  <body>
    <?php
    if (array_key_exists('content', $_POST)) {
      echo "You wrote:<pre>\n";
      echo htmlspecialchars($_POST['content']);
      echo "\n</pre>";
    }
    ?>
    <form action="/sign" method="post">
      <div><textarea name="content" rows="3" cols="60"></textarea></div>
      <div><input type="submit" value="Sign Guestbook"></div>
    </form>
  </body>
</html>
```

View File | View Project | Download Project ZIP

Reload the page to see the form, then try submitting a message.

When the form is submitted, the application receives a request using the HTTP POST method (`method="post"`) and PHP makes POSTed form variables available using the `$_POST` superglobal.

Let's take a closer look at how the form is processed:

```
if (array_key_exists('content', $_POST)) {
  echo "You wrote:<pre>\n";
  echo htmlspecialchars($_POST['content']);
  echo "\n</pre>";
}
```

View File | View Project | Download Project ZIP

Before displaying the message (stored in `<textarea name="content"`), the application checks to see if is present in the `$_POST` superglobal. If it is then special HTML characters in the message (like "<") are replaced with their corresponding HTML entities (like <) using the `htmlspecialchars` function.

Next...

Every web application returns dynamically generated HTML from the application code, via templates or some other mechanism. Most web applications also need to serve static content, such as images, CSS stylesheets, or JavaScript files. For efficiency, App Engine treats static files differently from application source and data files. You can use App Engine's static files feature to serve a CSS stylesheet for this application.

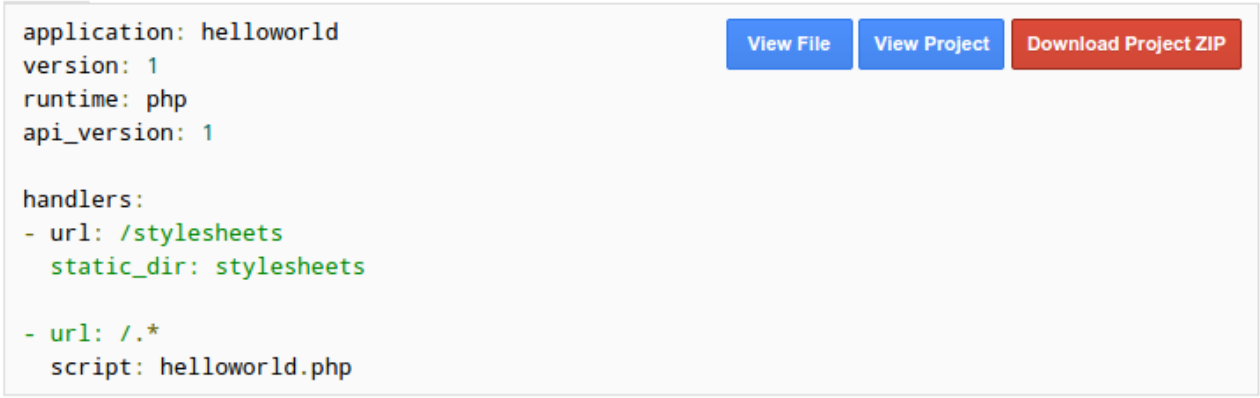
Using Static Files

Unlike a traditional web hosting environment, Google App Engine does not serve files directly out of your application's source directory unless configured to do so.

But there are many cases where you want to serve static files directly to the web browser. Images, CSS stylesheets, JavaScript code, movies and Flash animations are all typically stored with a web application and served directly to the browser. You can tell App Engine to serve specific files directly without your having to code your own handler.

Using Static Files

Edit `helloworld/app.yaml` and replace its contents with the following:




