электронный научно-технический журнал
ИНЖЕНЕРНЫЙ ВЕСТНИК
Издатель: Общероссийская общественная организация "Академия инженерных наук им. А.М. Прохорова". Эл No. ФС77-51036. ISSN 2307-0595

Сравнение возможностей PHP и Python при создании информационных систем

 # 01, март 2018

Пивоварова Н. В.[1,*], Головко А. П.[1,]**

УДК: 004.432

[1]Россия, МГТУ им. Н.Э. Баумана
[*]pivovarova.natasha2013@yandex.ru
[**]Antonio.Golovko@gmail.com

Введение

В последнее время на форумах нередко возникают сообщения о “смерти” PHP, при этом в качестве альтернативы приводят популярный сегодня язык программирования Python. Такие заявления часто проистекают из-за использования разработчиками уже известных им языков для решения нетипичных (для этого языка) задач. Так, например, для Python были созданы модули для работы и разработки сетевых приложений.

Разумеется, языки создаются для разных целей, и сравнивать их, не ограничив область разработки, не имеет никакого смысла. Действительно, если мы будем использовать PHP для серверной разработки, которая подразумевает запуск приложений демонов, мы наткнемся на ряд трудностей, таких как утечка памяти в PHP. При создании этого языка не ставилась цель создавать такие приложения - после выполнения скрипт должен был “умирать”, а не работать в цикле, так что очистка памяти была не нужна. Заметим, что в современной версии языка появился сборщик мусора, наподобие тех, что используется в других высокоуровневых языках программирования, однако он не столь совершенен как в том же Python. Эта и другие проблемы, возникающие при решении задач для которых язык не приспособлен, делают его не конкурентно-способным в областях, отличных от той, для которой он создан, и сравнение языков оказывается не корректным.

Совсем другую картину мы видим, если речь идет о создании информационных систем, включающих с веб-приложения, поскольку именно для этих целей PHP и создавался. Целью работы является сравнение языков PHP и Python в данной области. Мы не рассматриваем фреймворки, а ограничимся лишь стандартными модулями, ведь наша цель - сравнить языки, а не фреймворки.

Итогом данной статьи не является прямое утверждение, что один язык лучше другого: в конечном счете, разработчик сам выбирает то, что ему ближе. Вместо этого в статье показаны преимущества одного из языков в конкретной области, а также его недостатки. В отличие от статей, говорящих о скорой «смерти» PHP, здесь показано, что язык по-

прежнему остается лидером в области разработки информационных систем и веб-приложений. Не зря на этом языке реализовано более 60% веб-ресурсов.

В первую очередь определяем границы, в которых проводится сравнение языков PHP и Python. Поскольку PHP создан для разработки веб-приложений, рассматриваем самую обширную по функциональности область – информационные системы, где необходима реализация большинства типовых задач веб-разработки. Из числа этих систем исключаем те, в которых неприемлемо использование языка PHP. Для четкого определения, какую часть работы выполняет скрипт, написанный на рассматриваемом языке, представляем принципы клиент-серверного взаимодействия в информационных системах. В завершении работы рассматриваем типичные задачи, возникающие при разработке информационных систем, и пытаемся ответить на вопрос, имеет ли какой-либо из рассматриваемых языков явные преимущества для выбора его в качестве инструмента разработки.

1. Программная архитектура информационной системы

Подавляющее большинство информационных систем имеет клиент серверную архитектуру. В общем случае программное обеспечение серверной части включает веб сервер, препроцессор одного или нескольких серверных языков (PHP, Python, Perl и пр.), собственно серверное приложение на соответствующем языке, SQL сервер и базу данных. Часто в состав серверного ПО включается фреймворк соответствующего языка. Поскольку целью настоящей статьи является сравнение языков, а не фреймворков, то возможности последних далее в статье не рассматриваются.

Клиентское ПО обязательно включает браузер и, возможно, клиентские приложения на языке JavaScript или ему подобных. Типовая высокоуровневая программная архитектура ИС представлена на рис.1.

