Разработка многостраничного сайта на PHP

4.3 РАБОТА С КОНТЕЙНЕРАМИ И СИСТЕМОЙ СБОРКИ DOCKER
ЗАНЯТИЕ № 9 - ПРАКТИКА

Тема занятия - Работа с контейнерами и системой сборки Docker

Цель занятия –

Формирование умения разработки веб-приложения и перемещения его в контейнер

Содержание

Введение

Docker — это инструмент для автоматического развертывания и управления приложениями в среде контейнеризации. Он позволяет упаковывать приложения в контейнеры, запускаемые на любом ПК, на которой установлен Docker, что делает его портативным и гибким.

В ходе данного занятия мы рассмотрим 3 созданных нами приложения, которые после мы поместим в контейнеры.

Задача № 1

Создадим приложение под названием «Решай примеры».

Это простое веб-приложение предлагает ввести два числа и выбрать математическую смесь (+, -, *, /). При отправке приложения приложение вычисляет результат и выводит его на экран.

Решение

```html
<!DOCTYPE html>
<html>
<head>
    <title>Решай примеры</title>
    <meta charset="utf-8">
</head>
<body>
    <h1>Решай примеры</h1>
    <form action="index.php" method="post">
        <label for="num1">Первое число:</label>
        <input type="number" name="num1" id="num1" required>
        <br>
        <label for="num2">Второе число:</label>
        <input type="number" name="num2" id="num2" required>
        <br>
        <label for="operation">Операция:</label>
        <select name="operation" id="operation" required>
            <option value="+">Сложение</option>
            <option value="-">Вычитание</option>
            <option value="*">Умножение</option>
            <option value="/">Деление</option>
        </select>
        <br>
        <input type="submit" value="Решить">
    </form>
```

```php
    <?php
        if ($_SERVER['REQUEST_METHOD'] === 'POST') {
            $num1 = $_POST['num1'];
            $num2 = $_POST['num2'];
            $operation = $_POST['operation'];
            switch ($operation) {
                case '+':
                    $result = $num1 + $num2;
                    break;
                case '-':
                    $result = $num1 - $num2;
                    break;
                case '*':
                    $result = $num1 * $num2;
                    break;
                case '/':
                    $result = $num1 / $num2;
                    break;
            }
            echo '<h2>Результат: ' . $result . '</h2>';
        }
    ?>
</body>
</html>
```

Связь с Docker

Для того чтобы разместить веб-приложение на PHP в контейнере Docker, необходимо разрешить следующие шаги:

1. Создать Dockerfile для сборки образа. Указать базовый образ. Например, можно выбрать образ php:latest.
2. Добавить все необходимые файлы и папки в образ с помощью команды COPY. В случае необходимости необходимы файлы приложений.
3. Установить все зависимости и дополнения, необходимые для работы приложений. Например, можно установить расширение для работы с MySQL.
4. Открыть порт, который будет для доступа к приложению.
5. Запустить приложение при запуске контейнера с помощью команды CMD или ENTRYPOINT

Связь с Docker

Пример Dockerfile для веб-приложения на PHP может выглядеть так:

```
FROM php:latest

WORKDIR /app

COPY . /app

RUN apt-get update && \
    apt-get install -y libpq-dev && \
    docker-php-ext-install pdo pdo_mysql && \
    pecl install xdebug && \
    docker-php-ext-enable xdebug

EXPOSE 80

CMD ["php", "-S", "0.0.0.0:80"]
```

Связь с Docker

Далее, чтобы собрать образ контейнера, нужно запустить команду в терминале в каталоге с Dockerfile:

```
docker build -t my-php-app .
```

После этого можно использовать контейнер на основе созданного образа с помощью команды:

```
docker run -p 8080:80 my-php-app
```

Задача № 2

Создадим простую форму обратной связи, которая позволяет пользователям получать сообщения администратору сайта:

Решение

```php
<?php
// Проверяем, была ли отправлена форма
if ($_SERVER['REQUEST_METHOD'] === 'POST') {

  // Получаем данные из формы
$name = $_POST['name'];
$email = $_POST['email'];
$message = $_POST['message'];

  // Отправляем сообщение администратору сайта
$to = 'admin@example.com';
$subject = 'Новое сообщение от ' . $name;
$body = "Email: $email\nСообщение:\n$message";
$headers = "From: $email";
mail($to, $subject, $body, $headers);

  // Перенаправляем пользователя на страницу подтверждения
header('Location: thank-you.html');
exit;
}
?>
```

```html
<!DOCTYPE html>
<html>
<head>
    <title>Обратная связь</title>
</head>
<body>
    <h1>Обратная связь</h1>
    <form method="post">
    <label>
        Имя:
        <input type="text" name="name" required>
    </label>
    <br>
    <label>
        Email:
        <input type="email" name="email" required>
    </label>
    <br>
    <label>
        Сообщение:
        <textarea name="message" required></textarea>
    </label>
    <br>
    <button type="submit">Отправить</button>
    </form>
</body>
</html>
```

Решение

Это простое веб-приложение содержит HTML-форму, которая позволяет пользователям просматривать сообщения администратору сайта. Когда пользователь отправляет форму, PHP-скрипт получает данные из формы и отправляет их на заданный адрес электронной почты. Пользователь затем перенаправляется на страницу подтверждения.

Связь с Docker

Чтобы разместить это веб-приложение в контейнере Docker, вам необходимо будет создать файл Dockerfile, который определяет, как собирать образ контейнера. Например, Dockerfile может выглядеть так:

```
FROM php:7.4-apache

COPY . /var/www/html/
```

Связь с Docker

Этот Dockerfile использует официальный образ PHP с Apache в качестве базового образа. Затем он копирует все файлы из популярных каталогов в каталоге /var/www/html/ в контейнере.