```
application: helloworld
version: 1
runtime: php
api_version: 1

handlers:
- url: /stylesheets
  static_dir: stylesheets

- url: /.*
  script: helloworld.php
```

The new `handlers` section defines two handlers for URLs. When App Engine receives a request with a URL beginning with `/stylesheets`, it maps the remainder of the path to files in the `stylesheets` directory and, if an appropriate file is found, the contents of the file are returned to the client. All other URLs match the `/.*` path, and are handled by the `helloworld.php` script.

By default, App Engine serves static files using a MIME type based on the filename extension. For example, a file with a name ending in `.css` will be served with a MIME type of `text/css`. You can configure explicit MIME types by using the `mime_type` setting when configuring your handlers in `app.yaml`.

URL handler path patterns are tested in the order they appear in `app.yaml`, from top to bottom. In this case, the`/stylesheets` pattern will match before the `/.*` pattern will for the appropriate paths. For more information on URL mapping and other options you can specify in `app.yaml`, see the app.yaml reference.

Create the directory `helloworld/stylesheets`. In this new directory, create a new file named `main.css` with the following contents:

```
body {
  font-family: Verdana, Helvetica, sans-serif;
  background-color: #DDDDDD;
}
```

View File View Project Download Project ZIP

Finally, edit `helloworld/helloworld.php` and insert the following lines just after the `<html>` line at the top:

```
<head>
  <link type="text/css" rel="stylesheet" href="/stylesheets/main.css" />
</head>
```

View File View Project Download Project ZIP

Reload the page in your browser. The new version uses the stylesheet.

Uploading Your Application

You create and manage App Engine applications using the Google Developers Console. Once you have registered an application ID for your application, you upload it to your website using `appcfg.py`, a command-line tool provided in the SDK.

Note: Application IDs must begin with a letter. Once you register an application ID, you can delete it, but you can't re-register that same application ID after it has been deleted. You can skip these next steps if you don't want to register an ID at this time.

Note: If you have an App Engine Premier account, you can specify that your new application should reside in the European Union rather than the United States. Developers that do not have a Premier account need to fill out this form and enable billing for applications that should reside in the European Union.

Hosting applications in the European Union is especially useful if your users are closer to Europe than to the United States. There is less network latency and the End User Content will be stored at rest in the European Union. You must specify this location by clicking the "Edit" link in the "Location Options" section when you register the application; you cannot change it later.

Registering the Application

You create and manage App Engine applications from the Developers Console, at the following URL:

https://console.developers.google.com/

Sign in to App Engine using your Google account. If you do not have a Google account, you can create a Google account with an email address and password.

Note: You may have already created a project using the Google Developers Console. If this is the case, you do not have to create a new application. Your project has a title and an ID. In the instructions that follow, the*project* title and ID can be used wherever an *application* title and ID are mentioned. They are the same thing.

To create a new application, click the "Create an Application" button. Follow the instructions to register an application ID, a name unique to this application. If you elect to use the free appspot.com domain name, the full URL for the application will be `http://your_app_id.appspot.com/`. You can also purchase a top-level domain name for your app, or use one that you have already registered.

Edit the `app.yaml` file, then change the value of the `application:` setting from `helloworld` to your registered application ID.

If your application uses Google Cloud SQL for data storage (none of the examples in the PHP Getting Started tutorial do) then you must also request a Google Cloud SQL instance.

Uploading the Application

You can upload your finished application to Google App Engine in a few ways.

Uploading via command line

Run the following command:

```
appcfg.py update helloworld/
```

Enter your Google username and password at the prompts.

Use the App Engine Launcher

You can also use the interface of the Google App Engine Launcher to upload your application. Simply select the projet in Launcher and click **Deploy**.

Using Git

If you work with the Git version control system, you can create a remote repository in Google's cloud, and configure your development environment to deploy the latest version of your code each time you push it to that repository. See Using Git to Push and Deploy.

You can now see your application running on App Engine. If you set up a free appspot.com domain name, the URL for your website begins with your application ID:

`http://your_app_id.appspot.com`

Congratulations!

You have completed this tutorial. For more information on the subjects covered here, see the rest of the App Engine documentation.