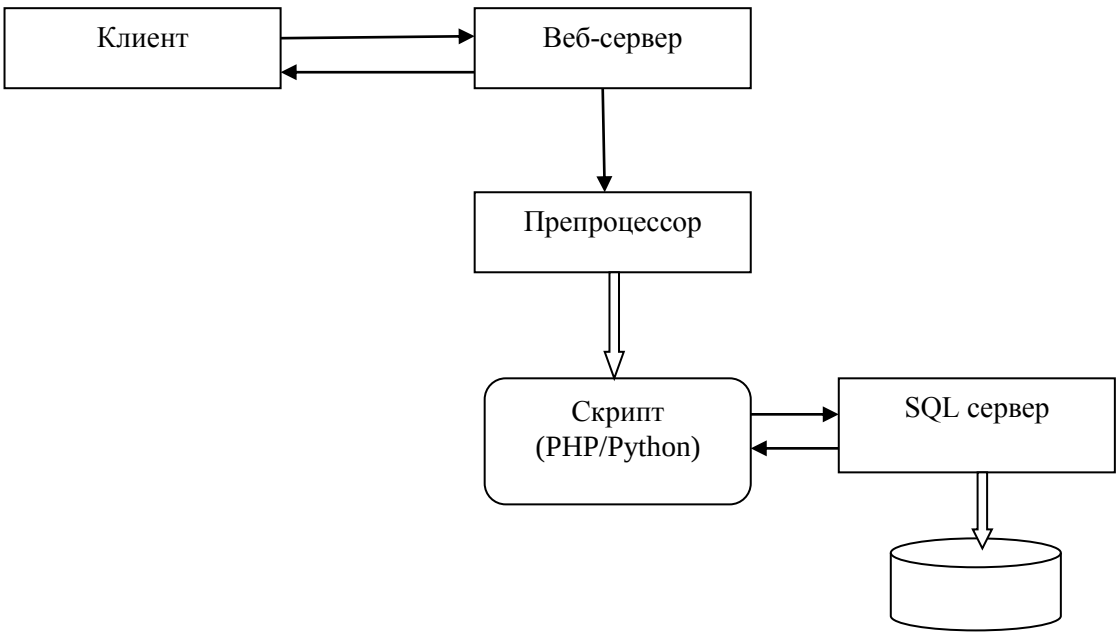

Рис. 1. Программная архитектура ИС

Обмен сообщениями и данными между клиентом и сервером происходит по протоколу HTTP. Клиент посылает серверу запрос на ресурс (URL, URI), а ответ получает либо готовую HTML страницу, либо данные, например, в формате json, которые должны быть вставлены в HTML страницу клиентским приложением. В настоящей статье за основу взят первый вариант, когда на запрос клиента сервер отсылает либо статическую HTML страницу, либо динамическую, сформированную работой препроцессоров языков PHP или Python. В этом варианте на клиенте достаточно иметь только браузер. А дополнительные клиентские приложения на JavaScript, позволяющие делать HTML страницы реагирующими на действия пользователей, являются опциональными.

В самом общем виде сеанс взаимодействия между клиентом и сервером по HTTP протоколу следующий. Клиент устанавливает соединение и отсылает на сервер запрос в формате URL и возможно необходимые параметры. Сервер находит или формирует ответную HTML страницу. Сервер возвращает пользователю HTML страницу и разрывает установленное соединение, «забывая» обо всех действиях, выполненных в ходе сеанса взаимодействия. Именно вследствие того, что протокол HTTP не сохраняет данных о выполненных действиях, его называют «протоколом без памяти». Поэтому задача сохранения необходимых данных о предыдущих сеансах взаимодействия ложится на серверное приложение и должна реализоваться встроенными средствами серверного языка.

Рассмотрим алгоритм сеанса взаимодействия между клиентом и сервером более подробно. Клиент устанавливает соединение с веб-сервером и отправляет ему запрос, содержащий URL и, возможно параметры запроса. Приняв запрос, веб-сервер совмещает путь директории веб-приложения/сайта из конфигурационного файла и указанный URL для определения пути к запрашиваемому ресурсу. Затем проверяет доступ и в зависимости от его типа определяет дальнейшие действия. Если запрошена статическая страница, то веб-сервер извлекает ее и отсылает клиенту. Если требуется сформировать ответную страницу динамически, то веб-сервер по полученному URL находит скрипт (в нашем случае на PHP или Python) и передает управление соответствующему препроцессору.

Для вызова и выполнения пользовательских скриптов разработан стандарт CGI и соответствующий модуль Apache mod_cgi. CGI является шлюзом между веб-сервером и исполняемыми скриптами. Через этот шлюз скриптам передаются данные от сервера (информация о запросе, метод передачи данных, порт и т.д.), которые называются окружением. После выполнения скрипта результат его работы считывается через стандартный поток вывода (stdout), а все ошибки - через стандартный поток ошибок (stderr). Схема работы CGI приведена на рис. 2.

