PHP (https://cloud.google.com/php/) Guides

Using Cloud SQL for PostgreSQL with PHP

This part of the PHP Bookshelf app tutorial shows how to create, read, update, and delete structured data in Cloud SQL for PostgreSQL (https://cloud.google.com/sql/docs/postgres).

This page is part of a multipage tutorial. To start from the beginning and read the setup instructions, go to PHP Bookshelf app (https://cloud.google.com/php/getting-started/tutorial-app).

Creating a Cloud SQL instance and database

Installing the Cloud SQL Proxy

Download and install the Cloud SQL Proxy. The Cloud SQL Proxy connects to your Cloud SQL instance when running locally.

LINUX 64-BIT | LINUX 32-BIT | MORE

1. Download the proxy:

```
wget https://dl.google.com/cloudsql/cloud_sql_proxy.linux.amd64 -O cloud_sql_
```

2. Make the proxy executable:

```
chmod +x cloud_sql_proxy
```

If your operating system isn't included here, you can also compile the proxy from source (http://github.com/GoogleCloudPlatform/cloudsql-proxy).

Creating a Cloud SQL instance

1. Create a Cloud SQL for PostgreSQL instance. (https://cloud.google.com/sql/docs/postgres/create-instance)

   Name the instance `library` or similar. It can take a few minutes for the instance to be ready. When the instance is ready, it's visible in the instances list.

2. Use the Cloud SDK to run the following command where `[YOUR_INSTANCE_NAME]` represents the name of your Cloud SQL instance:

   ```
   gcloud sql instances describe [YOUR_INSTANCE_NAME]
   ```

   In the output, note the value shown for `[CONNECTION_NAME]`.

   The `[CONNECTION_NAME]` value is in the format `[PROJECT_NAME]:[REGION_NAME]:[INSTANCE_NAME]`.

Initializing your Cloud SQL instance

1. Start the Cloud SQL Proxy by using the `[CONNECTION_NAME]` value from the previous step:

   LINUX/MACOS | WINDOWS

   ```
   ./cloud_sql_proxy -instances="[YOUR_INSTANCE_CONNECTION_NAME]"=tcp:5432
   ```

   Replace `[YOUR_INSTANCE_CONNECTION_NAME]` with the `[CONNECTION_NAME]` value that you recorded in the previous step.

   This step establishes a connection from your local computer to your Cloud SQL instance for local testing purposes. Keep the Cloud SQL Proxy running the entire time you test your app locally.

2. Create a Cloud SQL user and database:

   CLOUD CONSOLE | POSTGRES CLIENT

   a. Create a new database by using the Cloud Console (https://cloud.google.com/sql/docs/postgres/create-manage-databases#create) for your Cloud SQL instance `library`. For example, you can use the name `bookshelf`.

b. Create a new user by using the Cloud Console (https://cloud.google.com/sql/docs/postgres/create-manage-users#creating) for your Cloud SQL instance `library`.

Configuring settings

1. In your terminal window, go to the `getting-started-php/2-structured-data` directory, and copy the `settings.yml.dist` file:

```
cp config/settings.yml.dist config/settings.yml
```

2. Open `config/settings.yml` for editing.
3. Replace `[YOUR_PROJECT_ID]` with your Google Cloud project ID.
4. Set the value of `bookshelf_backend` to `postgres`.
5. Set the values of `cloudsql_connection_name`, `cloudsql_database_name`, `cloudsql_user`, `cloudsql_password`, and `cloudsql_port` to the appropriate values for your Cloud SQL instance. Since you're using `postgres`, use port `5432`. For example:

```
cloudsql_connection_name: [YOUR_PROJECT_NAME]:[YOUR_REGION]:[YOUR_INSTANCE]
cloudsql_database_name: bookshelf
cloudsql_user: phpapp
cloudsql_password: password
cloudsql_port: 5432
```

6. Save and close `config/settings.yml`.

You also need to update the `app.yaml` file before deploying:

1. Open `app.yaml` for editing.
2. Uncomment the `beta_settings` and `cloud_sql_instances` lines.
3. Set the value of `cloud_sql_instances` to the value you used for `cloudsql_connection_name` in `config/settings.yml`. The value uses the format `[YOUR_PROJECT_NAME]:[YOUR_REGION]:[YOUR_INSTANCE]`.
4. Save and close `app.yaml`.

