МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ДЕРЖАВНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД

«УЖГОРОДСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ»

ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра програмного забезпечення систем

МАТРАФАЙЛО МИРОСЛАВ ВАСИЛЬОВИЧ

**РОЗРОБКА МЕТОДІВ ПОШУКУ ПОСЛІДОВНОСТЕЙ ЗОБРАЖЕНЬ В БАЗАХ ВІДЕОДАНИХ**

6.050103 «Програмна інженерія»

Дипломна робота на здобуття освітнього ступеня

бакалавра

Науковий керівник:

**проф. Машталір В.П.**

доктор технічних наук, професор кафедри інформатики Харківського національного університету радіоелектроніки (ХНУРЕ)

Ужгород – 2017

**Реєстрація \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(номер)

«\_\_\_» червня 2017 р. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ М.І. Роль

(підпис)

**Дипломна робота допущена до захисту:**

В.о. завідувача кафедри

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ проф. Головач Й.І.

(підпис)

«\_\_\_» червня 2017 р.

**ПІБ:** МАТРАФАЙЛО МИРОСЛАВ ВАСИЛЬОВИЧ

**Назва:**РОЗРОБКА МЕТОДІВ ПОШУКУ ПОСЛІДОВНОСТЕЙ ЗОБРАЖЕНЬ В БАЗАХ ВІДЕОДАНИХ

**Факультет:** інформаційних технологій

**Напрям підготовки:** 6.050103 – програмна інженерія

**Науковий керівник:**

По даній темі подано на публікацію 0 статей

Анотація

Дана дипломна робота складається з трьох розділів. У першому розділі наведено теоретичні відомості, які стосуються методики пошуку зображень. описані основні поняття алгоритмів пошуку зображень за зразком в базах даних і синтез методів індексації в метричному просторі для забезпечення швидкого пошуку зображень.

.В другому розділі описано методику розробки програмного продукту, а також мови програмування за допомогою яких виконувалась робота. В третьому розділі особливу увагу приділено актуальності роботи, детально описано структуру веб - ресурсу та етапи його розробки. Крім того наведено інструкцію для користувача та проаналізовано один з існуючих аналогів.

Дипломна робота містить в собі 66 сторінок, 3 розділи із поділом на підрозділи, 1 додаток, 10 рисунків.

Ключові слова: індексація, метрика, зображення, база даних.

**Список умовних скорочень.**

JavaScript – назва реалізації стандарту мови програмування ECMAScript компанії Netscape, базується на принципах прототипного програмування.

jQuery – популярна JavaScript-бібліотека з відкритим сирцевим кодом.

HTML (HyperText Markup Language) – стандартна мова розмітки веб-сторінок в Інтернеті.

CSS (Cascading Style Sheets) – спеціальна мова, що використовується для відображення сторінок, написаних мовами розмітки даних.

PHP (Hypertext Preprocessor) – скриптова мова програмування, була створена для генерації HTML-сторінок на стороні веб-сервера.

**ЗМІСТ**

ВСТУП

РОЗДІЛ 1 АНАЛІЗ МЕТОДІВ ПОШУКУ ЗОБРАЖЕНЬ 11

1.1 Дослідження основних підходів до пошуку зображень 11

1.2 Порівняльний аналіз моделей пошуку зображень 15

1.3 Аналіз методів індексації зображень 20

1.4 Теоретичні основи метричного пошуку зображень 24

1.5 Постановка завдання 25

РОЗДІЛ 2 ІНСТРУМЕНТИ РОЗРОБКИ 26

2.1 JavaScript 26

2.2 HTML 27

2.3 CSS 28

2.4 PHP 28

2.5 Symfony 30

РОЗДІЛ 3. ПРАКТИЧНА ЧАСТИНА 32

3.1 Структура веб-ресурсу 32

3.2 Етапи розробки проекту 37

3.3. Огляд аналогів 38

ВИСНОВКИ 39

Список використаних джерел 41

Додаток 1 Лістинг програми

**ВСТУП**

Актуальність даного завдання обумовлена зростанням розмірів колекцій мультимедіа даних в різних сферах людської діяльності, що призводить до численних питань забезпечення ефективного зберігання, стиснення, інтерпретації і, в тому числі, забезпечення швидкого пошуку.

Метою є дослідження алгоритмів пошуку зображень за зразком в базах даних і синтез методів індексації в метричному просторі для забезпечення швидкого пошуку зображень.

Об'єктом дослідження є система управління базами даних зображень.

Предметом дослідження є методи метричної індексації вмісту баз відеоданих на базі метричного порівняння результатів сегментації зображень.