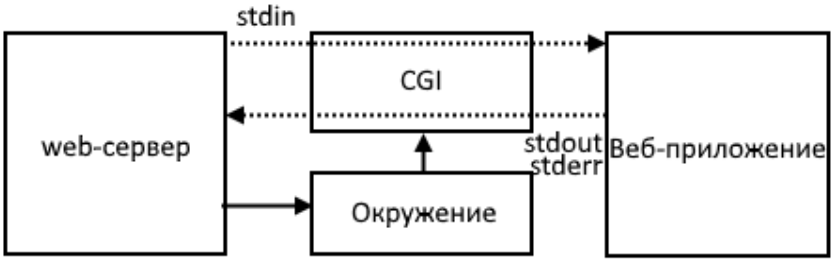

Рис. 2. Схема работы CGI

Другой популярный модуль mod_php появился в результате развития языка PHP. Если CGI позволяет работать практически с любыми скриптовыми языками, то mod_php специализирован под PHP, в результате чего предоставляет удобный интерфейс для работы PHP скриптов. Отметим также стандарты FCGI и WSGI. Первый стандарт служит для ускорения работы обычного CGI, а второй является стандартом для python скриптов. WSGI был разработан на основе стандарта CGI, в первую очередь, из-за наличия большого числа различных фреймворков Python. WSGI предоставляет единый интерфейс между различными фреймворками и веб-серверами. Указанные стандарты передают данные скриптам через окружение, которое служит для передачи всей необходимой информации о запросе от сервера исполняемому скрипту. Обычно эти данные передаются пользователем через формы на веб-страницах и используются для авторизации, поиска в БД, занесения данных, редактирования и т.д.

Препроцессор выполняет полученный скрипт, формируя выходную HTML страницу. По ходу выполнения скрипт может устанавливать соединение с SQL сервером для получения необходимых данных из базы данных. По окончании работы скрипта созданная HTML страница возвращается веб-серверу для передачи клиенту.

После выполнения пунктов 3 или 4-5, веб-сервер считает сеанс взаимодействия с клиентом завершенным и разрывает установленное соединение.

2. Типовые задачи при реализации информационных систем

Сформулируем типовые задачи, которые должны быть выполнены серверным приложением в ходе формирования выходной HTML страницы.

Как уже отмечалось, в управляющем скрипте придется реализовать сохранение данных из предыдущих сеансов взаимодействия между клиентом и сервером. Типичной задачей, требующей данных из предыдущих сеансов, является задача авторизации пользователя. Действительно, если пользователь при вызове приложения ввел логин и пароль и успешно прошел авторизацию, то эти параметры должны быть сохранены в приложении для использования при переходе пользователя с одной страницы на другую.

Кроме того, типичной для ИС является работа с “корзиной”, которая используется для выбора некоторого числа объектов с возможностью их последующего редактирования, которое также потребует запоминания ранее сделанного выбора.

Очевидно также, что приложению необходимо будет устанавливать соединение с SQL сервером для доступа в базу с целью извлечения и редактирования данных.

Таким образом, к числу рассматриваемых типовых задач отнесем следующие.

1. Сохранение данных от предыдущих сеансов взаимодействия на примере задачи авторизации пользователя.
2. Соединение с SQL сервером.
3. Выполнение запросов к базе данных и вставка полученных данных в HTML страницу.
4. Редактирование данных в базе из интерфейса конечного пользователя.

3. Реализация типовых задач на языках PHP и Python

История PHP. PHP был разработан датским программистом Расмусом Лердорфом в 1995 году, Изначально PHP представлял собой набор скриптов на Perl и служил для обработки шаблонов HTML документов. Впоследствии Лердорф разработал с использованием языка C новый интерпретатор шаблонов PHP/FI, который включал в себя базовую функциональность современного PHP. В 1997 году вышла вторая версия PHP, также написанная на C - PHP/FI 2.0. В том же году два разработчика Энди Гутманс и Зеев Сураски, объединив силы с Расмусом, начали разработку PHP 3.0. Важнейшей частью нового языка стала возможность расширения его ядра дополнительными модулями, что впоследствии позволило PHP работать с большим числом баз данных и поддерживать многие API. Вскоре была разработана версия PHP 4, а позднее - и PHP 5, привнесшие большое число улучшений и нововведений.

История Python. Разработка Python была начата в 1989 году Гвидо ван Россумом. Python – язык широкого применения, активно используется для создания разных типов приложений. Автор языка по сей день принимает решения по дальнейшему развитию языка. В 2000 году была выпущена версия Python 2.0, включающая в себя такие функции как “сборщик мусора” и поддержка Unicode.

В 2008 году была выпущена обратно-несовместимая версия Python 3.0, функции которой были включены в версии 2.6 и 2.7. Таким образом, в настоящий момент существует две ветки развития Python, версии 2 и 3, обе версии продолжают свое развитие. Хоть версия 3.0 и не имеет обратной совместимости, основана она на тех же принципах, что и Python 2.0.

3.1. Взаимодействие с сервером

PHP. Как сказано выше, обычно сервер вызывает на исполнение скрипты, используя стандарт CGI. Однако для такого популярного языка как PHP в Apache существует специальный модуль mod_php. Хотя работать с PHP можно используя CGI, мы не рассматриваем эту возможность (хотя свои плюсы есть и у этого подхода). Вместо этого мы разберем наиболее распространенный вариант взаимодействия скрипта с сервером, основывающийся на модуле mod_php.

