Министерство образования и науки Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования «Петрозаводский государственный университет» Физико-технический институт Кафедра информационно-измерительных систем и физической электроники

ПРАКТИЧЕСКАЯ РАБОТА

ОТЧЁТ

по предмету «Языки описания сценариев»

Программа для оценки читаемости сложных html-страниц

Автор работы: студент группы 21612 Н. Ю. Новохатько \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2020 г.

Научный руководитель: доцент А.В. Соловьёв

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2020 г.

Петрозаводск 2020

**Содержание**

[Введение 3](#_Toc55409501)

[Постановка задачи 4](#_Toc55409502)

[Описание объёма выполненной работы 5](#_Toc55409503)

[Описание алгоритма и методов решения 7](#_Toc55409504)

[Руководство пользователя программы 21](#_Toc55409505)

[Отчёт о тестировании 23](#_Toc55409506)

[Заключение 28](#_Toc55409507)

[Исходный код 29](#_Toc55409508)

**Введение**

В данной работе будет рассмотрена программа, написанная для реализации задачи оценки читаемости сложных html-страниц. Разработка данной программы позволит осуществлять автоматизированную проверку большого количества страниц html и htmls и выявлять области, констраст между цветами текста и фона которых недостаточен для чтения без усилий для глаз.

Перед выполнением работы нами была поставлена цель:

Создание программы, осуществляющей автоматизированную проверку читаемости html-страниц.

Исходя из цели, нами были поставлены задачи:

1. Создание страниц с формой загрузки zip-архива и таблицей с количество загруженных архивов, со списком найденных страниц для архива, со списком найденных тегов для каждой страницы.
2. Создание контроллеров и методов для каждой из отображаемых страниц .
3. Создание необходимых таблиц и моделей к ним.
4. Создание метода, позволяющего найти на страницах ссылки на файлы css и, разобрав эти файлы, найти селекторы с заполненными полями для текста и фона и сравнить их, чтобы вычислить контрастность.

**Постановка задачи**

Дизайн и верстка сайта являются одной из задач при его создании. Их могут делать как один специалист, так и группа дизайнеров и верстальщиков, которые работают удалённо друг от друга. Результат работы – достаточно большое количество страниц, получает заказчик, у которого может возникнуть потребность дополнительно проверить html-страницы на соответствие общим требованиям.

Одним из требований является читаемость текста html-страниц. Строго говоря, это важное требование, которое предъявляется ко всем страницам, так как читаемость влияет на нагрузку, которая оказывается на глаза пользователей, что, в свою очередь, косвенно влияет на посещаемость сайта.

Для автоматизации проверки читаемости сложных веб-страниц требуется создать веб-приложение, которое позволит выполнять следующие задачи.

1. Загрузка архива zip на сервер.
2. Распаковка архива zip.
3. Отображение архива на html-странице с возможностью перехода к содержимому архива.
4. Поиск html и htmls страниц в архиве и отображение их списка на странице.
5. Поиск css файлов, связанных с каждой html страницей.
6. Обнаружение в каждом css файле селекторов с заполненными полями color – цвет текста в данной области страницы, и background-color – цвет фона в данной области страницы.
7. Сравнение цвета текста и фона для каждого селектора, определение разницы между ними и выявление соответствия относительно критериев достаточности контрастности. По стандарту WCAG 2.0:

|  |  |  |
| --- | --- | --- |
| Соотношение | | Пояснение |
| 4,5 | 1 | Минимальная контрастность для пользователей с остротой зрения 20/40 и потерей контрастной чувствительности (уровень АА). Так же применим для обычного текста. |
| 3 | 1 | Минимальная контрастность для обычных наблюдателей (уровень А). Применяется так же для заголовков. |
| 7 | 1 | Минимальная контрастность для пользователей с остротой зрения 20/80 и потерей контрастной чувствительности (уровень ААA). Так же применим для обычного текста и заголовков. |

1. Отображение таблицы для каждой html-страницы, состоящей из следующих столбцов: наименования исследуемых селекторов, цвет фона и цвет текста, флаги наличия минимальной требуемой контрастности для уровней A, AA, AAA.

**Описание объёма выполненной работы**

Объём выполненных работ:

1. Установлены и настроены: веб-сервер Apache, база данных MySQL, интерпретатор PHP, скрипт phpmyadmin для работы с базой данных.
2. Загружен и настроен фреймворк Yii2.
3. Загружен и подключен скрипт html-dom-parser.php для работы со сложными веб-страницами.
4. Загружен и подключен скрипт css-parser.php для работы с css. Для исправления ошибок работы код парсера понадобилось немного скорректировать.
5. В соответствии с моделью Model-View-Controller создан контроллер AddressarchivesController, включающий логику для распаковки архива, поиска веб-страниц, поиска ссылок внутри них на файлы css, обнаружение селекторов с заполненными полями color и background-color, и сравнения этих полей.
6. В соответствии с моделью Model-View-Controller созданы виды, генерирующие веб-страницу после работы контроллера Addressarchives – index, view.
7. В соответствии с моделью Model-View-Controller создан контроллер AddresspagesController, включающий логику для отображения таблицы со страницами заданного архива.
8. В соответствии с моделью Model-View-Controller создан вид, генерирующий веб-страницу после работы контроллера Addresspages – index.
9. В соответствии с моделью Model-View-Controller создан контроллер AddresspageviewController, включающий логику для отображения таблицы с исследованными селекторами заданной страницы из архива.
10. В соответствии с моделью Model-View-Controller создан вид, генерирующий веб-страницу после работы контроллера Addresspageview – index.
11. Создана база данных htmldatabase, в которой созданы таблицы addressarchives - список архивов, htmlpages – список страниц из архивов, textbackgroundclasses – список исследованных селекторов.
12. В соответствии с моделью Model-View-Controller создана модель Addressarchives для доступа к таблице addressarchives базы данных htmldatabase.
13. В соответствии с моделью Model-View-Controller создана модель Htmlpages для доступа к таблице htmlpages базы данных htmldatabase.
14. В соответствии с моделью Model-View-Controller создана модель Textbackgroundcolors для доступа к таблице textbackgroundcolors базы данных htmldatabase.
15. В соответствии с моделью Model-View-Controller создана модель Textbackgroundclasses для доступа к таблице textbackgroundclasses базы данных htmldatabase.
16. В соответствии с моделью Model-View-Controller создана модель UploadForm для выполнения скрипта распоковки zip-файлов, распаковки для доступа к таблице textbackgroundclasses базы данных htmldatabase.
17. Создан класс htmlcssparser.php, к которому подключёны парсеры html и css, и реализация парсинга и поиска селекторов при помощи его методов.
18. Создан класс SGRB\_func.php, который реализует сравнение цвета текста и фона.

**Описание алгоритма и методов решения**

*Наименование программы:*

Программа для оценки читаемости сложных html-страниц

*ПО, необходимое для функционирования программы:*

1. ОС Windows 7.
2. Apache версии 2.4.18
3. Framework Yii2 версии 2.0.7
4. Интерпретатор php версии 5.6.19
5. Html-парсер simple-html-dom.php
6. CSS-парсер css-parser v2-1.php (доработанный из v2).
7. Установка phpmyadmin.
8. Установка БД MySQL 5.7

*Язык, на котором написана программа:*

php

*Функциональное назначение:*

Автоматизация оценки читаемости текста на фоне из определённых областей путём вычисления контрастности между цветами текста и фона.

*Сведения об ограничениях:*

1. Сравнение цвета фона и текста происходит по алгоритму ...
2. Сравнение цвета фона и текста ведётся только в файлах css.
3. Поиск файлов css ведётся только в файлах html, htmls.
4. Файлы html, htmls с ошибками не записываются в базу данных.
5. Если ссылка на css не верна, или файл не открывается, он так же не записывается в базу данных.
6. Названия всех файлов обязательно должны быть на латинице, иначе скрипт пометит их как не существующие.
7. Если числа в полях background-color и color проверяемого селектора не соответствуют шаблону '#XXXXXX' или '#XXX', где X - число от 0 до F(f), пара будет промаркирована низкой контрастностью.
8. В сервисе не реализована возможность удаления файлов html и архива, так как процесс записи информации в несколько связанных таблиц базы данных требует доработки.

*Описание каждой функции, входящей в список:*

**Class** Addressarchives**()** – класс модели для доступа к таблице addressarchives, наследуется от класса ActiveRecord без описания дополнительных методов. Входные данные: значения колонок таблицы (id\_archive – первичный ключ загруженного архива, address\_archive – название архива, address\_folder –адресс загруженногог архива на диске, flag\_of\_archive – флаг наличия/отсутствия в архиве страниц с недостаточной контрастностью). Выходные данные: строки таблицы при запросе в базу данных.

**Class** Htmlpages**()** – класс модели для доступа к таблице htmlpages, наследуется от класса ActiveRecord без описания дополнительных методов. Входные данные: значения колонок таблицы (id\_page – первичный ключ найденной страницы, address\_page – адрес страницы html, id\_of\_archive - связанный ключ из таблицы addressarchives, flag\_of\_page – флаг наличия/отсутствия селекторов с недостаточной контрастностью). Выходные данные: строки таблицы при запросе в базу данных.

**Class** Textbackgroundcolors**()** – класс модели для доступа к таблице textbackgroundcolors, наследуется от класса ActiveRecord без описания дополнительных методов. Входные данные: значения колонок таблицы (id\_text\_back – первичный ключ для пары цветов текста и фона, найденных в селекторе, text\_color – цвет текста, back-color – цвет фона, flag\_of\_colors\_good – флаг достаточности контрастности для людей с хорошим зрением, flag\_of\_colors\_def – флаг достаточности контрастности для людей с дефектами зрения (близорукость, дальнозоркость), flag\_of\_colors\_contr - – флаг достаточности контрастности для людей потерей контрастной чуквствительности). Выходные данные: строки таблицы при запросе в базу данных.

**Class** Textbackgroundclasses**()** – класс модели для доступа к таблице textbackgroundclasses, наследуется от класса ActiveRecord без описания дополнительных методов. Входные данные: значения колонок таблицы (id\_class – первичный ключ для строки найденного селектора, class – наименование селектора, id\_text\_back – связанный ключ для таблицы textbackgroundcolors, определяющий строку с цветом текста и фоном, которым соответствует селектор, id\_page – связанный ключ для таблицы htmlpages, определяющий страницу, с которой связан селектор). Выходные данные: строки таблицы при запросе в базу данных.

**Class** UploadForm**()** – класс модели для операции распаковки архива, наследуется от класса Model.

Методы класса UploadForm:

public function rules(**) –** возвращает правила обработки данных для данной модели. Выходные данные: правила обработки данных.

public function upload() – функция загрузки zip-архива на сервер. Входные данные: файл zip-архива. Выходные данные: копия файла zip-архива на сервере.

public function unzip() – функция распаковки zip-архива на сервере. Входные данные: файл zip-архива на сервере. Выходные данные: папка с содежимым zip-архива.

public function htmllist() – функция поиска и создания списка файлов html, htmls из созданного архива. Входные данные: файл zip-архива. Выходные данные: массив из массивов, содежащих название страницы, путь к распакованной странице.

class HtmlAndCssParsing() – класс для парсинга html, htmls, css –файлов.