Чтобы собрать образ контейнера, необходимо в каталоге с Dockerfile и выполнить команду:

```
docker build -t my-php-app .
```

Связь с Docker

Здесь мы используем команду docker build для сборки образа, которая будет называться my-php-app.

Далее запускаем контейнер:

```
docker run -p 80:80 my-php-app
```

Эта команда запускает контейнер, основанный на образе my-php-app, и связывает порт 80 внутри контейнера с портом 80.

Теперь наше веб-приложение на PHP готово к запуску в контейнере Docker. Мы можем использовать этот образ для развертывания приложений на предыдущем сервере, где установлен Docker.

Задача № 3

Создадим простой блог, написанный на языке программирования PHP.

Решение

```php
<?php

// Файл index.php

// Подключаем файл с функциями
require_once 'functions.php';

// Получаем список статей
$articles = getArticles();

?>

<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Мой блог</title>
</head>
<body>
    <h1>Мой блог</h1>
    <?php foreach ($articles as $article) : ?>
        <div>
            <h2><?= $article['title'] ?></h2>
            <p><?= $article['content'] ?></p>
        </div>
    <?php endforeach; ?>
</body>
</html>
```

```php
<?php
// Файл functions.php

// Функция для получения списка статей
function getArticles()
{
    // Подключаемся к базе данных
    $pdo = new PDO('mysql:host=localhost;dbname=blog', 'username', 'password');

    // Запрос на получение списка статей
    $query = 'SELECT * FROM articles';

    // Выполняем запрос
    $statement = $pdo->query($query);

    // Получаем список статей
    $articles = $statement->fetchAll(PDO::FETCH_ASSOC);

    // Возвращаем список статей
    return $articles;
}
?>
```

Решение

Это простой пример блога на PHP. Он использует базу данных MySQL для хранения статей и функцию getArticles() для получения списка статей. Вы можете дополнить приложение другими функциями и добавить новые статьи или возможность комментирования. Чтобы разместить это приложение в Docker, вам необходимо создать Dockerfile, где вы должны написать, как установить и настроить PHP и MySQL внутри контейнера, а также как получить код вашего контейнера.

Связь с Docker

Для размещения кода блога на PHP в Docker необходимо создать Dockerfile и определить в нем все необходимые зависимости, а также инструкции для сборки и запуска приложений.

Вот пример Dockerfile для нашего блога на PHP:

Связь с Docker

```
FROM php:7.4-apache

# Установка необходимых пакетов
RUN apt-get update && \
    apt-get install -y \
    libpq-dev \
    libzip-dev \
    unzip

# Установка расширений PHP
RUN docker-php-ext-install pdo pdo_pgsql zip

# Копирование исходного кода приложения в контейнер
COPY . /var/www/html/

# Установка прав на папки и файлы
RUN chown -R www-data:www-data /var/www/html && \
    chmod -R 755 /var/www/html

# Открытие порта для Apache
EXPOSE 80

# Запуск Apache
CMD ["apache2-foreground"]
```

Связь с Docker

Чтобы собрать образ Docker, необходимо реализовать команду docker buildи указать путь к каталогу с Dockerfile.

Например:

```
docker build -t myblog .
```

Здесь myblog - имя образуемого образа, а . - текущая директория, в которой находится Dockerfile.

Связь с Docker

После успешной сборки образа можно реализовать контейнер с помощью команды docker run.

Например:

```
docker run -p 80:80 -d myblog
```

Здесь -p 80:80- порты, чтобы веб-приложение было доступно по сценарию порта 80, а myblog- имя запускаемого образа.

Связь с Docker

Создадим файл docker-compose.yml:

```
version: '3'
services:
    web:
    build: .
    ports:
        - "8000:80"
    volumes:
        - .:/var/www/html
    environment:
        MYSQL_HOST: mysql
        MYSQL_DATABASE: blog
        MYSQL_USER: root
        MYSQL_PASSWORD: root
    mysql:
    image: mysql:5.7
    environment:
        MYSQL_DATABASE: blog
        MYSQL_USER: root
        MYSQL_PASSWORD: root
        MYSQL_ROOT_PASSWORD: root
```

А затем соберем все в контейнер:

```
docker-compose up -d
```

Связь с Docker

Теперь вы можете открыть веб-браузер и перейти по адресу http://localhost:8000 , чтобы увидеть свой блог на PHP в Docker-контейнере.

Обратите внимание, что мы использовали базу данных MySQL, которая также была запущена в контейнере. Если вы хотите использовать другую базу данных, вам нужно будет изменить время в файле docker-compose.yml.

Заключение

Docker является средством упрощения разработки, тестирования и развертывания приложений на PHP. Он позволяет упаковать приложение в контейнере, который можно запускать в любой среде, где установлен Docker. Это дает большую гибкость и скорость процесса разработки и развертывания приложений. Кроме того, мы рассмотрели несколько примеров создания Docker-образов для приложений на PHP, в том числе с использованием MySQL, а также использовали примеры простых и более сложных задач по тестированию приложений на PHP. Полагаю, эта информация поможет вам повысить ваши навыки в разработке и тестировании приложений на PHP с помощью Docker.

Рефлексия

Сегодня мы создали 3 веб-приложения, поместили их в контейнеры и проверили их работу.

Ответьте на вопросы:

1. Что такое Docker?
2. Зачем мы помещаем наши приложения в контейнеры?
3. Какое приложение понравилось больше всего?
4. Какое приложение не понравилось?
5. Какие у вас остались вопросы?


Спасибо
за
внимание