В первую очередь, mod_php позволяет не задумываться о составлении заголовках http-ответа, таких как код ответа и Content-Type, поскольку эти поля заполняются автоматически, хотя пользователь может влиять на их формирования, если это необходимо. Таким образом, данный модуль позволяет работать с сервером, не задумываясь о его устройстве. Главным недостатком данного модуля является то, что выполнение скрипта начинается каждый раз с самого начала - при каждом пришедшем запросе выполнение кода начинается заново, что понижает производительность сервера.

Модуль mod_php предоставляет окружение, в котором передается вся информация о запросе. В PHP доступ к этому окружению можно получить, используя глобальные ассоциированные массивы. Например, массив $_GET позволяет использовать данные, полу-

ченные от пользователя методом GET, а $_POST - данные полученные методом POST. Для доступа ко всем данным, предоставленным сервером, используется массив $_SERVER.

Python. Для Python, как и для PHP, существует специализированный модуль mod_wsgi. Главным отличием и плюсом использования стандарта WSGI, является возможность запуска веб-приложения (ИС) в фоне, что значительно ускоряет выполнение очередного запроса, так как исполняемые файлы уже находятся в памяти. Как и PHP, так и Python можно запускать в режиме стандарта CGI, однако mod_wsgi для Python является более практичным решением, как и mod_php для PHP.

Так как python разрабатывался как язык общего назначения, а не узкоспециализированный серверный, в нем нет реализаций таких механизмов как сессии, переменные окружения и т.д. Для доступа к окружению используются различные модули, которые предоставляют полноценный функционал для взаимодействия приложения с сервером. Самый распространенный модуль – это cgi. Для доступа к данным, полученным при помощи методов GET и POST, можно следующим образом:

```
import cgi
form = cgi.FieldStorage()
print form["login"]
```

Этот скрипт выведет данные по ключу login, переданные пользователем.

3.2. Взаимодействие с БД

PHP. Поскольку PHP разрабатывался как серверный язык, взаимодействие с БД в нем максимально простое, нет необходимости использовать сторонние библиотеки или фреймворки в PHP имеются функции для работы с большинством СУБД. Рассмотрим взаимодействие с СУБД на примере MySQL.

Сервер базы данных может располагаться, как и на той же машине что и веб-сервер, так и на удаленной машине, однако это не влияет на способ подключения приложения к СУБД. Самый простой способ соединиться с MySQL – воспользоваться функцией mysql_connect:

```
$link = mysql_connect('mysql_host', 'mysql_user', 'mysql_password')
   or die('Не удалось соединиться: ' . mysql_error());
echo 'Соединение успешно установлено';
```

Данную функцию предоставляет расширение PHP для MySQL, которое позволяет взаимодействовать приложению с MySQL. Однако у такого подхода есть ряд недостатков, самый очевидный из которых - невозможность сменить СУБД (например, MySQL на Oracle), незначительно меняя при этом исходный код. В PHP 7.0 возможности использовать это расширение, вовсе нет. Вместо этого разработан универсальный метод, заключающийся в использовании PHP Data Objects (PDO).

Ниже представлен пример подключения приложения к MySQL, используя PDO.

```
try{
        $pdo=new PDO ("mysql:host=localhost;dbname=$db",$login,$password);
```

```
        $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
}
catch(pdoexception $e){
        echo "Error connecting to database ".$e->GetMessage();
        exit();
}
```

PDO предоставляет прослойку, которая позволяет не заботиться об устройстве СУБД. Так, например, для подключения приложения к Oracle вместо MySQL достаточно заменить mysql на oci при создании нового объекта PDO.

Получив объект PDO, мы можем напрямую работать с СУБД. Формируя sql запрос, мы способны не только запрашивать данные, но и заносить, изменять, добавлять таблицы, в общем, все то, что могли бы делать напрямую в СУБД. Разумеется, в СУБД предусмотрены права доступа для разных пользователей, но об этом позже. Пока предположим, что у пользователя есть права для выполнения всех описанных выше операций.

Python. Для работы с различными СУБД для Python существуют соответствующие модули. Рассмотрим на примере MySQL. Используя модуль mysql, можно работать с СУБД практически также просто, как и в PHP. Рассмотрим следующий скрипт для подключения приложения к БД.

```
import mysql.connector
config = {
      'user': 'Guest',
      'password': 'Guest',
      'host': 'localhost',
      'database': 'example',
   }
try:
      conn = mysql.connector.connect(**config)
      if conn.is_connected():
         return conn

except Error as e:
      print(e)
      return -1
```

В этом алгоритме, используется словарь config, содержащий необходимые данные для подключения к СУБД.