Метода класса HtmlAndCssParsing:

public function getCSSadresses($address) – получение со страницы всех адресов css. Входные данные: $address - адресс файла формата html. Выходные данные: массив $e\_href – массив адресов css файлов, полученных из входного файла, $errorLog – массив с ошибками при поиске и открытии файла html.

private function getCSSObject($strOfhtml, $strOfcss, $flag\_color\_back) – функция получения из файла css массива селекторов, имеющих заполненые поля «color» и «background-color». Входные данные: $strOfcss – относительный (относительно расположения страницы html) адрес файла css, ($strOfhtml – абсолютный адрес связанной html-страницы для получения асолютного адрес а css, $flag\_color\_back – флаг выбора поведения функции: 0 – в массив пишутся только селекторы с заполненным полем «background-color», 1 – в массив пишутся только селекторы с заполненным полем «color», 2 – в массив пишутся только селекторы с заполненными полями «color» и «background-color». Выходные данные: $ColorAndBack – массив с выбранными по флагу селекторами, $errorLog – массив с ошибками при поиске и открытии файла html.

public function getCSSObjects($htmlAdress, $e\_href) – функция получения массива из массивов селекторов, соответствующих входному массиву адресов css. Входные данные: $htmlAdress – адрес html, которому принадлежит $e\_href – список относительных адресов css, к которым html обращается. Выходные данные: $AllColorAndBack – массив из массивов с селекторами всех css файлов, $errorLog – массив с ошибками при поиске и открытии файла html.

private function cutWays($strOfhtml, $strOfcss) – функция для создания абсолютного пути для css из относительного пути. Входные данные: $strOfhtml – путь к html-странице, которая обращается к данному css-файлу по его относительному пути, $strOfcss – относительный путь к css-странице, который должен быть пробразован. Выходные данные: $outputstr – абсолютный путь к css файлу.

**Class** SRGBclass() – класс для подсчёта контрастности между цветами текста и фона селекторов.

Методы класса SRGBclass:

private functionSRGBcolor($color, &$L) – высчитывание критерия контрастности. Входные данные: $color - строка, соответствующая шаблону '#XXXXXX' или '#XXX', где X - число от 0 до F(f). Выходные данные: &$L – численное значение критерия контрастности для введённого цвета.

private function SRGBdelta ($color, $backgroundcolor, &$delta) – высчитывание разницы между критериями контрастности цвета текста и фона. Входные данные: $color – строка цвета текста, соответствующая шаблону '#XXXXXX' или '#XXX', где X - число от 0 до F(f) , $backgroundcolor– строка цвета фона, соответствующая шаблону '#XXXXXX' или '#XXX', где X - число от 0 до F(f). Выходные данные: &$delta - численное значение разницы между критериями контрастности.

public function compareColorAndBackForDataBase ($color, $backgroundcolor, &$answer) – определение, соответствует ли разница между критериями контрастности трём критериям: для людей с хорошим зрением, с дефектами зрения. Входные данные: $color – строка цвета текста, соответствующая шаблону '#XXXXXX' или '#XXX', где X - число от 0 до F(f) , $backgroundcolor– строка цвета фона, соответствующая шаблону '#XXXXXX' или '#XXX', где X - число от 0 до F(f). Выходные данные: &$answer - массив со следующими ключами изначениями: 'delta' -> разница между критериями цвета текста и фона, 'с хорошим зрением' – флаг 0/1 - прошёл/не прошёл, 'с дефектами зрения' – флаг 0/1 - прошёл/не прошёл, 'с дефектами зрения' – флаг 0/1 - прошёл/не прошёл.

public function verifyColor($color) – функция проверки цвета на соответствие шаблону '#XXXXXX' или '#XXX', где X - число от 0 до F(f). Входные данные: $color – строка с цветом в шестнадцатеричном формате. Выходные данные: $answer – булева перменная, true – соответствует шаблону.

**Сlass** AddressarchivesController – класс контроллера для работы с таблицей addressarchives, наследуется от класса Controller.

Методы класса AddressarchivesController:

public function actionIndex() – контроллер, создающий начальную страницу работы скрипта. Вызывается при начальном выхове и при отработке алгоритма:

1. Распаковка zip – архива.
2. Получение списка html-страниц
3. Парсинг html-страниц, получение ссылок на css-файлы, соответствующие каждой странице.
4. Парсинг css-файлов, нахождение селекторов, имеющих заполненные поля «color» и «background-color».
5. Сравнение заполненных полей для каждого селектора.
6. Заполнение таблиц базы данных.

Входные данные: zip-архив на ПК клиента.

Выходные данные: строки для заполнения таблиц базы данных, $dataProvider – оьъект-таблица всех строк таблицы addressarchives для отображения во view, $ArrayOfErrors – список с ошибками после работы с архивом во view.

public function actionView($id) – метод, перенаправляющий во view страницы «addresspages/index». Входные и выходные данные: $id – первичный ключ таблицы архивов addressarchives.

**Сlass** AddresspagesController – класс контроллера для работы с таблицей addresspages, наследуется от класса Controller.

Методы класса AddresspagesController:

public function actionIndex($id = null) – метод, показывающий строки таблицы addresspages, удовлетворяющие значению связанного ключа $id, имеющего одно из значений первичного ключа таблицы addresspages. Входные данные: $id – первичный ключ таблицы страниц addresspages. Выходные данные: $dataProvider – объект-таблица строк таблицы addressarchives, отвечающих значению связанного ключа $id (поле id\_of archive) для отображения во view.

public function actionView($id) – метод, перенаправляющий во view страницы «addresspageview/index». Входные и выходные данные: $id – первичный ключ таблицы страниц addresspages.

**Сlass** AddresspageviewController – класс контроллера для работы с таблицами textbackgroundclasses, textbackgroundcolors, наследуется от класса Controller.

Методы класса AddresspageviewController:

public function actionIndex($id = null) – метод, формирующий таблицу селекторов с флагами «соответствия/не соответствия» критериям контрастности. Входные данные: $id – первичный ключ таблицы страниц addresspages (поле id\_page), Выходные данные: $query – список строк с селекторами, соответствующими им цветоми текста и фона, таблицы сформированной из таблиц textbackgroundclasses, textbackgroundcolors.

*Используемые технические средства:*

1. Процессор Intel Core i3-6300 3,8 ГГц.
2. Оперативная память DDR3 8Гб.
3. Жесткий диск SATA 1Тб.

*Способ вызова и загрузки программы*:

Программа вызывается путём вставки строки «http://localhost/nikita/web/addressarchives/index» в командную строку браузера.

*Входные данные:*

Zip-архив с веб-страницами формата html, htmls.

*Выходные данные:*

Addressarchives\index.php – отображение формы загрузки для zip-архива, таблицы загруженных архивов и консоли, где в момент отработки алгоритма распознавания и сравнения отображается список ошибок .

Addresspages\index.php – отображение таблицы страниц загруженного архива.

Addresspageview\index.php – отображение таблицы селекторов, пар цветов текст-фон и флагов прохождения по критериям контрастности.

**Руководство по сборке и подготовке программы к работе**

1. Установка сервера Apache 2.4.18:

- Скачать сервера с сайта github по ссылке: <https://github.com/apache/httpd/releases?after=2.4.22>

- Распаковать сервер в определённую папку.

- Найти в подпапке conf файл httpd.conf, в котором:

Прописать путь к серверу, раскомментировав переменную ServerRoot "k:/Server/bin/Apache24".

Прописать порт, на котором будет висеть сервер Listen 80.

Раскомментировать нужные модули, которые сервер будет использовать в работе:

LoadModule access\_compat\_module modules/mod\_access\_compat.so

LoadModule actions\_module modules/mod\_actions.so

LoadModule alias\_module modules/mod\_alias.so

LoadModule allowmethods\_module modules/mod\_allowmethods.so

LoadModule asis\_module modules/mod\_asis.so

LoadModule auth\_basic\_module modules/mod\_auth\_basic.so

LoadModule authn\_core\_module modules/mod\_authn\_core.so

LoadModule authn\_file\_module modules/mod\_authn\_file.so

LoadModule authnz\_ldap\_module modules/mod\_authnz\_ldap.so

LoadModule authz\_core\_module modules/mod\_authz\_core.so

LoadModule authz\_groupfile\_module modules/mod\_authz\_groupfile.so

LoadModule authz\_host\_module modules/mod\_authz\_host.so

LoadModule authz\_user\_module modules/mod\_authz\_user.so

LoadModule autoindex\_module modules/mod\_autoindex.so

LoadModule cgi\_module modules/mod\_cgi.so

LoadModule dir\_module modules/mod\_dir.so

LoadModule env\_module modules/mod\_env.so

LoadModule include\_module modules/mod\_include.so

LoadModule isapi\_module modules/mod\_isapi.so

LoadModule ldap\_module modules/mod\_ldap.so

LoadModule log\_config\_module modules/mod\_log\_config.so

LoadModule mime\_module modules/mod\_mime.so

LoadModule negotiation\_module modules/mod\_negotiation.so

LoadModule rewrite\_module modules/mod\_rewrite.so

Создать папку, в которой будут находиться сайты, и прописать путь к ней: DocumentRoot "k:/Server/data/htdocs/"

Создать подпапки с названиями сайтов в созданной папке и прописать пути к ним согласно образцу:

<Directory "k:/Server/data/htdocs/nikita/web">

Options Indexes FollowSymLinks

AllowOverride All

Require all granted

</Directory>

1. Установка php 5.6.19:

Скачать php 5.6.19: <https://phpsources.net/scripts/php/php/versions/80_php>

Распаковать в определённую папку.

В httpd.conf сервера Apache прописать:

PHPIniDir "K:/Server/bin/PHP" – каталог, куда распакован php/

AddHandler application/x-httpd-php .php

LoadModule php5\_module "K:/Server/bin/PHP/php5apache2\_4.dll" – библиотека, с которой будет взаимодейтсвовать Apache для работы с php.

В папках php нужно найти php.ini и раскомментировать модули, которые будем использовать:

extension=php\_bz2.dll

extension=php\_curl.dll

extension=php\_fileinfo.dll

extension=php\_gd2.dll

extension=php\_gettext.dll

extension=php\_gmp.dll

extension=php\_intl.dll

extension=php\_imap.dll

extension=php\_interbase.dll

extension=php\_ldap.dll

extension=php\_mbstring.dll

extension=php\_exif.dll ; Must be after mbstring as it depends on it

extension=php\_mysql.dll

extension=php\_mysqli.dll

;extension=php\_oci8\_12c.dll ; Use with Oracle Database 12c Instant Client

extension=php\_oci8\_11g.dll ; Use with Oracle Database 11g Instant Client

;extension=php\_oci8.dll ; Use with Oracle Database 10g Instant Client

extension=php\_openssl.dll

extension=php\_pdo\_firebird.dll

extension=php\_pdo\_mysql.dll

extension=php\_pdo\_oci.dll

extension=php\_pdo\_odbc.dll

extension=php\_pdo\_pgsql.dll

extension=php\_pdo\_sqlite.dll

extension=php\_pgsql.dll

extension=php\_shmop.dll

extension=php\_soap.dll

extension=php\_sockets.dll

extension=php\_sqlite3.dll

extension=php\_sybase\_ct.dll

extension=php\_tidy.dll

extension=php\_xmlrpc.dll

extension=php\_xsl.dll

За тем найти и изменить переменную extension\_dir= "K:\Server\bin\PHP\ext" – где хранятся эти библиотеки.

1. Установка MySQL 5.7:

Скачать: <https://dev.mysql.com/downloads/windows/installer/5.7.html>

Распаковать и следовать указаниям установщика.

1. Установка phpmyadmin:

Скачать: <https://www.phpmyadmin.net/downloads/>

Распаковать в папку, предварительно созданную, как папка для сайта сервера Apache.

Создать в корне папки phpmyadmin файл conf.inc.php с примерно таким содержанием:

<?php

/\* Servers configuration \*/

$i = 0;

/\* Server: localhost [1] \*/

$i++;

$cfg['Servers'][$i]['verbose'] = '';

$cfg['Servers'][$i]['host'] = 'localhost';

$cfg['Servers'][$i]['port'] = '';

$cfg['Servers'][$i]['socket'] = '';

$cfg['Servers'][$i]['connect\_type'] = 'tcp';

$cfg['Servers'][$i]['extension'] = 'mysqli';

$cfg['Servers'][$i]['auth\_type'] = 'cookie';

$cfg['Servers'][$i]['user'] = 'root';

$cfg['Servers'][$i]['password'] = '';

$cfg['Servers'][$i]['nopassword'] = true;

$cfg['Servers'][$i]['AllowNoPassword'] = true;

/\* End of servers configuration \*/

$cfg['blowfish\_secret'] = '4c525c652d3f54.24302686';

$cfg['DefaultLang'] = 'ru';

$cfg['ServerDefault'] = 1;

$cfg['UploadDir'] = '';

$cfg['SaveDir'] = '';

?>

1. Установка Yii 2.0.7:

Скачать: <https://www.yiiframework.com/download>.

