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

Using MongoDB with PHP

This part of the PHP Bookshelf app tutorial shows how to create, read, update, and delete structured data in a MongoDB database.

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).

Running MongoDB on Compute Engine

If you haven't set up a MongoDB instance, you can create a managed instance running on Google Cloud by using Atlas on Google Cloud (https://www.mongodb.com/cloud/atlas/mongodb-google-cloud) or mLab (https://mlab.com/google/). If you'd rather use an unmanaged MongoDB instance running on Compute Engine, check out Bitnami's preconfigured click-to-deploy solution (https://console.cloud.google.com/marketplace/config/bitnami-launchpad/mongodb).

For best performance, make sure your MongoDB instance is hosted on Google Cloud and located in the same region as your app.

By default, MongoDB uses port `27017` for communication. After setting up your MongoDB instance, you might need to configure the firewall rules to allow traffic to and from this port. For instances running on Google Cloud, see Using firewall rules (https://cloud.google.com/vpc/docs/using-firewalls) for instructions.

Warning: mLab's and Bitnami's defaults allow access to your MongoDB instance from anywhere. In production environments, create secure firewall rules and configure MongoDB to require specific IP addresses on login.

Install and enable MongoDB for PHP

1. To connect to MongoDB while running the Bookshelf app locally, install the PHP MongoDB extension (http://php.net/manual/en/mongodb.installation.php). If you're using Linux, install with PECL (https://pecl.php.net/).
2. Enable the extension in `php.ini`:

LINUX/MACOS WINDOWS

```
echo "extension=mongodb.so" >> `php --ini | grep "Loaded Configuration" | sed
```

3. Add the MongoDB library to Cloud Composer:

```
composer require "mongodb/mongodb:^1.0.0"
```

Configuring settings

1. 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 `mongodb`.
5. Set the values of `mongo_url`, `mongo_database`, and `mongo_collection` to the appropriate values for your MongoDB instance. For example:

```
mongo_url: mongodb://104.197.3.232:27017
mongo_database: getting_started_php
mongo_collection: books
```

6. Save and close `settings.yml`.

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.

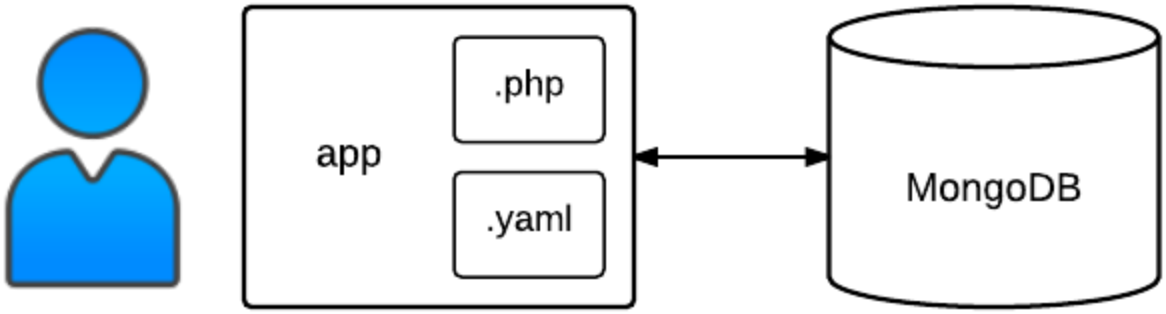
You can reduce costs by deleting the non-default versions of your app (https://cloud.google.com/php/getting-started/using-cloud-storage#delete_non-default_versions_of_your_app)

.

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

The following 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 `mongodb`. This setting tells the app to load the `MongoDb` class, which is defined in `src/DataModel/MongoDb.php`. The `MongoDb` class wraps the MongoDB API and stores books in your MongoDB database.

The following code in `controllers.php` defines and registers a handler for the `GET '/books'` route. The `$model` variable is an instance of the `MongoDb` class. The `$model->listBooks` method returns an array that contains an array of books and a cursor. Then the Twig template engine (http://twig.sensiolabs.org) 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's 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 %}
```

The following 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 extra 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/TUTORIAL-APP)

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

Last updated December 4, 2019.