Самым очевидным минусом при работе с СУБД в Python является отсутствие механизма подобного PDO. В случае необходимости использовать другую СУБД придется переписывать значительную часть кода.

PHP. Итак, у нас есть объект PDO. Как при помощи него занести данные в БД, к которой мы подключены? Предположим, у нас есть некая таблица USERS с полями ID, Login, Password. Для занесения данных воспользуемся следующим алгоритмом (подразумевается, что объект PDO сформирован ранее):

```
<?php
$query = ("INSERT INTO USERS (login, password) VALUES (:login,:password)");
$stmt = $pdo->prepare($query);
$stmt->bindParam(':login', $login);
```

```
$stmt->bindParam(':password', $password);
$login = "login1"
$password = "qwerty1"
$stmt->execute();
?>
```

Разберем скрипт подробнее: метод prepare() объекта PDO позволяет подготовить к выполнению SQL запрос, при этом вместо реальных данных используются псевдопеременные :login и :password. На настоящие переменные заменяются при помощи метода bindParam(). Такая последовательность используется в целях безопасности, для предотвращения SQL инъекций. Выполняется запрос методом execute(). Такой способ удобен для выполнения одинаковых SQL запросов как в данном случае.

Python. Алгоритм занесения данных в Python аналогичен, за исключением, как уже упоминалось, работы с PDO.

```
cursor = conn.cursor()
query = ("INSERT INTO USERS (login, password) VALUES (%s, %s)");
login = "login1"
password = "qwerty1"
cursor.execute(query,(login,password))
```

Редактирование и удаление данных из БД происходит аналогичным образом. Изменения касаются в основном лишь SQL-запроса. Можно сделать вывод, что работа с БД практически одинакова для обоих языков.

3.3. Авторизация

Прежде чем перейти к реализации самой авторизации, необходимо разобраться с куками и механизмом сессий, и для чего они нужны.

Протокол HTTP, используемый для передачи данных, не имеет памяти. Тем не менее, часто необходимо сохранять данные о пользователе и его действиях на сайте. Реализовать это можно при помощи куков, которые сохраняются на стороне клиента и передаются при запросах на сервер. Куки содержат некоторую информацию о пользователе, например, его уникальный идентификатор, или данные о его действиях на веб-сайте. Они содержат данные в виде пары “ключ-значение”. Эта пара создается веб-сервером с помощью заголовка set cookie и затем посылается клиенту.

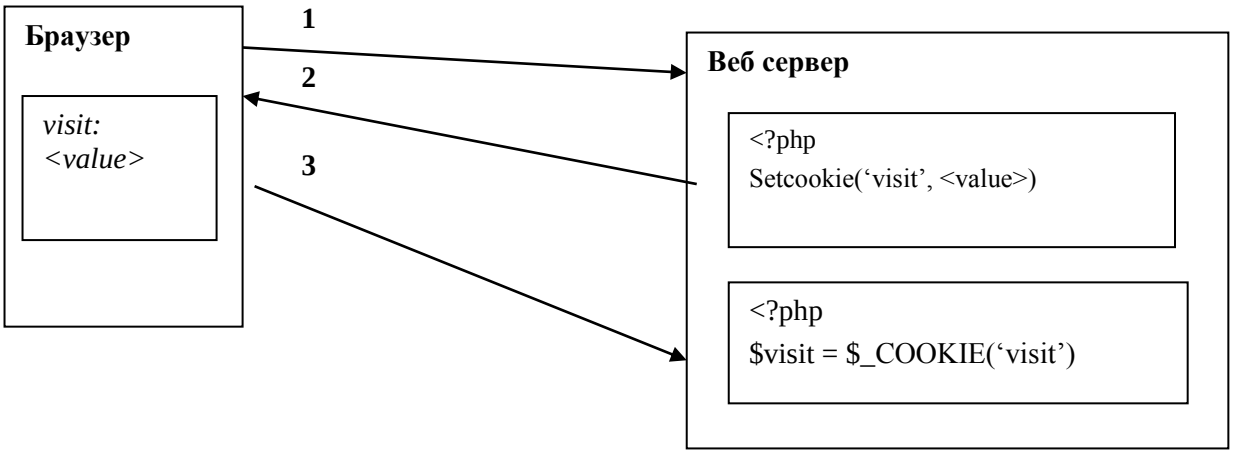

Рис. 3. Взаимодействие между клиентом и сервером при помощи куки

Пример использования cookie:

PHP

```
<?php
if (!isset($_COKKIE['visits']))
{
        $_COOKIE['visits']=0;
}
$visits=$_COOKIE['visits'] +1;
setcookie('visits',$visits);
include 'welcome.html.php';
?>
```

Welcome.html.php

```
<?php
 if ($visits >1)
 {echo " Hello for $visits visits"; }
else
  {   echo "Hello for the first visit!!!";}   ?>
```