Распаковать в одну из подпапок папки, созданной для размещения сайтов.

Прописать в файле config/db.php для доступа к базе данных:

<?php

return [

'class' => 'yii\db\Connection',

'dsn' => 'mysql:host=localhost;dbname=htmldatabase',

'username' => 'root',

'password' => '',

'charset' => 'utf8',

];

1. Установить парсер simple-html-dom-parser.php

Скачать: <https://simplehtmldom.sourceforge.io/>

Рапаковать.

Переместить в папку controllers фреймворка yii2 следующие файлы:

simple-htm-dom.php

script\_for\_simple\_html\_dom.php

constatnts.php

HtmlDocument.php

HtmlNode.php

HtmlWeb.php

Debug.php

В начало файлов simple-htm-dom.php следует добавить namespace app\controllers; - это нужно, чтобы скрипты были видны в пространстве имён контроллеров.

1. Установить парсер css-parser.php

Скачать: <https://www.phpclasses.org/package/1289-PHP-CSS-parser-class.html#information>

Распаковать.

Переместить в папку controllers фреймворка yii2 следующие файлы:

V2/css-parser.php.

В файле парсера следует строку:

[59] $c = $code{$idx};

Поменять на:

[59]if (isset($code{$idx})) { /\* обработка зависания. Новохатько Н.Ю. г. Петрозаводск 07.10.2020 \*/

[60] $c = $code{$idx};

[61] } else {

[62] break;

[63] }

В начало файла добавить namespace app\controllers;

1. Предварительно нужно создать базу данных согласно выгрузке в приложении.

Нужно загрузить скрипт phpmyadmin.

Войти, вбив логии и пароль root пользователя.

Создать Новую базу данных.

Открыть «Импорт», «Выберите файл», найти файл htmldatabase.sql , нажать «Вперёд». Будет произведет импорт нашей БД.

Создастся БД из следующих таблиц:

Addressarchives

Htmlpages

Textbackhtmlclasses

Textbackhtmlcolors

1. Создать в папке views/addressarchives подпапки: archives – здесь будут храниться скачанные архивы, folders\_of\_archives – расположение распакованных архивов.
2. Установить скрипты сервиса обработки:

В папку controllers фреймворка yii2:

AddressarchivesController.php

AddresspagesController.php

AddressPageViewController.php

html\_dom\_parsing.php

SRGB\_func.php

В папку models фреймворка yii2::

Addressarchives.php

Htmlpages.php

TextbackGroundClasses.php

TextbackGroundColors.php

UploadForm.php

В папку views фреймворка yii2::

addressarchives/index.php

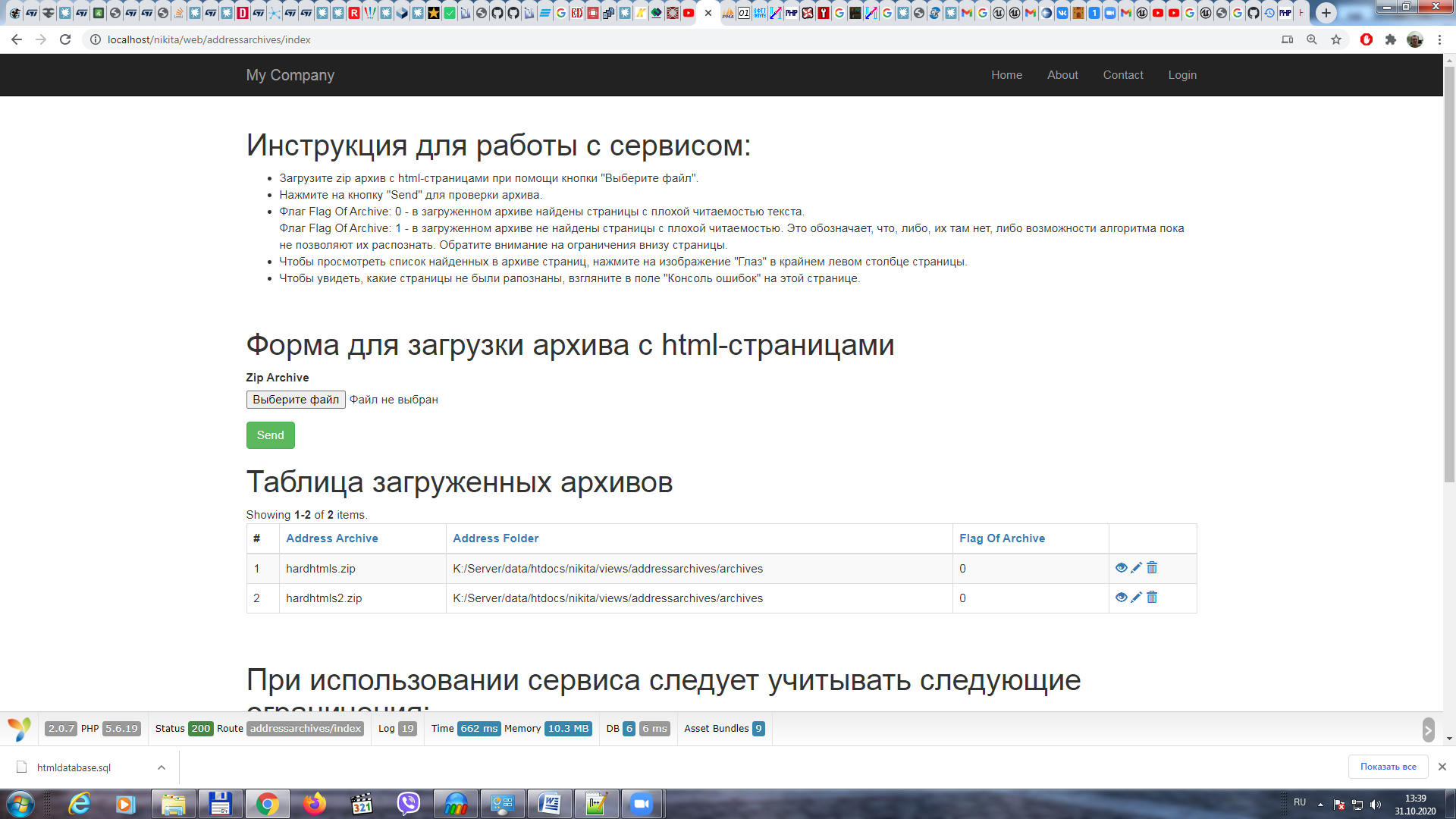
addresspages/index.php

addresspageview/index.php

1. Запустить исполняемый файл сервера httpd.exe.
2. Пройти по ссылке <http://localhost/nikita/web/addressarchives/index> - ссылка на локальном ПК.

**Руководство пользователя программы**

1. Заходим в браузер Internet\_Explorer, Chrome, Firefox. Вводим в строку ввода адреса <http://localhost/nikita/web/addressarchives/index>, жмём Enter.



На начальной странице расположена форма загрузки архива, таблица с загруженными архивами.

На данной странице возможно выполнить следующие действия:

 Загрузите zip архив с html-страницами при помощи кнопки "Выберите файл".

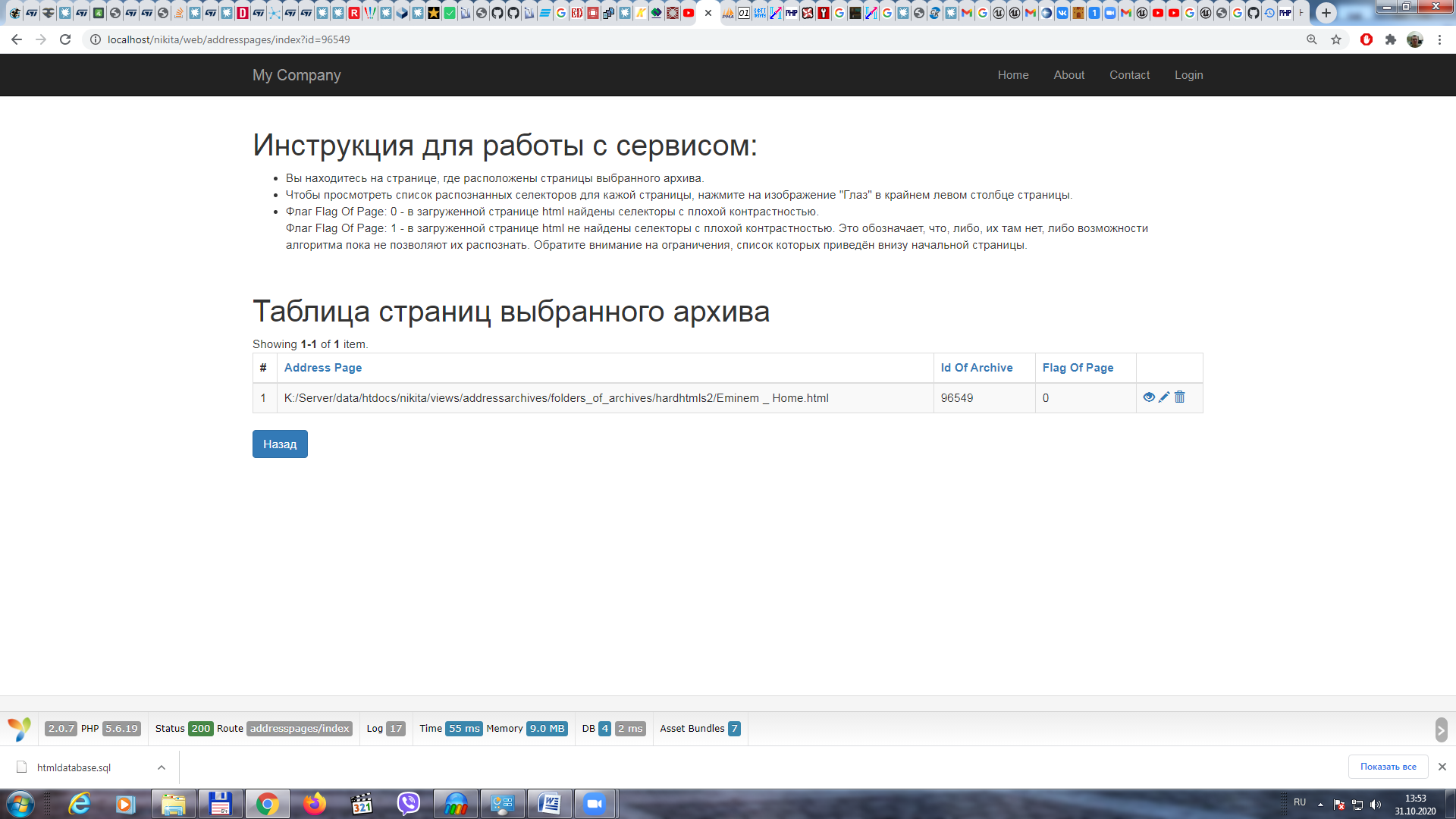
 Нажмите на кнопку "Send" для проверки архива.

 Флаг Flag Of Archive: 0 - в загруженном архиве найдены страницы с плохой читаемостью текста.  
Флаг Flag Of Archive: 1 - в загруженном архиве не найдены страницы с плохой читаемостью. Это обозначает, что, либо, их там нет, либо возможности алгоритма пока не позволяют их распознать. Обратите внимание на ограничения внизу страницы.