Installing dependencies

In the `2-structured-data` directory, enter this command:

```
composer install
```

Running the app on your local machine

1. Start a local web server:

   ```
   php -S localhost:8000 -t web
   ```

2. In your browser, enter this address:

   http://localhost:8000 (http://localhost:8000)

Now you can browse the app's web pages and add, edit, and delete books.

Deploying the app to the App Engine flexible environment

1. In your terminal window, deploy the sample app:

   ```
   gcloud app deploy
   ```

2. In your browser, enter this address. Replace `[YOUR_PROJECT_ID]` with your Google Cloud project ID:

   ```
   https://[YOUR_PROJECT_ID].appspot.com
   ```

If you update your app, you can deploy the updated version by entering the same command you used when you first deployed the app. The new deployment creates a version of your app and promotes it to the default version.

The older versions of your app remain, as do their associated virtual machine (VM) instances. Be aware that all of these app versions and VM instances are billable resources.

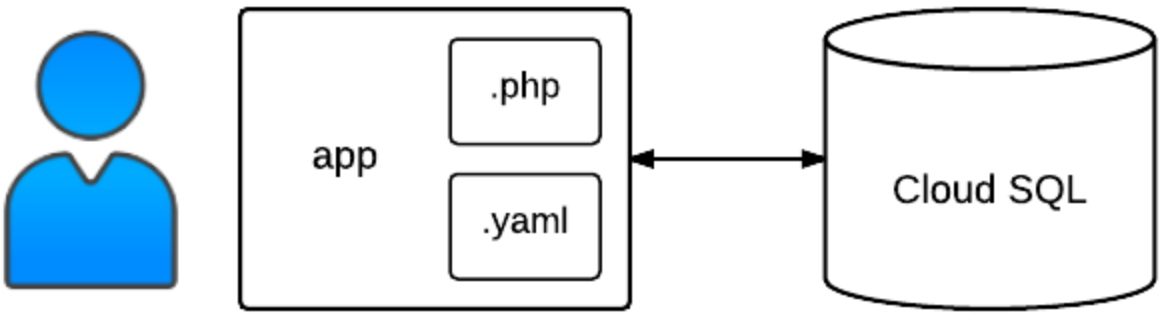
You can reduce costs by deleting the non-default versions of your app (https://cloud.google.com/php/getting-started/using-cloud-storage#delete_non-default_versions_of_your_app)

.

When you finish this tutorial, you can avoid continued billing by deleting the resources you created. See Cleaning up (https://cloud.google.com/php/getting-started/using-cloud-storage#clean-up) for more detail.

App structure

This diagram shows the app's components and how they fit together.



Understanding the code

Previously, you edited `settings.yml` and set the value of `bookshelf_backend` to `postgres`. This setting tells the app to load the `Sql` class, which is defined in `src/DataModel/Sql.php`. The `Sql` class wraps the PDO API (https://php.net/manual/en/book.pdo.php) and is responsible for storing books in your Cloud SQL database.