Python

```
import http.cookies
cookie = http.cookies.SimpleCookie(os.environ.get("HTTP_COOKIE"))
visits = cookie.get("visits ")
if visits is None:
        visits =0
visits = visits + 1
print("Set-cookie: visits="+visits)
print("Content-type: text/html\n")
import welcome
```

welcome.py

```
if (visits>1)
        print(“Hello for”+visits+”visits”)
else
        print(“Hello for the first visit!!!”)
```

При помощи куки можно реализовать авторизацию. Для этого можно сохранять в них данные пользователя, однако не трудно догадаться, что это не безопасно, так как эти личные данные будут не только хранится в явном виде на компьютере пользователя, но и постоянно передаваться по сети, при каждом запросе на сервер. Для решения этой проблемы в PHP был придуман механизм сессий.

PHP. Сессии позволяют сохранять данные о конкретном пользователе и его действиях. В PHP данный механизм встроенный, для старта сессии необходимо воспользоваться функцией session_start(). Рассмотрим случай, когда сессия устанавливается впервые (первое взаимодействие клиента с сервером), при вызове функции в заголовки http-ответа будет добавлен set-cookie, в поле которого будет значиться уникальный идентификатор PHPSESSID. С этого момента в глобальный ассоциативный массив $_SESSION могут быть занесены любые данные, которые будут сопоставлены с PHPSESSID. Этот ID будет передаваться от браузера серверу при каждом дальнейшем взаимодействии с клиентом,

пока не будет удален. При повторном обращении к ресурсу функция session_start() обнаружит существование PHPSESSID и восстановит всю информацию в массив $_SESSION.

В процессе авторизации этот механизм активно используется. Однако перед тем как выдавать пользователю идентификатор, необходимо удостовериться, что он имеет доступ к запрашиваемому ресурсу. Обычно логин и пароль пользователей хранится в БД имеющей непосредственное отношение к данной ИС. Допустим, что такие данные хранятся в некоторой таблице USERS, хранящейся в БД ресурса. В таблице содержится информация о логине и пароле пользователей.

Ниже приведен простейший скрипт реализации механизма авторизации пользователя.

```
<?php
session_start();

$login=$_POST['login'];
$password=$_POST['password'];

$_SESSION['login']=$login;

$sql = "SELECT login FROM users WHERE login=:login and password=:password";
$conn = $pdo->prepare($sql);
$conn->bindValue(':login', $login);
$conn->bindValue(':password', $password);
$conn->execute();
$row=$conn ->fetch();
 if ($row)
{
 echo "Authorization is successful";
 }
  else
  {
echo "User Name Or Password Invalid!";
  }
}
?>
```

Данные пользователя, переданные от клиента на сервер методом POST, используются для формирования SQL запроса. Индикатором существования пользователя в таком случае является наличие данных в переменной $row (если такого пользователя в БД нет, то $row равняется 0), что и проверяется далее.

При создании реальных ИС подобные таблицы пользователей, разумеется, не используются. Данные о пользователях хранятся в зашифрованном виде, а также используется более безопасный алгоритм авторизации.

В любом случае при дальнейшей работе с ресурсом необходимо проверять авторизован ли пользователь. Сделать это можно проверив существование данных в массиве $_SESSION, если пользователь не авторизовался ранее, то и данные восстановлены не будут. Таким образом, в начале каждого скрипта можно добавить строки проверки авторизации. Если пользователь не авторизован, обычно производится перенаправление на стра-

ницу авторизации, при помощи функции header(), в которой устанавливается значение 'Location: ***URL***' (где URL – адрес страницы авторизации). Эта функция позволяет устанавливать заголовки http-ответа, в данном случае заголовок Location.

Python. С авторизацией в Python обстоит все значительно сложнее. Хотя в Python нет проблем с формированием и передачи cookie, тем не менее тут нет встроенного механизма сессий. Соответственно есть только два пути для организации авторизации. Простейшим является использовать куки для передачи информации о пользователе. Такой способ весьма плох с точки зрения безопасности, так как данные пользователя будут передаваться по сети при каждом запросе. Можно организовать механизм похожий на тот, что реализован в PHP своими силами, но в таком случае проще воспользоваться фреймворками. Однако фреймворки выходят за пределы темы статьи, так что их мы рассматривать не будем.

3.4. Формирование динамической HTML страницы

Если при работе с БД в основном сравниваются библиотеки, нежели сами языки, то при формировании страниц используются средства языка. Сравнивая скрипты, написанные на этих языках, можно обратить внимание на их схожесть, так, например, для вывода данных из БД в виде таблицы (последний скрипт), используется один и тот же алгоритм, и его реализации на этих языках очень похожи. Однако есть и различия, о которых будет сказано далее. Но в начале пару слов о том, что такое HTML страница.