 Чтобы просмотреть список найденных в архиве страниц, нажмите на изображение "Глаз" в крайнем левом столбце страницы.

 Чтобы увидеть, какие страницы не были распознаны, взгляните в поле "Консоль ошибок" на этой странице.

1. При нажатии на изображение «Глаз» для любой из строк таблицы мы перейдём к списку страниц, найденных в архиве и проверенных.



На данной странице возможно выполнить следующие действия:

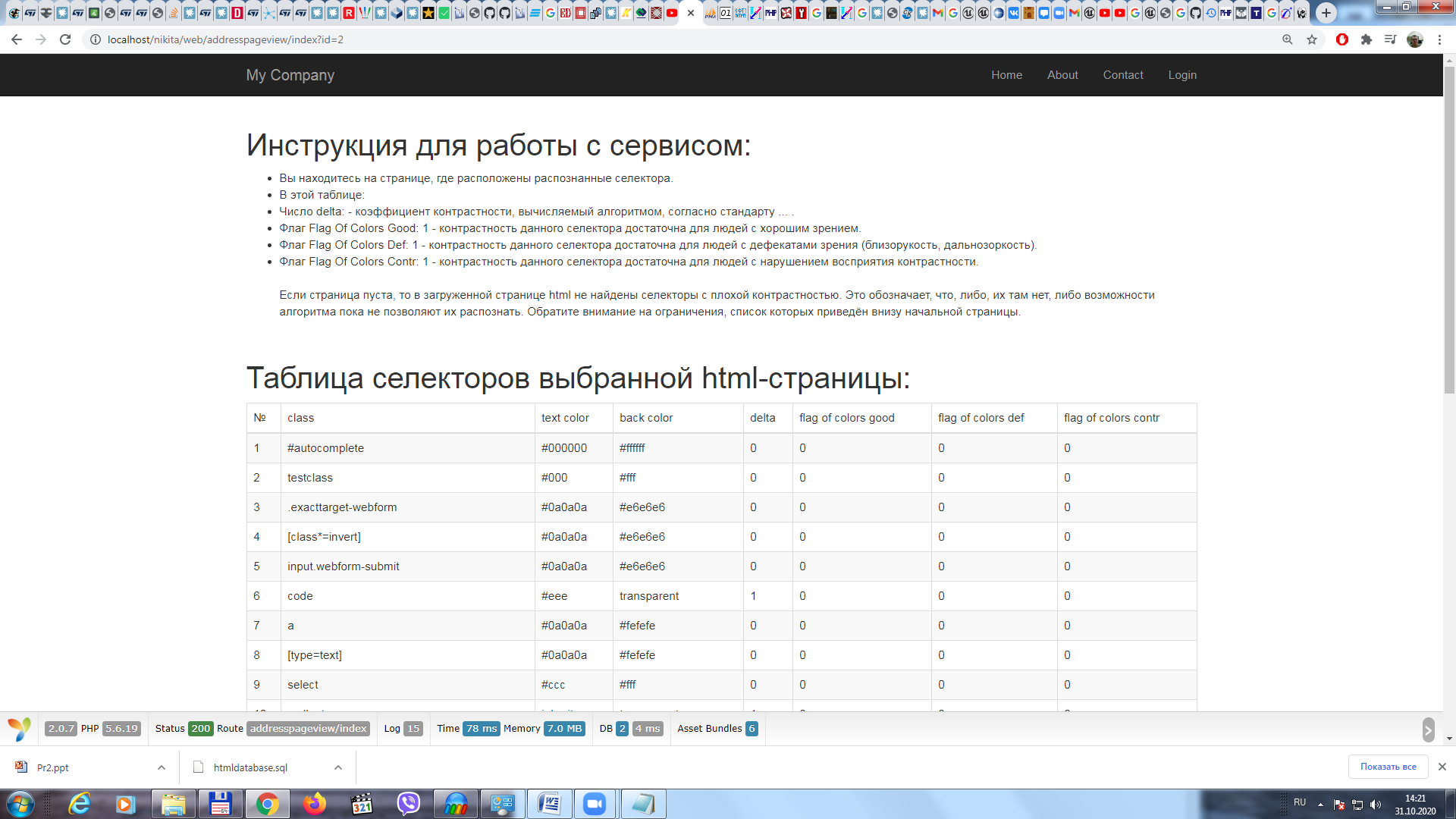
 Чтобы просмотреть список распознанных селекторов для кажой страницы, нажмите на изображение "Глаз" в крайнем левом столбце страницы.

 Флаг Flag Of Page: 0 - в загруженной странице html найдены селекторы с плохой контрастностью. Флаг Flag Of Page: 1 - в загруженной странице html не найдены селекторы с плохой контрастностью. Это обозначает, что, либо, их там нет, либо возможности алгоритма пока не позволяют их распознать. Обратите внимание на ограничения, список которых приведён внизу начальной страницы.

1. При нажатии на изображение «Глаз» для любой из строк таблицы мы перейдём к таблице, где отображены пары цвет текста - цвет фона из файлов css, к которым обращается html-страница. Данная таблица содержит:

* Число delta: - коэффициент контрастности, вычисляемый алгоритмом, согласно стандарту ... .
* Флаг Flag Of Colors Good: 1 - контрастность данного селектора достаточна для людей с хорошим зрением.
* Флаг Flag Of Colors Def: 1 - контрастность данного селектора достаточна для людей с дефекатами зрения (близорукость, дальнозоркость).
* Флаг Flag Of Colors Contr: 1 - контрастность данного селектора достаточна для людей с нарушением восприятия контрастности.

Если страница пуста, то в загруженной странице html не найдены селекторы с плохой контрастностью. Это обозначает, что, либо, их там нет, либо возможности алгоритма пока не позволяют их распознать. Обратите внимание на ограничения, список которых приведён внизу начальной страницы.

****

1. Для возврата на предыдущие страницы следует использовать кнопку «Назад».

**Отчёт о тестировании**

Данные для тестирования содержатся в тестовых переменных проекта:

Тестовые zip-архивы в папке controllers/archives:

Hardhtmls.zip: - содержит:

Eminem-home – должна быть обработана корректно и записана в базу данных.

Яндекс – должна быть выдана ошибка из-за кодировки кириллических символов.

Hardhtmls1.zip: - содержит:  
 подпапку hardhtmls – должна быть открыта корректно.

Eminem-home – должна быть обработана корректно и записана в базу данных.

ya – должна быть выдана ошибка из-за кодировки кириллических символов и невозможности открыть страницу по указанному пути.

Hardhtmls2.zip: - содержит:  
 подпапку hardhtmls – должна быть открыта корректно.

Eminem-home – должна быть обработана корректно и записана в базу данных.

HardhtmlsYa.zip:

Eminem-home – должна быть обработана корректно и записана в базу данных.

Яндекс – должна быть выдана ошибка из-за невозможности открыть страницу по указанному пути.

Manyzip.zip:

Должен выдать ошибки, связанные с тем, что файл пуст.

ManyzipHI.zip:

Все файлы пусты, кроме test1.htmls – в нём текст без тегов. Должен выдать ошибки, связанные с тем, что файл пуст.

*Результаты тестирования:*

Массив ошибок Hardhtmls.zip:

Array

(

[0] => Array

(

[0] => файл css не существует

[1] => K:/Server/data/htdocs/nikita/views/addressarchives/folders\_of\_archives/hardhtmls/Яндекс\_files/ZcQKkS0U6rxo560H5a\_XEfePg.css

)

[1] => Array

(

[0] => файл css не существует

[1] => K:/Server/data/htdocs/nikita/views/addressarchives/folders\_of\_archives/hardhtmls/yastatic.net/s3/home-static/\_/s/\_/ZcQKkS0U6rxo560H5a\_XEfePg.css

)

)

Название папки Яндекс\_files распознаётся некорректно и не существует пути yastatic.net, соответственно, файл не находится.

Массив ошибок Hardhtmls1.zip:

Array

(

[0] => Array

(

[0] => файл css не существует

[1] => K:/Server/data/htdocs/nikita/views/addressarchives/folders\_of\_archives/hardhtmls1/hardhtmls/Яндекс\_files/ZcQKkS0U6rxo560H5a\_XEfePg.css

)

[1] => Array

(

[0] => файл css не существует

[1] => K:/Server/data/htdocs/nikita/views/addressarchives/folders\_of\_archives/hardhtmls1/hardhtmls/yastatic.net/s3/home-static/\_/s/\_/ZcQKkS0U6rxo560H5a\_XEfePg.css

)

)

Название папки Яндекс\_files распознаётся некорректно некорректно и не существует пути yastatic.net, соответственно, файл не находится.

Массив ошибок Hardhtmls2.zip:

Array

(

)

Массив ошибок HardhtmlsYa.zip:

Array

(

[0] => Array

(

[0] => файл css не существует

[1] => K:/Server/data/htdocs/nikita/views/addressarchives/folders\_of\_archives/hardhtmlsYa/yastatic.net/s3/home-static/\_/s/\_/ZcQKkS0U6rxo560H5a\_XEfePg.css

)

)

Массив ошибок manyzip.zip:

Array

(

[0] => Array

(

[0] => страница html пуста

[1] => K:/Server/data/htdocs/nikita/views/addressarchives/folders\_of\_archives/manyzip/manyzip/one/free/test1.htmls

)

[1] => Array

(

[0] => страница html пуста

[1] => K:/Server/data/htdocs/nikita/views/addressarchives/folders\_of\_archives/manyzip/manyzip/one/free/test2.htmls

)

[2] => Array

(

[0] => страница html пуста

[1] => K:/Server/data/htdocs/nikita/views/addressarchives/folders\_of\_archives/manyzip/manyzip/one/test1.html

)

[3] => Array

(

[0] => страница html пуста

[1] => K:/Server/data/htdocs/nikita/views/addressarchives/folders\_of\_archives/manyzip/manyzip/one/test2.html

)

[4] => Array

(

[0] => страница html пуста

[1] => K:/Server/data/htdocs/nikita/views/addressarchives/folders\_of\_archives/manyzip/manyzip/test1.html

)

[5] => Array

(

[0] => страница html пуста

[1] => K:/Server/data/htdocs/nikita/views/addressarchives/folders\_of\_archives/manyzip/manyzip/two/test1.htmls

)

[6] => Array

(

[0] => страница html пуста

[1] => K:/Server/data/htdocs/nikita/views/addressarchives/folders\_of\_archives/manyzip/manyzip/two/test2.html

)

)

Массив ошибок manyzipHI.zip:

Array

(

[0] => Array

(

[0] => страница html пуста

[1] => K:/Server/data/htdocs/nikita/views/addressarchives/folders\_of\_archives/manyzipHI/manyzip/one/free/test1.htmls

)

[1] => Array

(

[0] => страница html пуста

[1] => K:/Server/data/htdocs/nikita/views/addressarchives/folders\_of\_archives/manyzipHI/manyzip/one/free/test2.htmls

)

[2] => Array

(

[0] => страница html пуста

[1] => K:/Server/data/htdocs/nikita/views/addressarchives/folders\_of\_archives/manyzipHI/manyzip/one/test1.html

)

[3] => Array

(

[0] => страница html пуста

[1] => K:/Server/data/htdocs/nikita/views/addressarchives/folders\_of\_archives/manyzipHI/manyzip/one/test2.html

)

[4] => Array

(

[0] => страница html пуста

[1] => K:/Server/data/htdocs/nikita/views/addressarchives/folders\_of\_archives/manyzipHI/manyzip/two/test1.htmls

)

[5] => Array

(

[0] => страница html пуста

[1] => K:/Server/data/htdocs/nikita/views/addressarchives/folders\_of\_archives/manyzipHI/manyzip/two/test2.html

)

)

**Заключение**

Цель - создание программы, осуществляющей автоматизированную проверку читаемости html-страниц. – нами выполнена.