Проблема пошуку набору елементів, які близькі до даного, запитуваного, елементу за деякими критеріями подібності є на сьогоднішній день надзвичайно актуальною. Для зберігання інформації в усіх сучасних аналізаторах зображень передбачено використання бази даних. База даних візуальної інформації є програмне забезпечення, призначене для накопичення, зберігання, архівування інформації, її систематизації і подальшого швидкого пошуку зображень, подібних заданому. Це дозволяє вирішувати різні прикладні завдання, пов'язані з обробкою і розпізнаванням зображень.

Для вирішення проблеми забезпечення швидкого пошуку здійснюється попередня обробка бази даних, яка включає в себе побудову індексної структури на базі аксіоми нерівності трикутника метричного простору, мета якої полягає в зменшенні витрат часу пошуку в порівнянні з повним перебором. Ефективність індексної структури оцінюється за такими параметрами, як кількість звернень до пам'яті, витрати на пошук по індексного структурі і кількість операцій обчислення відстані між поданням запиту і об'єктами бази даних. Очевидно, що в разі пошуку зображень має місце порівняння ознак зображень високої розмірності. Крім цього, витрати на порівняння об'єктів зростають в разі використання складних метрик, спрямованих на порівняння високорівневих концепцій. Як приклад слід привести значне поширення метрик, що дозволяють порівнювати результати сегментації. Безсумнівно, прагнення до усунення семантичного конфлікту призводить до все більш і більш широкого використання методів пошуку відеоданих по областям, а значить, до більш складних метрик порівняння об'єктів. Таким чином, для створення ефективної індексної структури для баз даних, що містять зображення, слід в першу чергу акцентувати увагу на методах, які краще інших мінімізують кількість операцій обчислення відстані.

Випадок, коли схожість визначається в деякому метричному просторі, найбільш цікавий в прикладному аспекті. В результаті оцінки результатів сегментації - розбиття поля зору - двох зображень маємо скалярную величину відстані, яка не дає інформації про позицію об'єктів у векторному просторі, а значить, робить неможливим облік геометричних властивостей і координат, але дозволяє оцінювати відносність розташування при пошуку схожих об'єктів. Далі, часто операція визначення відстані є обчислювально витратною, особливо в разі пошуку по областях, наприклад, відстань Earth Mover's Distance. Таким чином, високу актуальність набувають методи індексації даних в метричному просторі, що дозволяють на підставі інформації про взаємні відстанях між об'єктами бази даних скорочувати кількість операцій порівнянь, тобто уникати лінійного пошуку.

Слід розпочати з того, що сучасний користувач всесвітньої мережі Internet є надзвичайно вибагливий, а оскільки майже кожен розробник будь-якого веб-ресурсу бажає сповна догодити цьому користувачу, все більше і більше сайтів стають інтерактивним, краще графічно оформленими та зручнішими у використання. Для того, щоб на розроблюваному сайті все більше і більше елементів могли напряму взаємодіяти з гостем, часто застосовуються можливості CSS та JavaScript, що дозволяють більш «оживити» сторінку, додати анімацію на неї, зробити її динамічною.

За допомогою розробленого додатку можна завантажувати, переглядати та шукати зображення. Користувач може створювати власні бібліотеки та редагувати їх вміст. Великою перевагою розробленого веб-ресурсу є те, що він має легкий та зрозумілий інтерфейс, не потребує великих затрат часу на освоєння та може використуватися будь-ким незалежно від рівня володіння комп’ютером та знаннями в сфері роботи з подібними додатками.

Розроблюваний проект являє собою повноцінне середовище, де знаходяться різного роду зображення, наповнене різноманітними функціями та підтримуючи при цьому можливість доповнення згадуваного функціоналу та додання нових рівнів та складових.

Проте, не беручи до уваги той факт, що використовуючи можливості PHP та фреймворку Symfony можна створити практично будь-яку базу із зображеннями, організувати пошук в ній, але задля кращого зовнішнього вигляду, покращенню структури та функціоналу застосовується ряд сторонніх інструментів, таких як HTML, CSS

Актуальність даного завдання обумовлена ​зростанням розмірів колекцій мультимедіа даних в різних сферах людської діяльності, що призводить до численних питань забезпечення ефективного зберігання, стиснення, інтерпретації і, в тому числі, забезпечення швидкого пошуку.

Метою є дослідження алгоритмів пошуку зображень за зразком в базах даних і синтез методів індексації в метричному просторі для забезпечення швидкого пошуку зображень.Об'єктом дослідження є система управління базами даних зображень.

Предметом дослідження є методи метричної індексації вмісту баз відеоданих на базі метричного порівняння результатів сегментації зображень.