This code in `controllers.php` defines and registers a handler for the `GET '/books'` route. The `$model` variable is an instance of the `Sql` class. The `$model->listBooks` method returns an array that contains an array of books and a cursor. Then the Twig template engine (https://twig.symfony.com/) renders the list of books according to the `list.html.twig` template:

2-structured-data/src/controllers.php (https://github.com/GoogleCloudPlatform/getting-started-php/blob/steps/2-structured-data/src/controllers.php)

LOUDPLATFORM/GETTING-STARTED-PHP/BLOB/STEPS/2-STRUCTURED-DATA/SRC/CONTROLLERS.PHP)

```
$app->get('/books/', function (Request $request) use ($app) {
    /** @var DataModelInterface $model */
    $model = $app['bookshelf.model'];
    /** @var Twig_Environment $twig */
    $twig = $app['twig'];
    $token = $request->query->get('page_token');
    $bookList = $model->listBooks($app['bookshelf.page_size'], $token);

    return $twig->render('list.html.twig', array(
        'books' => $bookList['books'],
        'next_page_token' => $bookList['cursor'],
    ));
});
```

Here is the Twig template for listing books that are retrieved from the Cloud SQL database. The template receives an array variable named `books`. For each book in the array, it displays the title and author. The template also receives a `next_page_token` variable that determines whether the **More** button is displayed.

2-structured-data/templates/list.html.twig (https://github.com/GoogleCloudPlatform/getting-started-php/blob/steps/2-structured-data/templates/list.html.twig)

DPLATFORM/GETTING-STARTED-PHP/BLOB/STEPS/2-STRUCTURED-DATA/TEMPLATES/LIST.HTML.TWIG)

```
{% for book in books %}
<div class="media">
  <a href="/books/{{book.id}}">
    <div class="media-left">
      <img src="http://placekitten.com/g/128/192">
    </div>
    <div class="media-body">
      <h4>{{book.title}}</h4>
      <p>{{book.author}}</p>
    </div>
  </a>
</div>
{% else %}
<p>No books found</p>
{% endfor %}
```

This code defines and registers a handler for the `GET '/books/{id}'` route, where `{id}` is the ID of an individual book. The handler calls the `$model->read` method to get the specified book from Cloud SQL. The Twig template engine renders the book according to the `view.html.twig` template:

2-structured-data/src/controllers.php (https://github.com/GoogleCloudPlatform/getting-started-php/blob/steps/2-structured-data/src/controllers.php)

LOUDPLATFORM/GETTING-STARTED-PHP/BLOB/STEPS/2-STRUCTURED-DATA/SRC/CONTROLLERS.PHP)

```
$app->get('/books/{id}', function ($id) use ($app) {
    /** @var DataModelInterface $model */
    $model = $app['bookshelf.model'];
    $book = $model->read($id);
    if (!$book) {
        return new Response('', Response::HTTP_NOT_FOUND);
    }
    /** @var Twig_Environment $twig */
    $twig = $app['twig'];

    return $twig->render('view.html.twig', array('book' => $book));
});
```

The `view.html.twig` template receives a variable named `book` and displays the book's title, publication date, author, and description.

2-structured-data/templates/view.html.twig (https://github.com/GoogleCloudPlatform/getting-started-php/blob/steps/2-structured-data/templates/view.html.twig)

)PLATFORM/GETTING-STARTED-PHP/BLOB/STEPS/2-STRUCTURED-DATA/TEMPLATES/VIEW.HTML.TWIG)

```
<div class="media">
  <div class="media-body">
    <h4 class="book-title">
      {{book.title}}
      <small>{{book.published_date}}</small>
    </h4>
    <h5 class="book-author">By {{book.author|default('Unknown', True)}}</h5>
    <p class="book-description">{{book.description}}</p>
  </div>
</div>
```

When the user clicks **Add book**, the handler for `GET /books/add` displays a form for entering the title, author, and other information about a book. When the user clicks **Save**, the handler for `POST /books/add` gets the new book from the request and calls `$model->create` to store the book in Cloud SQL:

2-structured-data/src/controllers.php (https://github.com/GoogleCloudPlatform/getting-started-php/blob/steps/2-structured-data/src/controllers.php)