Исходя из цели, нами были поставлены задачи:

1. Созданы страницы с формой загрузки zip-архива и таблицей с количество загруженных архивов, со списком найденных страниц для архива, со списком найденных тегов для каждой страницы.
2. Созданы контроллеры и методы для каждой из отображаемых страниц.
3. Созданы необходимые таблицы и модели к ним.
4. Создан метод, позволяющий найти на страницах ссылки на файлы css и, разобрав эти файлы, найти селекторы с заполненными полями для текста и фона и сравнить их, чтобы вычислить контрастность.

**Исходный код**

**AddressarchivesController.php**

<?php

namespace app\controllers;

use yii\web\Controller;

use yii\data\Pagination;

use app\models\Addressarchives;

use app\models\ContrastForm;

use app\models\Htmlpages;

use app\models\Textbackgroundcolors;

use app\models\Textbackgroundclasses;

use yii\data\ActiveDataProvider;

use Yii;

use app\models\UploadForm;

use yii\web\UploadedFile;

include 'html\_dom\_parsing.php';

include 'SRGB\_func.php';

//include 'testingClass.php';

class AddressarchivesController extends Controller

{

/\*Создания экшена (index) по умолчанию\*/

public function actionIndex()

{

/\* Создание формы для загрузки (с зелёной кнопкой) \*/

$form\_model = new ContrastForm();

/\*----- \*/

/\* Нахождение всех строк таблицы \*/

$query = Addressarchives::find();

/\*----- Создание списка \*/

/\* Объявление разбивки на страницы. Размер стр - 5, количество строк в таблице получаем по count() \*/

/\* $pagination - выдача одновременно только одной страницы данных, в зависимости от выбранной страницы \*/

$pagination = new Pagination([

'defaultPageSize' => 5,

'totalCount' => $query->count(),

]);

/\* Сортировка по столбцу address\_archive\*/

$addresses = $query->orderBy('address\_archive')

->offset($pagination->offset)

->limit($pagination->limit)

->all();

/\*----- \*/

/\*----- Создание таблицы \*/

$dataProvider = new ActiveDataProvider([

'query' => $query,

'pagination' => [

'pageSize' => 5,

],

]);

/\*----- \*/

/\* Создание формы для загрузки (со светлыми кнопками) \*/

$up\_model = new UploadForm();

/\* isPost - если поступил POST запрос с кнопки формы, то проделываем следующие операции: загрузка файла на сервер через вызов метода upload() модели формы UploadForm \*/

if (Yii::$app->request->isPost) {

/\* Формирование данных для таблиц Addressarchives и Htmlpages \*/

$up\_model->zipArchive = UploadedFile::getInstance($up\_model, 'zipArchive');

$flag\_upload = $up\_model->upload();

$flag\_unzip = $up\_model->unzip();

/\* htmls - архив список с названиями и адресами файлов \*/

$up\_model->cut = array ( 0 => '/', 1 => '.html', 2 => '.htmls', );

$flag\_htmls = $up\_model->htmllist();

/\* Формирование данных для таблиц textbackgroundclasses и textbackgroundcolors \*/

$HtmlCss = new HtmlAndCssParsing();

$ArrayOfErrors = array(); /\* Отправка ссылки на ошибки \*/

$HtmlCss->arrayOferrors = &$ArrayOfErrors;

//$str = 'K:/Server/data/htdocs/nikita/views/addressarchives/folders\_of\_archives/';

//$str = $str.$up\_model->zipNoFormat.'/';

$AllColorAndBacks = Null;

$test1 = array();

$test2 = array();

$test3 = array();

$htmls\_count = count($up\_model->htmls); //количество найденных файлов подходящего формата. В тестовом архиве - 7

for($i = 0; $i < $htmls\_count; $i++)

{

$e\_href = $HtmlCss->getCSSadresses($up\_model->htmls[$i][1]);

/\*

1. Решить проблемы с кодировкой раскрываемого файла. - html, ссылки на которые не работают, будут пропускаться. В данном случае +

2. Подумать о выводе ошибок и их показе на странице. +

3. Подумать об обработке отказа с пустыми страницами. +

4. Вывести список страниц. +

\*/

$test1[$i] = $up\_model->htmls[$i][1];

if (isset($e\_href)) {

$test2[$i] = $e\_href; /\* тестовый e\_href \*/

$AllColorAndBack = $HtmlCss->getCSSObjects($up\_model->htmls[$i][1], $e\_href); //???

if (isset($AllColorAndBack)) {

$test3[$i] = $AllColorAndBack; //На сложной тестовой странице работает

// Разделение по запятым, чтобы отделить каждый класс в перечислении, и пробелам, чтобы отделить разные ступени вложенных классов

$AllColorAndBacks[$i] = $HtmlCss->getSearchFromCSSs($AllColorAndBack);

//$test3[$i] = $AllColorAndBacks[$i] ;

}

}

}

/\*Вставка данных в таблицы\*/

/\*Вставка данных в таблицу Addressarchives\*/

$addarch = new Addressarchives();

/\* Создание индивидуального ключа \*/

$flag\_primary\_key = true;

while ($flag\_primary\_key) {

$addarch\_primary\_key= rand(0, 100000);

$query = $addarch->find()->where(['id\_archive' => $addarch\_primary\_key])->one();

if (!$query) { // Если такого ключа нет, то берём его и завершаем цикл

$flag\_primary\_key = false;

}

}

$single\_table = 'Addressarchives';

$single\_row = [

'id\_archive' => $addarch\_primary\_key, //Позже использовать для update строки в ячейке флага

'address\_archive' => $up\_model->zipName,

'address\_folder' => $up\_model->downloadWay,

];

Yii::$app->db->createCommand()->insert($single\_table, $single\_row)->execute();

/\* Вставка данных в таблицу Htmlpages \*/

$htmlp = new Htmlpages();

$htmlp\_count = $htmlp->find()->count() + 2; //Количество строк в заполянемой таблице + 1, то есть номер первой несуществующей строки

$multiple\_table = 'Htmlpages';

$multiple\_columns = ['id\_page', 'address\_page', 'id\_of\_archive'];

$multiple\_rows;

for($i = 0; $i < $htmls\_count; $i++)

{

$multiple\_rows[] = [$htmlp\_count + $i, $up\_model->htmls[$i][1], $addarch\_primary\_key];

$FortextbackgroundclassesTable[$i] = $htmlp\_count + $i;

}

Yii::$app->db->createCommand()->batchInsert($multiple\_table, $multiple\_columns, $multiple\_rows)->execute();

/\*Вставка данных в таблицу textbackgroundcolors\*/

$textBackColors = new Textbackgroundcolors();

$colorsp\_count = $textBackColors->find()->count() + 2;

$colorsp\_i = 0;

$colorsp\_table = 'Textbackgroundcolors';

$colorsp\_columns = ['id\_text\_back', 'text\_color', 'back\_color', 'delta', 'flag\_of\_colors\_good', 'flag\_of\_colors\_def', 'flag\_of\_colors\_contr'];

$colorsp\_rows;

/\*Вставка данных в таблицу textbackgroundclasses\*/

$textBackClasses = new Textbackgroundclasses();

$classesp\_count = $textBackClasses->find()->count() + 2;

$classesp\_i = 0;

$classesp\_table = 'Textbackgroundclasses';

$classesp\_columns = ['id\_class', 'class', 'id\_text\_back', 'id\_page'];

$classesp\_rows;

/\*Обновление данных в таблице htmppages\*/

//$htmlpages\_table = 'HtmlPages';

//$htmlpages\_columns = ['id\_class', 'class', 'id\_text\_back', 'id\_page'];

//$htmlpages\_rows;

$RGBforcompare = new SRGBclass();

//$RGBforcompare->arrayOferrors = &$ArrayOfErrors;

$parentsFlag = 1;

if (isset($AllColorAndBacks)) {

//$ArrayOfErrors = array(); /\* текстовый \*/

foreach ($AllColorAndBacks as $htmlKey => $AllColorAndBack) { /\*htmlKey - цифровой ключ, соответствующий ключу i в массиве $up\_model->htmls[$i][1] \*/

foreach ($AllColorAndBack as $ColorAndBack) {

/\* уровень перечисления строк селекторов \*/

foreach ($ColorAndBack as $ColorAndBack\_element) {

/\* уровень перечисления массивов 0 - цвет, 1 - фон, 2 - одиночный селектор \*/

//array\_push($ArrayOfErrors, array( 0 => $ColorAndBack\_element[1][0], 1 => $ColorAndBack\_element[1][1], 2 => $ColorAndBack\_element[0][0][0],) );

$answer = null;

$RGBforcompare->compareColorAndBackForDataBase($ColorAndBack\_element[1][0], $ColorAndBack\_element[1][1], $answer);

$colorsp\_rows[] = [$colorsp\_count + $colorsp\_i, $ColorAndBack\_element[1][0], $ColorAndBack\_element[1][1], $answer['delta'], $answer['с хорошим зрением'], $answer['с дефектами зрения'], $answer['с дефектами зрения']];

$colorsp\_i++;

/\*Адаптация сформированных данных для вставки в таблицу textbackgroundclasses\*/

//$ColorAndBack\_element[1] - нас интересует этот массив

foreach ($ColorAndBack\_element[0][0] as $element) {

$htmlArchivePrimary = $FortextbackgroundclassesTable[$htmlKey]; //Индивидуальные ключи в таблице HtmlPages. Так как лишние html-страницы, которых не сущетсвует, ещё не переберались, они там есть.

$classesp\_rows[] = [$classesp\_count + $classesp\_i, $element, $colorsp\_count + $colorsp\_i, $htmlArchivePrimary];

$classesp\_i++;

}

/\* Проверка, есть ли в массиве флагов прохождения барьеров контрастности хоть одна 1, и если да, то выставление

в ячейки строки таблицы HtmlPages флага 0 (страница) и в таблице AddressArchives флага 0 (архив)\*/

$parentsFlag = in\_array(0,$answer);

if ($parentsFlag) {

//Исправяем таблицу HtmlPages

$htmlUpdate = $htmlp->findOne($htmlArchivePrimary);

$htmlUpdate['flag\_of\_page'] = 0;

$htmlUpdate->update();

}

}

}

}

Yii::$app->db->createCommand()->batchInsert($colorsp\_table, $colorsp\_columns, $colorsp\_rows)->execute();

Yii::$app->db->createCommand()->batchInsert($classesp\_table, $classesp\_columns, $classesp\_rows)->execute();

if ($parentsFlag) {

//Исправяем таблицу AddressArchives

$htmlUpdate = $addarch->findOne($addarch\_primary\_key);

$htmlUpdate['flag\_of\_archive'] = 0;

$htmlUpdate->update();

}

}

if ($flag\_upload & $flag\_unzip & $flag\_htmls) {

// file is uploaded successfully

return $this->render('index', [

'form\_model' => $form\_model,

'addresses' => $addresses,

'pagination' => $pagination,

'dataProvider' => $dataProvider,

'up\_model' => $up\_model,

'AllColorAndBacks' => $AllColorAndBacks,

'ArrayOfErrors' => $ArrayOfErrors,

'test1' => $test1,

'test2' => $test2,

'test3' => $test3

]);

}

}

/\*----- \*/

$AllColorAndBacks = Null; /\*тестовый\*/

$ArrayOfErrors = Null; /\*тестовый\*/

$test1 = Null;

$test2 = Null;

$test3 = Null;

/\* Выдача во view (index) информации, которая будет показываться в списке и таблице \*/

return $this->render('index', [

'form\_model' => $form\_model,

'addresses' => $addresses,

'pagination' => $pagination,

'dataProvider' => $dataProvider,

'up\_model' => $up\_model,

'AllColorAndBacks' => $AllColorAndBacks,

'ArrayOfErrors' => $ArrayOfErrors,

'test1' => $test1,

'test2' => $test2,

'test3' => $test3

]);

}

/\*

Создание экшена (view), показывающего, какие страницы проверены и находятся в данном архиве.