Проблема пошуку набору елементів, які близькі до даного, запитуваного, елементу за деякими критеріями подібності є на сьогоднішній день надзвичайно актуальною. Для зберігання інформації в усіх сучасних аналізаторах зображень передбачено використання бази даних. База даних візуальної інформації перед-ставлять собою програмне забезпечення, призначене для накопичення, зберігання, архівування інформації, її систематизації і подальшого швидкого пошуку з-браженій, подібних заданому. Це дозволяє вирішувати різні прикладні завдання, пов'язані з обробкою і розпізнаванням зображень.

Для вирішення проблеми забезпечення швидкого пошуку здійснюється попередньо-кові обробка бази даних, яка включає в себе побудову індексного структури на базі аксіоми нерівності трикутника метричного простору, мета якої полягає в зменшенні витрат часу пошуку в порівнянні з повним перебором. Ефективність індексної структури оцінюється за такими параметрами, як кількість звернень до пам'яті, витрати на пошук по індексного структурі і кількість операцій обчислення відстані між поданням запиту і об'єктами бази даних. Очевидно, що в разі пошуку зображень має місце порівняння описів високої розмірності. Крім цього, витрати на порівняння об'єктів зростають в разі використання складних метрик, спрямованих на порівняння високорівневих концепцій. Як приклад слід привести отримане значне поширення метрик, що дозволяють порівнювати результати сегментації.

Безсумнівно, прагнення до усунення семантичного конфлікту призводить до все більш і більш широкого використання методів пошуку відеоданих по областям, а значить, до більш складним метрик порівняння об'єктів. Таким чином, для створення ефективної індексної структури для баз даних, що містять зображення, слід в першу чергу акцентувати увагу на методах, які краще інших мінімізують кількість операцій обчислення відстані.

Випадок, коли схожість визначається в деякому метричному просторі, найбільш цікавий в прикладному аспекті. В результаті оцінки результатів сегментації  розбиття поля зору  двох зображень маємо скалярну величину відстані, яка не дає інформації про позицію об'єктів у векторному просторі, а значить, робить неможливим облік геометричних властивостей і координат, але дозволяє оцінювати відносність розташування при пошуку схожих об'єктів.

Далі, часто операція ділення відстані є обчислювально - витратною, особливо коли ви шукаєте вздовж області, наприклад, відстань Earth Mover's Distance.

Таким чином, високу актуальність набувають методи індексації даних в метричному просторі, на підставі інформації про взаємні відстанях між об'єктами бази даних скорочувати кількість операцій порівнянь, тобто уникати лінійного пошуку.

**РОЗДІЛ 1**

**АНАЛІЗ МЕТОДІВ ПОШУКУ ЗОБРАЖЕНЬ**

* 1. **Дослідження основних підходів до пошуку зображень.**

Зараз неможливо переоцінити значимість і в той же час повсякденність використання мультимедіа даних. З виникненням цифрових камер і фотоапаратів, розширенням функціональних можливостей ЕОМ створення індивідуальних колекцій відео і зображень стало доступно кожному. Зростаючі запити користувачів в плані якості контенту та більшого обсягу пам'яті носіїв задовольняється, як правило, за рахунок інтенсивного розвитку сервісних програмно-апаратних засобів. Як результат, обсяги реєстрованих відеоданих стають все більше, а, отже, ними все важче управляти.

Традиційним підходом до зберігання і управління даними, зокрема зображеннями, є використання реляційних баз даних [3]. У загальному випадку в базу даних заноситься зображення і інформація про нього, наприклад, назва, дата створення, опис. Далі для пошуку і перегляду даних використовується мова запитів SQL, який дозволяє виконувати пошук і сортування результатів за певними кількісними критеріями. Слід зазначити, що якщо досить невеликі архіви відеоданих можна впорядкувати за датою, категоріями, важливості, то, наприклад, для колекцій медичних зображень, які можуть містити мільйони записів, даний підхід є малоефективним. Більш того, у міру розширення спектра джерел цифрових даних Користувач все частіше стикається з архівами з малознайомим тематичним змістом. Звідси досить очевидна необхідність в нових підходах до управління архівами відеоданих.

Існує два принципово різних підходи до реалізації систем пошуку відеоданих. У 1970-х роках набув поширення метод текстового опису зображень людиною-оператором . Запит задається ключовими словами і тим самим зводиться до задачі пошуку в базах даних, що містять текст. Очевидно, що даний підхід має два принципових нестачі. По-перше, зростаючий обсяг мультимедіа даних робить практично неможливим опис оператором всіх реєстрованих даних. По-друге, суб'єктивність сприйняття породжує значно різняться анотації одного і того ж зображення, що прямим чином відбивається на якості пошуку і перегляду.