LOUDPLATFORM/GETTING-STARTED-PHP/BLOB/STEPS/2-STRUCTURED-DATA/SRC/CONTROLLERS.PHP)

```
$app->get('/books/add', function () use ($app) {
    /** @var Twig_Environment $twig */
    $twig = $app['twig'];

    return $twig->render('form.html.twig', array(
        'action' => 'Add',
        'book' => array(),
    ));
});

$app->post('/books/add', function (Request $request) use ($app) {
    /** @var DataModelInterface $model */
    $model = $app['bookshelf.model'];
    $book = $request->request->all();
    $id = $model->create($book);

    return $app->redirect("/books/$id");
});
```

Here's the template for the book entry form:

2-structured-data/templates/form.html.twig (https://github.com/GoogleCloudPlatform/getting-started-php/blob/steps/2-structured-data/templates/form.html.twig)

PLATFORM/GETTING-STARTED-PHP/BLOB/STEPS/2-STRUCTURED-DATA/TEMPLATES/FORM.HTML.TWIG)

```
{% extends "base.html.twig" %}

{% block content %}
<h3>{{action}} book</h3>

<form method="POST" enctype="multipart/form-data">
```

```
  <div class="form-group">
    <label for="title">Title</label>
    <input type="text" name="title" id="title" value="{{book.title}}" class="form-co
  </div>

  <div class="form-group">
    <label for="author">Author</label>
    <input type="text" name="author" id="author" value="{{book.author}}" class="form
  </div>

  <div class="form-group">
    <label for="published_date">Date Published</label>
    <input type="text" name="published_date" id="published_date" value="{{book.publi
  </div>

  <div class="form-group">
    <label for="description">Description</label>
    <textarea name="description" id="description" class="form-control">{{book.descri
  </div>

  <button id="submit" type="submit" class="btn btn-success">Save</button>
</form>

{% endblock %}
```

The sample code includes more handlers for editing and deleting individual books:

2-structured-data/src/controllers.php (https://github.com/GoogleCloudPlatform/getting-started-php/blob/steps/2-structured-data/src/controllers.php)

LOUDPLATFORM/GETTING-STARTED-PHP/BLOB/STEPS/2-STRUCTURED-DATA/SRC/CONTROLLERS.PHP)

```
$app->get('/books/{id}/edit', function ($id) use ($app) {
    /** @var DataModelInterface $model */
    $model = $app['bookshelf.model'];
    $book = $model->read($id);
    if (!$book) {
        return new Response('', Response::HTTP_NOT_FOUND);
    }
    /** @var Twig_Environment $twig */
    $twig = $app['twig'];

    return $twig->render('form.html.twig', array(
```

```
        'action' => 'Edit',
        'book' => $book,
    ));
});

$app->post('/books/{id}/edit', function (Request $request, $id) use ($app) {
    $book = $request->request->all();
    $book['id'] = $id;
    /** @var DataModelInterface $model */
    $model = $app['bookshelf.model'];
    if (!$model->read($id)) {
        return new Response('', Response::HTTP_NOT_FOUND);
    }
    if ($model->update($book)) {
        return $app->redirect("/books/$id");
    }

    return new Response('Could not update book');
});
```

2-structured-data/src/controllers.php (https://github.com/GoogleCloudPlatform/getting-started-php/blob/steps/2-structured-data/src/controllers.php)

LOUDPLATFORM/GETTING-STARTED-PHP/BLOB/STEPS/2-STRUCTURED-DATA/SRC/CONTROLLERS.PHP)

```
$app->post('/books/{id}/delete', function ($id) use ($app) {
    /** @var DataModelInterface $model */
    $model = $app['bookshelf.model'];
    $book = $model->read($id);
    if ($book) {
        $model->delete($id);

        return $app->redirect('/books/', Response::HTTP_SEE_OTHER);
    }

    return new Response('', Response::HTTP_NOT_FOUND);
});
```

< PREV (HTTPS://CLOUD.GOOGLE.COM/PHP/GETTING-STARTED/USING-STRUCTURED-DATA)

NEXT > (HTTPS://CLOUD.GOOGLE.COM/PHP/GETTING-STARTED/USING-CLOUD-STORAGE)

Last updated December 4, 2019.