\*/

public function actionView($id)

{

return $this->redirect(['addresspages/index', 'id'=> $id]);

}

/\*

Создание экшена (Update), показывающего, какие страницы проверены и находятся в данном архиве.

\*/

public function actionUpdate($id)

{

//$this->findModel($id)->delete();

return $this->redirect(['index']);

}

/\*

Создание экшена (Delete), показывающего, какие страницы проверены и находятся в данном архиве.

\*/

public function actionDelete($id)

{

//$this->findModel($id)->delete();

return $this->redirect(['index']);

}

//Загрузка файла

public function actionUpload()

{

$model = new UploadForm();

if (Yii::$app->request->isPost) {

$model->imageFile = UploadedFile::getInstance($model, 'imageFile');

if ($model->upload()) {

// file is uploaded successfully

return;

}

}

return $this->render('upload', ['model' => $model]);

}

/\*\*

\* Finds the Usertable model based on its primary key value.

\* If the model is not found, a 404 HTTP exception will be thrown.

\* @param integer $id

\* @return Usertable the loaded model

\* @throws NotFoundHttpException if the model cannot be found

\*/

protected function findModel($id)

{

if (($model = Usertable::findOne($id)) !== null) {

return $model;

} else {

throw new NotFoundHttpException('The requested page does not exist.');

}

}

}

?>

**AddresspagesController.php**

<?php

namespace app\controllers;

use yii\web\Controller;

use yii\data\Pagination;

use app\models\Addressarchives;

use app\models\ContrastForm;

use app\models\Htmlpages;

use yii\data\ActiveDataProvider;

use Yii;

use app\models\UploadForm;

use yii\web\UploadedFile;

class AddresspagesController extends Controller

{

public function actions()

{

return [

'error' => [

'class' => 'yii\web\ErrorAction',

],

'captcha' => [

'class' => 'yii\captcha\CaptchaAction',

'fixedVerifyCode' => YII\_ENV\_TEST ? 'testme' : null,

],

'page' => [

'class' => 'yii\web\ViewAction',

],

];

}

/\*Создания экшена (index) по умолчанию\*/

public function actionIndex($id = null)

{

/\*Получение id - номера в таблице \*/

/\*Получение $typeView - тип вида (список страниц, либо страница) - необязательный параметр \*/

/\* Здесь будет нужно получитиь id, найти все совпадающие с ним строчки в другой таблице и отобразить ссылки на них \*/

/\* Нахождение всех строк таблицы \*/

//$query = Htmlpages::find();

/\*Нахождение строк только по нужному id\*/

$query = Htmlpages::find()->where(['id\_of\_archive' => $id]);

/\*----- Создание таблицы \*/

$dataProvider = new ActiveDataProvider([

'query' => $query,

'pagination' => [

'pageSize' => 5,

],

]);

/\*----- \*/

return $this->render('index', [

'dataProvider' => $dataProvider,

//'id' => $id,

]);

}

/\*

Создание экшена (view), показывающего скачанную страницу html.

\*/

public function actionView($id = Null)

{

//$this->findModel($id)->delete();

//echo $id;

return $this->redirect(['addresspageview/index', 'id'=> $id]);

//return Yii::$app->response->sendFile('K:\Server\data\htdocs\nikita\views\addresspages\pages\mytest.php', ['inline'=>true]);

}

/\*

Создание экшена (Update), показывающего, какие страницы проверены и находятся в данном архиве.

\*/

public function actionUpdate($id)

{

//$this->findModel($id)->delete();

return $this->redirect(['index']);

}

/\*

Создание экшена (Delete), показывающего, какие страницы проверены и находятся в данном архиве.

\*/

public function actionDelete($id)

{

//$this->findModel($id)->delete();

return $this->redirect(['index']);

}

//Загрузка файла

public function actionUpload()

{

$model = new UploadForm();

if (Yii::$app->request->isPost) {

$model->imageFile = UploadedFile::getInstance($model, 'imageFile');

if ($model->upload()) {

// file is uploaded successfully

return;

}

}

return $this->render('upload', ['model' => $model]);

}

/\*\*

\* Finds the Usertable model based on its primary key value.

\* If the model is not found, a 404 HTTP exception will be thrown.

\* @param integer $id

\* @return Usertable the loaded model

\* @throws NotFoundHttpException if the model cannot be found

\*/

protected function findModel($id)

{

if (($model = Usertable::findOne($id)) !== null) {

return $model;

} else {

throw new NotFoundHttpException('The requested page does not exist.');

}

}

}

?>

**AddressPageViewController.php**

<?php

namespace app\controllers;

use yii\web\Controller;

use yii\data\Pagination;

use app\models\Addressarchives;

use app\models\ContrastForm;

use app\models\Htmlpages;

use app\models\Textbackgroundclasses;

use yii\data\ActiveDataProvider;

use Yii;

use app\models\UploadForm;

use yii\web\UploadedFile;

class AddresspageviewController extends Controller

{

public function actions()

{

return [

'error' => [

'class' => 'yii\web\ErrorAction',

],

'captcha' => [

'class' => 'yii\captcha\CaptchaAction',

'fixedVerifyCode' => YII\_ENV\_TEST ? 'testme' : null,

],

'page' => [

'class' => 'yii\web\ViewAction',

],

];

}

/\*Создания экшена (index) по умолчанию\*/

public function actionIndex($id = null)

{

/\*Получение id - номера в таблице \*/

/\* Здесь будет нужно получитиь id, найти все совпадающие с ним строчки в другой таблице и отобразить ссылки на них \*/

/\* Соединение таблиц по id \*/

//$query = Textbackgroundclasses::find()->where(['id\_page' => $id])->join('RIGHT JOIN', 'textbackgroundcolors', 'textbackgroundcolors.id\_text\_back = textbackgroundclasses.id\_text\_back');

/\* Запрос выбора с соединением двух связанных таблиц \*/

$query = (new \yii\db\Query())

->select(['id\_class', 'class', 'text\_color', 'back\_color', 'delta', 'flag\_of\_colors\_good', 'flag\_of\_colors\_def', 'flag\_of\_colors\_contr'])

->from('textbackgroundclasses')

->join('RIGHT JOIN', 'textbackgroundcolors', 'textbackgroundclasses.id\_text\_back = textbackgroundcolors.id\_text\_back')

->where(['id\_page' => $id])

->all();

/\* Находим id\_архива для возврата \*/

$queryBack = (new \yii\db\Query())

->select(['id\_of\_archive', ])

->from('htmlpages')

->where(['id\_page' => $id])

->one();

/\*

$query = Textbackgroundclasses::find()

->select(['id\_class', 'class', 'text\_color', 'back\_color', 'delta', 'flag\_of\_colors\_good', 'flag\_of\_colors\_def', 'flag\_of\_colors\_contr'])

->from('textbackgroundclasses')

->join('RIGHT JOIN', 'textbackgroundcolors', 'textbackgroundclasses.id\_text\_back = textbackgroundcolors.id\_text\_back')

->where(['id\_page' => $id])

->all();

\*/

//$query

/\*

$address = $query->address\_page;

$buffer = array();

$handle = fopen($address, "r");

while (!feof($handle)) {

//$buffer = fgets($handle, 4096);

array\_push($buffer, fgets($handle, 4096) );

//echo $buffer;

}

fclose($handle);

$countBuffer = count($buffer);

\*/

/\* Нахождение всех строк таблицы \*/

//$query = Htmlpages::find();

/\*----- Создание таблицы \*/

$dataProvider = new ActiveDataProvider([

'query' => $query,

'pagination' => [

'pageSize' => 20,

],

]);

/\*----- \*/

return $this->render('index', [

'dataProvider' => $dataProvider,

'query' => $query,

'queryBack' => $queryBack,

//'id' => $id,

//'buffer' => $buffer, 'countBuffer' => $countBuffer//, 'address' => $address, 'query' => $query

]);

}

/\*

Создание экшена (view), показывающего скачанную страницу html.

\*/

public function actionView($id = Null)

{

return $this->redirect(['index']);

}

/\*

Создание экшена (Update), показывающего, какие страницы проверены и находятся в данном архиве.

\*/

public function actionUpdate($id)

{

//$this->findModel($id)->delete();

return $this->redirect(['index']);

}

/\*

Создание экшена (Delete), показывающего, какие страницы проверены и находятся в данном архиве.

\*/

public function actionDelete($id)

{

//$this->findModel($id)->delete();

return $this->redirect(['index']);

}

//Загрузка файла

public function actionUpload()

{

$model = new UploadForm();

if (Yii::$app->request->isPost) {

$model->imageFile = UploadedFile::getInstance($model, 'imageFile');

if ($model->upload()) {

// file is uploaded successfully

return;

}

}

return $this->render('upload', ['model' => $model]);

}

/\*\*

\* Finds the Usertable model based on its primary key value.

\* If the model is not found, a 404 HTTP exception will be thrown.

\* @param integer $id

\* @return Usertable the loaded model

\* @throws NotFoundHttpException if the model cannot be found

\*/

protected function findModel($id)

{

if (($model = Usertable::findOne($id)) !== null) {

return $model;

} else {

throw new NotFoundHttpException('The requested page does not exist.');

}

}

}

?>

**AddressArchives.php:**

<?php

/\*

Пространство имён, определяет, для какой группы функций

будет доступна эта модель.

models - пространство имён моделей (классов баз данных)

\*/

namespace app\models;

/\*

Объявление, от какого класса наследовать наш класс.

\*/

use yii\db\ActiveRecord;

/\*

Объявление класса, соответствующего имени таблицы в базе данных.

Связывание таблицы с этиим классом будет выполнено автоматически.

\*/

class Addressarchives extends ActiveRecord

{

/\*

rules - правила валидации, используются при проверке данных.

Также могут использоваться переопределения:

attributes()

attributeLabels()

Также могут использоваться объявления сценариев для больших проектов.

атрибуты rules:

required - поле обязательно для заполнения

trim - удаление пробелов из поля

message => "любой текст" - показ сообщения при неправильном вводе данного

when => function() - условная валидация. Можно проверять соответствие аттрибута какому то условию.

default - обработка пустого входного поля.

\*/

public function rules()

{

return [

// атрибут required указывает, что данные обязательны для заполнения

[['id\_archive', 'address\_archive', 'address\_folder'], 'required'],

];

}

}

?>

**Htmlpages.php:**

<?php

/\*

Пространство имён, определяет, для какой группы функций

будет доступна эта модель.

models - пространство имён моделей (классов баз данных)

\*/

namespace app\models;

/\*

Объявление, от какого класса наследовать наш класс.

\*/

use yii\db\ActiveRecord;

/\*

Объявление класса, соответствующего имени таблицы в базе данных.

Связывание таблицы с этиим классом будет выполнено автоматически.

\*/

class Htmlpages extends ActiveRecord

{

}

**Textbackgroundcolors.php:**

<?php

/\*

Пространство имён, определяет, для какой группы функций

будет доступна эта модель.

models - пространство имён моделей (классов баз данных)

\*/

namespace app\models;

/\*

Объявление, от какого класса наследовать наш класс.

\*/

use yii\db\ActiveRecord;

/\*

Объявление класса, соответствующего имени таблицы в базе данных.

Связывание таблицы с этиим классом будет выполнено автоматически.

\*/

class Textbackgroundcolors extends ActiveRecord

{

}

**Textbackgroundclasses.php:**

<?php

/\*

Пространство имён, определяет, для какой группы функций

будет доступна эта модель.

models - пространство имён моделей (классов баз данных)

\*/

namespace app\models;

/\*

Объявление, от какого класса наследовать наш класс.

\*/

use yii\db\ActiveRecord;

/\*

Объявление класса, соответствующего имени таблицы в базе данных.