HTML интерпретируется браузером на стороне клиента и представляется в удобочитаемом виде. Он содержит множество тегов служащих для разметки и структурирования информации. Для реализации простейшей информационной системы нам понадобится лишь несколько из них. Подробно HTML разметка разбираться не будет, все примеры содержат простейшие элементы, которые будут при необходимости прокомментированы.

Пользователь, взаимодействуя с информационной системой, постоянно запрашивает данные, которые по факту хранятся в БД. Для предоставления этих данных в виде HTML, используются скрипты, которые формируют на основе шаблонов полноценные страницы.

PHP. При помощи скриптов на языке PHP мы можем формировать ответ пользователю в виде HTML разметки. Самым простым способом сформировать HTML страницу используя PHP представлен ниже.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>HTML Document</title>
  </head>
  <body>
    <?php echo "Вывод текста или <b>тегов</b>"; ?>
  </body>
</html>
```

Первый тег DOCTYPE определяет тип документа для корректного отображение его в браузере, исполняемый PHP код заключен в тегах “<?php” и “ ?>”. Код выполняется ин-

терпретатором, который автоматический пишет все, что не заключено в теги “ <?php ?>” в стандартный поток вывода и исполняет весь код, написанный внутри. Это полезно для составления шаблонов страниц, заготовок на которых находятся все повторяющиеся и независимые от данных элементы, например, шапка страницы. Оператор echo отвечает за вывод в стандартный поток вывода, с его помощью можно также выводить данные и тэги. Результатом выполнения такого PHP кода, будет следующая HTML разметка.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>HTML Document</title>
  </head>
  <body>
        Вывод текста или <b>тегов</b>
  </body>
</html>
```

В PHP есть возможность подключать другие PHP скрипты, и даже простые текстовые файлы или шаблоны элементов страниц. Сделать это можно при помощи выражения include. Текстовые файлы или html разметка в таком случае будет просто выбрасываться в стандартный поток вывода.

```
include "script.php";
include "template.html";
```

Используя эти механизмы можно формировать страницы с выводом данных из БД, как это показано ниже.

Скрипт functions.php

```
<?php
function print_t ($result){
        $result->execute();
        $row=$result->fetch();
        if(!$row) echo "No such data";
        else {
                echo "<table><tr>";
                $keys=array_keys($row);
                $i=0;
                while ($i<-count($row)){
                        echo "<th><strong>".$keys[$i]. "</strong></th>";
                        $i=$i+2;
                }
                echo "</tr>";
                do {
                        echo "<tr>";
                        for ($i=0;$i<=count($row)/2;$i=$i+1){
                                echo "<td>".$row[$i]. "</td>";
                        }
                        echo "</tr>";
                }
```

```
                while ($row=$result->fetch());
                echo "</table>";
        }
}
?>
```

Основной скрипт index.php

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>HTML Document</title>
  </head>
  <body>
    <?php
        include "functions.php";
$stmt = $pdo->prepare(" SELECT * FROM USERS");
        print_t($stmt);
    ?>
  </body>
</html>
```

Результатом исполнения скрипта index.php будет являться страница содержащую полную таблицу USERS.

В результате развития языка в PHP появилась конструкция foreach, которую удобно использовать для вывода данных из БД. Используя эту конструкцию можно модифицировать скрипт выше (участок с формированием колонок таблицы):