Связывание таблицы с этиим классом будет выполнено автоматически.

\*/

class Textbackgroundclasses extends ActiveRecord

{

}

**Addressarchives/index.php:**

<?php

use yii\helpers\Html;

use yii\widgets\LinkPager;

use yii\grid\GridView;

use yii\widgets\ActiveForm;

?>

<h1>Инструкция для работы с сервисом:</h1>

<ul>

<li>Загрузите zip архив с html-страницами при помощи кнопки "Выберите файл".</li>

<li> Нажмите на кнопку "Send" для проверки архива.</li>

<li>Флаг Flag Of Archive: 0 - в загруженном архиве найдены страницы с плохой читаемостью текста.

<br>Флаг Flag Of Archive: 1 - в загруженном архиве не найдены страницы с плохой читаемостью. Это обозначает, что, либо, их там нет, либо возможности алгоритма пока не позволяют

их распознать. Обратите внимание на ограничения внизу страницы.</li>

<li>Чтобы просмотреть список найденных в архиве страниц, нажмите на изображение "Глаз" в крайнем левом столбце страницы.</li>

<li>Чтобы увидеть, какие страницы не были рапознаны, взгляните в поле "Консоль ошибок" на этой странице.</li>

</ul><br>

<h1>Форма для загрузки архива с html-страницами</h1>

<?php $form = ActiveForm::begin(['options' => ['enctype' => 'multipart/form-data']]) ?>

<?= $form->field($up\_model, 'zipArchive')->fileInput() ?>

<?= Html::submitButton('Send', ['class' => 'btn btn-success']) ?>

<?php ActiveForm::end() ?>

<h1>Таблица загруженных архивов</h1>

<?= GridView::widget([

'dataProvider' => $dataProvider,

'columns' => [

/\* отображение нумерации колонок \*/

['class' => 'yii\grid\SerialColumn'],

//'id\_archive',

'address\_archive',

'address\_folder',

'flag\_of\_archive',

/\* отображение панели "просмотреть-удалить-изменить \*/

['class' => 'yii\grid\ActionColumn'],

],

]); ?>

<br>

<h1>При использовании сервиса следует учитывать следующие ограничения:</h1>

<ul>

<li>Сравнение цвета фона и текста происходит по алгоритму ... </li>

<li>Сравнение цвета фона и текста ведётся только в файлах css.</li>

<li>Поиск файлов css ведётся только в файлах html, htmls.</li>

<li>Файлы html, htmls с ошибками не записываются в базу данных.</li>

<li>Если ссылка на css не верна, или файл не открывается, он так же не записывается в базу данных.</li>

<li>Названия всех файлов обязательно должны быть на латинице, иначе скрипт пометит их как не существующие.</li>

<li>Если числа в полях background-color и color проверяемого селектора не соответствуют шаблону '#XXXXXX' или '#XXX',

где X - число от 0 до F(f), пара будет промаркирована низкой контрастностью.</li>

<li>В сервисе не реализована возможность удаления файлов html и архива, так как процесс записи информации в несколько связанных таблиц базы данных требует доработки.</li>

<li>В сервисе не реализована возможность возврата на предыдущие страницы. На данный момент её можно осуществить только кнопкой "Назад" браузера. - ошибка исправлена [23.10.2020]</li>

<li>Если файлы архивировались папкой, а не по отдельности, название папки должно сопадать с названием архива. Ссылки на файлы css в папке, названной отлично от архива,

будут опознаны и записаны, как несуществующие и выдадут ошибку скрипта. - ошибка исправлена [20.10.2020]</li>

</ul><br>

<h1>Консоль ошибок:</h1>

<?php

echo "<pre>";

print\_r($ArrayOfErrors);

//print\_r($AllColorAndBacks);

//print\_r($test1);

//print\_r($test2);

//print\_r($test3);

//echo "</pre><br>";

?>

**Addresspages/index.php:**

<?php

use yii\helpers\Html;

use yii\widgets\LinkPager;

use yii\grid\GridView;

//use yii\widgets\ActiveForm;

?>

<h1>Инструкция для работы с сервисом:</h1>

<ul>

<li>Вы находитесь на странице, где расположены страницы выбранного архива.</li>

<li>Чтобы просмотреть список распознанных селекторов для кажой страницы, нажмите на изображение "Глаз" в крайнем левом столбце страницы.</li>

<li>Флаг Flag Of Page: 0 - в загруженной странице html найдены селекторы с плохой контрастностью.

<br>Флаг Flag Of Page: 1 - в загруженной странице html не найдены селекторы с плохой контрастностью. Это обозначает, что, либо, их там нет, либо возможности алгоритма пока не позволяют

их распознать. Обратите внимание на ограничения, список которых приведён внизу начальной страницы.</li>

</ul><br>

<h1>Таблица страниц выбранного архива</h1>

<?= GridView::widget([

'dataProvider' => $dataProvider,

'columns' => [

/\* отображение нумерации колонок \*/

['class' => 'yii\grid\SerialColumn'],

// 'id\_archive',

'address\_page',

'id\_of\_archive',

'flag\_of\_page',

/\* отображение панели "просмотреть-удалить-изменить \*/

['class' => 'yii\grid\ActionColumn'],

],

]); ?>

<?= Html::a('Назад', ['/addressarchives/index'], ['class'=>'btn btn-primary']) ?>

**Addresspageview/index.php:**

<?php

use yii\helpers\Html;

use yii\widgets\LinkPager;

use yii\grid\GridView;

use yii\widgets\ActiveForm;

?>

<h1>Инструкция для работы с сервисом:</h1>

<ul>

<li>Вы находитесь на странице, где расположены распознанные селектора.</li>

<li>В этой таблице:</li>

<li>Число delta: - коэффициент контрастности, вычисляемый алгоритмом, согласно стандарту ... .</li>

<li>Флаг Flag Of Colors Good: 1 - контрастность данного селектора достаточна для людей с хорошим зрением.</li>

<li>Флаг Flag Of Colors Def: 1 - контрастность данного селектора достаточна для людей с дефекатами зрения (близорукость, дальнозоркость).</li>

<li>Флаг Flag Of Colors Contr: 1 - контрастность данного селектора достаточна для людей с нарушением восприятия контрастности.</li>

<br>Если страница пуста, то в загруженной странице html не найдены селекторы с плохой контрастностью. Это обозначает, что, либо, их там нет, либо возможности алгоритма пока не позволяют

их распознать. Обратите внимание на ограничения, список которых приведён внизу начальной страницы.</li>

</ul><br>

<h1>Таблица селекторов выбранной html-страницы:</h1>

<div>

<table class='table table-striped table-bordered table-hover'>

<thead>

<td>№</td>

<td>class</td>

<td>text color</td>

<td>back color</td>

<td>delta</td>

<td>flag of colors good</td>

<td>flag of colors def</td>

<td>flag of colors contr</td>

</thead>

<?php

if (isset($query)) {

$i = 1;

foreach ($query as $column) {

//echo '<tr><td>'.$column['id\_class'].'</td>';

echo '<tr><td>'.$i.'</td>';

echo '<td>'.$column['class'].'</td>';

echo '<td>'.$column['text\_color'].'</td>';

echo '<td>'.$column['back\_color'].'</td>';

echo '<td>'.$column['delta'].'</td>';

echo '<td>'.$column['flag\_of\_colors\_good'].'</td>';

echo '<td>'.$column['flag\_of\_colors\_def'].'</td>';

echo '<td>'.$column['flag\_of\_colors\_contr'].'</td></tr>';

$i++;

}

}

?>

**html\_dom\_parsing.php**

<?php

namespace app\controllers;

include 'simple\_html\_dom.php';

include 'cssparser\_v2-1.php';

//include './scripts/simplehtmldom/simple\_html\_dom.php';

//include './scripts/cssparser-2014-08-08/v2.1/cssparser\_v2-1.php';

include 'SRGB\_func.php';

class HtmlAndCssParsing {

private $html; /\* Объект парсинга html-страницы \*/

private $e\_css; /\* Массив со ссылками на css для данной страницы \*/

private $css; /\* Объект парсинга css-страницы \*/

/\* конструктор \*/

public function \_\_construct()

{

//echo "<br>I'm alive!<br>";

}

/\* деструктор \*/

public function \_\_destruct()

{

//echo "<br>I'm dead now :(<br>";

}

/\* Получение со страницы всех адресов css \*/

public function getCSSadresses($address) {

$this->html = file\_get\_html($address);

$this->e\_css = $this->html->find('link[type="text/css"]'); /\* Находит все теги с ссылками на файлы css \*/

foreach ($this->e\_css as $e\_key => $e\_element) {

$e\_href[$e\_key] = $e\_element->href;

}

if (!isset($e\_href)) {

$e\_href = null;

}

return $e\_href;

}

public function getCSSObjects($e\_href, $str) {

$e\_count = count($e\_href); /\* количество элементов массива \*/

$e\_number = 0; /\* текущий элемент массива \*/

$e\_key\_number = 0; /\* текущий элемент массива \*/

$AllColorAndBack = Null;

foreach ($e\_href as $e\_key => $e\_element) {

$ColorAndBack = $this->getCSSObject($e\_element, 2, $str); //???

if (isset($ColorAndBack)) {

$AllColorAndBack[$e\_number] = $ColorAndBack;

$e\_number++;

}

$ColorAndBack = Null;

}

return $AllColorAndBack;

}

public function getSearchFromCSSs($AllColorAndBack) {

foreach ($AllColorAndBack as $e\_key => $e\_element) {

$AllColorAndBack[$e\_key] = $this->getSearchFromCSS($e\_element);

}

return $AllColorAndBack;

}

public function genParseHtmlAndCss($AllColorAndBack) {

foreach($AllColorAndBack as $ColorAndBack) {

foreach($ColorAndBack as $ColorAndBack\_elements) {

/\* перебор перечисленных элементов - через запятую \*/

foreach($ColorAndBack\_elements[0] as $ColorAndBack\_element) {

/\* перебор элементов, объединённых пробелом \*/

foreach ($ColorAndBack\_element as $element) {

echo $element.'<br>';

}

}

}

}

}

private function getSearchFromCSS($ColorAndBack) {

foreach ($ColorAndBack as $key => $element) {

$testPieces = explode(',', $key);

$testPiecesNumber = count($testPieces);

for ($i = 0; $i < $testPiecesNumber; $i++) {

$testPieces[$i] = explode(' ', $testPieces[$i]);

}

$ColorAndBack[$key] = array();

$ColorAndBack[$key][0] = $testPieces;

$ColorAndBack[$key][1] = $element;

}

return $ColorAndBack;

}

private function getCSSObject($e\_href\_element, $flag\_color\_back, $str) { /\* 2 - ColorAndBack, 1 - ColorNoBack, 0 - NoColorAndBack \*/

$this->css = new CSSparser();

//$str = 'K:/Server/data/htdocs/nikita/views/addressarchives/folders\_of\_archives/';

/\* Создание абсолютного пути для css \*/

/\* проверка на соответствие \*/

/\*Находим последние вхождения .zip в путях архива\*/

/\* $number = strpos($e\_href\_element, '//');

if (($number == 0) || ($number > 0)) {

$ColorAndBack = Null;

} \*/

$length = strlen($e\_href\_element);

$way = substr($e\_href\_element, 2, $length - 1); //echo $way.'<br>';

$css\_file = file\_get\_contents($str.$way);

$index = $this->css->ParseCSS($css\_file);

/\* Получение всех поддерживаемых типов \*/

$media = $this->css->GetMediaList($index);

/\* Пока только для media0 - all - теги для всех отображаемых типов устройств \*/

$tree = $this->css->GetCSSArray($index, $media[0]);

$ColorAndBack = array();

foreach ($tree as $tree\_element) {

/\* поиск по ключу \*/

$flag\_color = array\_key\_exists('color', $tree\_element);

$flag\_background = array\_key\_exists('background-color', $tree\_element);

$current\_key = key($tree); /\* Получаем текущий класс \*/

if ( ($flag\_color == true) && ($flag\_background == true) && ($flag\_color\_back == 2) ) {

$color = $tree\_element['color'];

$background = $tree\_element['background-color'];

$str = array();

array\_push($str, $color);

array\_push($str, $background);

$ColorAndBack[$current\_key] = $str;

} else if ( ($flag\_color == true) && ($flag\_background == false) && ($flag\_color\_back == 1) ) {

$color = $tree\_element['color'];

$str = array();

array\_push($str, $color);

$ColorAndBack[$current\_key] = $str;

} else if ( ($flag\_color == false) && ($flag\_background == true) && ($flag\_color\_back == 0) ) {

$background = $tree\_element['background-color'];

$str = array();

array\_push($str, $background);

$ColorAndBack[$current\_key] = $str;

}

next($tree);

}

return $ColorAndBack;

}

}

?>

**SRGB\_func.php**

<?php

namespace app\controllers;

class SRGBclass {

public $arrayOferrors;

/\* конструктор \*/

public function \_\_construct()

{

//echo "I'm alive!<br>";

}

/\* деструктор \*/

public function \_\_destruct()

{

//echo "I'm dead now :(<br>";

}

/\* высчитать критерий контрастности для текста или фона \*/

private function SRGBcolor ($color, &$L) {

/\* Поправка на длину входного числа и знак # \*/

$length = strlen($color);

$color = substr($color, 1, $length - 1);

$mod = fmod($length - 1, 2);

if (!$mod)

$div = 2;

else

$div = 1;

$color\_array = str\_split($color, $div);

//echo '<pre>';

//print\_r($color\_array);

//echo '</pre>';

foreach ($color\_array as $color\_key => $color\_el) {

//echo "color\_key=".$color\_key."; color\_el=".$color\_el."<br>";

/\* hexdec - перевод из шестнадцатеричной системы в двоичную \*/

$color\_dec = hexdec($color\_el)/255; //echo $shr.'<br>';

if ($color\_dec <= 0.0328) {

$color\_dec = $color\_dec/12.98;

} else {

$color\_dec = pow($color\_dec + 0.055, 2.4)/1.055;

}

$color\_dec\_array[$color\_key] = $color\_dec;

}

$L = 0.2126 \* $color\_dec\_array[0] + 0.7152 \* $color\_dec\_array[1] + 0.0722 \* $color\_dec\_array[2];

//echo 'L='.$L.'<br>';

}

/\* высчитать разницу между критериями контрастности для текста и фона \*/

private function SRGBdelta ($color, $backgroundcolor, &$delta) {

$L1 = 0;

$L2 = 0;

/\*

Для фона и текста, обозначения вместо цифр фона:

background-color: <цвет> | transparent | inherit

tranceparent - прозрачный

inherit - наследует фон родителя

\*/

$answer1 = $this->verifyColor($color);

$answer2 = $this->verifyColor($color);

if ( ($answer1) && ($answer2) ) {

$this->SRGBcolor ($color, $L1);

$this->SRGBcolor ($backgroundcolor, $L2);

}

/\*В случае несоответствия одного из вводимых чисел шаблону, сравнения не происходит и в базу данных записываются все 0, то есть контрастность слишком мала \*/

$delta = ($L1 + 0.05)/($L2 + 0.05);

}

/\* выдать результат в зависимости от величины разницы контрастностей \*/

public function compareColorAndBack ($color, $backgroundcolor, &$answer) {

/\* Порог \*/

$threshold = array

(

1 => 3,

2 => 4.5,

3 => 7

);

/\* Тип ответа \*/

$answer\_type = array

(

1 => 'с хорошим зрением',

2 => 'с дефектами зрения',

3 => 'с потерей контрастной чувствительности'

);

$delta = 0;

$this->SRGBdelta ($color, $backgroundcolor, $delta);

$conteiner ='delta = '.$delta.'<br>';

$flag\_contrast = true; /\* Контраст значителен (больше 1)\*/

foreach ($threshold as $threshold\_key => $threshold\_element) {

if ($delta <= $threshold\_element) {

$conteiner .= $threshold\_key.'. Контрастность слишком мала для людей '.$answer\_type[$threshold\_key].'!<br>';

if ($threshold\_key == 1) {

$flag\_contrast = false;

}

} else {

$conteiner .= ' контрастность для людей '.$answer\_type[threshold\_key].' достаточна!<br>';

}

}

$answer = $conteiner;

return $flag\_contrast; /\* Флаг показывает, значителен ли контраст \*/

}

/\* выдать результат в зависимости от величины разницы контрастностей, пригодный для записи в базу данных \*/

public function compareColorAndBackForDataBase ($color, $backgroundcolor, &$answer) {

/\* Порог \*/

$threshold = array

(

1 => 3,

2 => 4.5,

3 => 7

);

/\* Тип ответа \*/

$answer\_type = array

(

1 => 'с хорошим зрением',

2 => 'с дефектами зрения',

3 => 'с потерей контрастной чувствительности'

);

$delta = 0;

$this->SRGBdelta ($color, $backgroundcolor, $delta);

//$conteiner ='delta = '.$delta.'<br>';

$conteiner = null;

$flag\_contrast = true; /\* Контраст значителен (больше 1)\*/

foreach ($threshold as $threshold\_key => $threshold\_element) {

if ($delta <= $threshold\_element) {

$conteiner[ $answer\_type[$threshold\_key] ] = 0; /\* Контрастность слишком мала \*/

if ($threshold\_key == 1) {

$flag\_contrast = false;

}

} else {

$conteiner[ $answer\_type[$threshold\_key] ] = 1; /\* Контрастность достаточна \*/

}

}

$conteiner[ 'delta' ] = $delta;

$answer = $conteiner;

//return $flag\_contrast; /\* Флаг показывает, значителен ли контраст \*/

}

/\* Проверка на соответствие шаблону шестнадцатеричного числа \*/

public function verifyColor($color) {

$verify = '((#[0-9a-fA-F]{6})|(#[0-9a-fA-F]{3}))';

$answer = preg\_match($verify, $color);

return $answer;

}

}

/\*

$color = '333333';

$backgroundcolor = 'FFFFFF';

$sRGBcl = new SRGBclass();

$flag = $sRGBcl->compareColorAndBack ($color, $backgroundcolor, $answer);

echo $answer;

if (!$flag) {

echo '<< Контрастность слишком незначительна. Текст сливается! >> <br>';

}

\*/

?>

**UploadForm.php**

<?php

namespace app\models;

use yii\base\Model;

use yii\web\UploadedFile;

class UploadForm extends Model

{

/\*\*

\* @var UploadedFile

\*/

public $zipArchive;

public $downloadWay = 'K:/Server/data/htdocs/nikita/views/addressarchives/archives';

public $unzipWay = 'K:/Server/data/htdocs/nikita/views/addressarchives/folders\_of\_archives';

public $zipName;

public $zipNoFormat;

public $htmls;

public $cut;

/\* Метод, возвращающий правила обработки данных.

Обозначает - нужно проверить переменную zipArchive, её тип file, пропустить его, если пуст, со связями 'zip'\*/

public function rules()

{

return [

[['zipArchive'], 'file', 'skipOnEmpty' => false, 'extensions' => 'zip'],

];

}

/\* Функция загрузки зип-архива на сервер \*/

public function upload()

{

$this->zipName = $this->zipArchive->baseName . '.' . $this->zipArchive->extension;

if ($this->validate()) {

$this->zipArchive->saveAs($this->downloadWay . $this->zipName);

return true;

} else {

return false;

}

}

/\* Функция распаковки зип-архива \*/

public function unzip()

{

/\* \ - принудительно вызывает ZipArchive из стандартного пространства имён PHP, так как пространство имён у YII собсвенное, и по умолчанию операции происходят только в нём \*/

$zip = new \ZipArchive;

$res = $zip->open($this->downloadWay . $this->zipName);

/\*Находим последние вхождения .zip в путях архива\*/

$number = strrpos($this->zipName, '.zip'); //echo " number=".$number."<br>";

$this->zipNoFormat = substr($this->zipName, 0, $number); //echo " zipNoFormat=".$zipNoFormat."<br>";

/\*Создать папку\*/

$filename = $this->unzipWay .'/'. $this->zipNoFormat .'/';

if (!file\_exists($filename)) {

if (!mkdir($filename, 0777, true)) {

die('Не удалось создать директорию...');

}

}

if ($res === TRUE) {

$zip->extractTo($this->unzipWay .'/'. $this->zipNoFormat .'/');

$zip->close();

//echo 'ok';

return true;

} else {

//echo 'failed';

return false;

}

}

/\* Функция распаковки зип-архива \*/

public function unzip\_alternative()

{

exec('K:/Server/optionsprograms/7zip\_portable e '.$this->downloadWay . $this->zipName.' -o'.$this->unzipWay .'/'. $this->zipNoFormat .'/');

return true;

}

/\* Функция преобразования кодировки распакованного архива \*/

public function uncode()

{

/\*Создать папку\*/

$filename = $this->unzipWay .'/'. $this->zipNoFormat .'/';

$files = scandir($filename);

foreach ($files as $files\_el)

{

iconv('Windows-1252', 'utf-8', $files\_el).PHP\_EOL;

}

return true;

}

/\* Создание списка файлов html \*/

public function htmllist()

{

$zip = new \ZipArchive;

//$htmls = array ( array ( 1 => 'test1', 2 => 'testway1'), array ( 1 => 'test2', 2 => 'testway2'));

$res = $zip->open($this->downloadWay . $this->zipName);

//$this->cut[] = '/';

//$this->cut = array ( 0 => '/', 1 => '.html', 2 => '.htmls', );

if ($res === TRUE) {

for($i = 0; $i < $zip->numFiles; $i++)

{

//echo 'Filename: ' . $zip->getNameIndex($i) . '<br />';

$getNI = $zip->getNameIndex($i);

$getCount = strlen($getNI);

$cutArrayCount = count($this->cut);

for ($j = 1; $j < $cutArrayCount; $j++)

{

/\*Находим последние вхождения .html и .htmls в путях архива\*/

$number = strrpos($getNI, $this->cut[$j]);

/\*Находим длину проверяемой подстроки с расширением файла страницы\*/

$cutСount = strlen($this->cut[$j]);

/\*находим разность между длинной входной строки и длинной последнего вхождения пути .html и .htmls в архиве \*/

$numberLen = $getCount - $number;

if ($cutСount == $numberLen) {

/\*Если нашли, то находим вхождение последнего / \*/

$number = strrpos($getNI, $this->cut[0]);

if ($number) {

$getNAME = substr($getNI, $number + 1);

} else {

$getNAME = $getNI;

}

/\* Добавление элемента в конец массива \*/

//array\_push($this->htmls, array(0 => "test.zip", 1 => "C:/text") ); //array ( 0 => $getNAME, 1 => $this->unzipWay . $getNI) ); //echo " ".$htmls[0]." ".$htmls[1];

$this->htmls[] = array ( 0 => $getNAME, 1 => $this->unzipWay .'/'. $this->zipNoFormat .'/'. $getNI ); //array(0 => "test.zip", 1 => "C:/text"); //array ( 0 => $getNAME, 1 => $this->unzipWay . $getNI) ); //echo " ".$htmls[0]." ".$htmls[1];

}

}

} //echo '<pre>'; print\_r($this->htmls); echo '</pre>';//$this->htmls = array ( array(0 => "test.zip", 1 => "C:/text"), array(0 => "test.zip", 1 => "C:/text") );

return true;

} else {

//echo 'Error reading zip-archive!';

return false;

}

//return $htmls;

}

/\* Парсинг файлов html \*/

public function htmlparsing()

{

}

}

?>