```
foreach ($row as $col){
                echo "<td>".$col. "</td>";
    }
```

Конструкция появилась как результат развития языка в области ИС, и является удобным и универсальным средством для работы не только с массивами из БД, но и любыми другими массивами.

Python. Как описано выше, в PHP используется удобный механизм, скрипт при необходимости отделяется от HTML разметки, в результате чего удобно писать шаблоны страниц. В Python нет аналогичного механизма, вывод данных в стандартный поток вывода осуществляется при помощи стандартных функций вывода. С другой стороны, это ограничение позволяет не захламлять код HTML разметкой. Работа с шаблонами в Python осуществляется как с обычными текстовыми файлами. Кроме того, так как HTML код является XML-образным, можно использовать многочисленные модули для работы с XML разметкой, что позволяет организовать удобную работу с шаблонами. Выше в статье был представлен код на PHP для формирования HTML таблицы, для сравнения представим тот же код на языке Python.

Таблица 1. Сравнение кода на языке PHP и Python

PHP	Python
function print_t ($result){* *$result->execute(); *$row = $result->fetch();* *if(!row) echo "No such data";* *if($row){* *echo "<table><tr>";* *$keys=array_keys($row);* *$i=0;* *while ($i<=count($row)){* *echo "<th>";* *echo $keys[$i];* *echo "</th>";* *$i=i+2;* *}* *echo "</tr>";* *do{* *echo "<tr>";* *for($i=0; $i<=count($row)/2; $i=$i+1){* *echo "<td>".$row[$i]. "</td>";* *}* *echo "</tr>"* *}* *while ($row=$result->fetch());* *echo "</table>";* *}* *}*	*def table_print(SQL, cursor):* *cursor.execute(SQL)* *row = cursor.fetchone()* *table ="<table>"* *if row is not None:* *num_fields = len(cursor.description)* *field_names = [i[0] for i in cursor.description]* *i = 0* *table = table + "<tr>"* *while i < num_fields:* *table = table + "<th>"+ * *field_names[i]+"</th>"* *i=i+1* *table = table + "</tr>"* *while row is not None:* *table = table + "<tr>"* *i=0* *while i < len(row):* *table = table + "<td>"* *table = table + str(row[i])* *table = table + "</td>"* *i=i+1* *row = cursor.fetchone()* *table = table + "</tr>"* *table = table + "</table>"* *return table*

Можно обратить внимание на то, что это один и тот же алгоритм, единственным отличием в Python является занесение сформированных строк в переменную, а не вывод. Сделано это для возможности дальнейшей обработки, например, занесения в шаблон методами модулей для работы с XML.

В python нет конструкции foreach как в PHP, но аналогично здесь работает цикл for:

for col in row

table = table + "<td>" + col + "</td>";

Таким образом, если говорить о сравнении языков при формировании страниц, можно отметить, что языки похожи. Основное отличие python от PHP в этом случае является отсутствие возможности писать код прямо в html шаблоне.

Заключение

Информационные системы были и остаются значимой областью разработки. В первую очередь они необходимы для обеспечения удобной работы с информацией. Для создания таких систем PHP подходит как нельзя лучше. В нем реализованы необходимые механизмы, как например механизм сессий, с его помощью можно с легкостью формировать HTML страницы и обрабатывать данные. Python не показывает той же простоты использования в данной сфере. Это обусловлено тем, что PHP является специализированным языком для разработки в этой области, однако и Python имеет свои преимущества.

Благодаря тому, что Python является языком общего назначения, для него реализовано множество библиотек, используемых для самых разнообразных целей. Однако сравнивая библиотеки Python и PHP, можно заметить лишь их схожесть, так как реализованы они для одних и тех же целей. Если же проводить сравнение непосредственно самих языков в области создания ИС, можно сделать вывод, что и тут языки весьма похожи, но у PHP имеется преимущество – возможность писать код непосредственно в шаблонах, что, несомненно, облегчает разработку.

Если для вас при создании ИС нужны скорость и простота разработки, то PHP отличный выбор, а заявления о том, что PHP “умер” вовсе неуместны. Плюсы Python выявляются при использовании фреймворков и его многочисленных модулей, но в таком случае и сравнивать необходимо фреймворки. И все же окончательный выбор языка разработки остается за разработчиком, какие бы плюсы не были у того или иного языка, лучше всего работать с тем с чем привычнее и удобнее.

Список литературы

[1]. Apache.RU / Документация [Электронный ресурс]. Режим доступа: http://www.apache.ru/docs/ (дата обращения: 3.03.2018)

[2]. Таненбаум Э. Компьютерные сети. 4-е изд. – СПб.: Питер, 2003. – 992 с .: ил.

[3]. Функционирование HTTP-сервера [Электронный ресурс]. Режим доступа: http://apachedev.ru/2006/03/12/the-apache-modeling-project-glava-2-chast-1/ (дата обращения: 3.03.2018)

[4]. RFC 2068. Hypertext Transfer Protocol [Электронный ресурс]. – Режим доступа: http://lib.ru/WEBMASTER/rfc2068/rfc2068.txt (дата обращения: 3.03.2018)

[5]. Руководство по PHP [Электронный ресурс]. – Режим доступа: http://php.net/manual/ru/index.php (дата обращения: 3.03.2018)

[6]. Лутц М. Программирование на Python. 4-е издание. Том I. – Пер. с англ. – СПб.: Символ-Плюс, 2011. – 992 с.

[7]. Лутц М. Программирование на Python. 4-е издание. Том II. – Пер. с англ. – СПб.: Символ-Плюс, 2011. – 992 с.

[8]. Прохоренок Н.А., Дронов В.А. HTML, JavaScript, PHP и MySQL. Джентльменский набор Web-мастера. 3-е изд., перераб. и доп. СПб.: БХВ-Петербург, 2010. 912 с.: ил.

[9]. Кевин Янк. PHP и MySQL. От новичка к профессионалу. Простой способ создать сайт на основе базы данных. 5-е изд. – М.: Эксмо, 2013. – 384 с.

[10]. Марк Вандшнайдер. Основы разработки веб-приложений с помощью PHP и MySQL: полное руководство. Москва: ЭКОМ, 2008. - 828 с.: ил.