Досить природним здається підхід, вперше сформульований Чангом в якому пропонується отримувати опис для індексації і структурування вмісту баз відеоданих з самого змісту зображень. Під «змістом» тут слід розуміти певний набір візуальних ознак, що витягають із зображення або його частин. Тоді при завданні запиту за допомогою зображення-зразка пошук виконується в певному просторі ознак по заданій мірі подібності. В останні десятиліття цей підхід набув значного поширення і практично повністю витіснив методи «ручного» текстового опису. Зростаюча актуальність проблеми пошуку зображень привернула дослідників з різних областей науки, в їх числі комп'ютерне зір, навчання машин, інформаційний пошук, взаємодія комп'ютера і людини, системи баз даних, теорія інформації, статистика, психологія. Як результат, сучасні системи пошуку можна вважати системами штучного інтелекту.

Різноманіття областей використання, цілей Користувача, способів завдання запиту визначає архітектуру системи пошуку за змістом. Були зроблені спроби класифікації систем за цими критеріями. Прийнято розділяти джерела архівів відеоданих на предметно-орієнтовані і універсальні області. Проблемно-орієнтована область характеризується обмеженим і тематично схожим набором даних, що належать до певної предметної області, а також незначними змінами умов реєстрації зображень. Найчастіше це дає можливість отримувати прізнаковие опису зображень, що відображають приховану в них семантику. До колекціям зображень з предметно-орієнтованої області можна віднести медичні, супутникові зображення, зображення відбитків пальців, товарних знаків і ін. Архіви відеоданих з універсальних областей характерні широкої тематичної інтерпретацією зображень, отриманих, можливо, в умовах геометричних спотворень, при різних умовах освітленості і перешкодах . Тут вибір адекватного простору ознак і міри схожості для «розуміння» зображень є надзвичайно складним завданням. Колекції репродукцій музеїв, архіви відеоданих для видавничої справи, інженерного дизайну і реклами, персональні колекції цифрових фотографій і т.п. продукуються універсальними областями. Слід особливо наголосити різнорідні, невпорядковані масиви мультимедіа даних WEB, що вимагають особливих підходів до індексування.

Найбільшого поширення набули такі варіанти завдання запиту:

- Зображення-зразок. Найпоширеніший спосіб. Пошук виконується по всьому полю зору зображення або по областям інтересу. У більшості випадків зображення запиту має бути надано Користувачем або воно може бути вибрано з ансамблю зображень - представників кластерів.

- Замальовка. Користувач, використовуючи сервісний графічний редактор, надає ескіз об'єкта запиту. Тут важливу роль відіграє форма об'єктів, колірні характеристики і їх просторове розташування.

- Чисельні характеристики низькорівневих ознак. Користувач задає візуальні примітиви, які повинні містити зображення.

- Ключові слова. Область використання обмежена колекціями відеоданих з WEB, де враховуються обрамляють медіа ресурс текстові теги з описами (наприклад, пошукові системи Google і AltaVista), а також системами, які виконують анотування зображень з бази даних.

- Комбінований. Найбільш часто поєднують завдання запиту зображенням-зразком з ключовими словами.

Слід зазначити, що системи пошуку зображень з предметно-орієнтованих областей виконують порівняння зображень на базі низькорівневих прізнакових описів, що робить спосіб завдання об'єкта запиту через зображення-зразок найбільш поширеним. Далі, ретроспективний аналіз (мова про нього піде нижче) існуючих систем пошуку відеоданих з універсальних областей показав, що при відсутності достовірних метаданих в колекціях інтелектуальний аналіз візуального змісту зображення-зразка в більшості випадків є найбільш ефективним, і, як наслідок, поширеним на практиці.

Залежно від цілей Користувача розрізняють:

- Пошук по асоціації. На початку пошуку у Користувача немає чіткої мети, крім як знайти цікаві зображення. Пошук складається їх серії незв'язаних за смисловим змістом запитів, що дозволяють переглядати вміст бази даних. Пошукова система повинна містити елементи зворотного зв'язку. Мета пошуку задається замальовкою, зображенням запиту або ключовими словами. Результатом запиту є сімейство зображень, сортувати за релевантністю до об'єкта запиту, або ієрархія зображень, представлена у вигляді дерева.

- Пошук за категоріями. Мета Користувача - знайти цікавий клас зображень з подальшою навігацією в його вмісті. Система пошуку даного типу повинна бути інтерактивною. Запит задається одним або декількома зображеннями, ключовими словами або замальовкою. Результат пошуку організовують і візуалізують у вигляді кластерів, в ідеалі відповідних концептуальним класів зображень.

- Цільовий пошук. Користувач має чітке уявлення про мету пошуку. Запит задається зображенням-образом. Пошук, як правило, здійснюється або в колекціях предметно-орієнтованих областей, або в архівах універсальних областей з метою знаходження дублікатів. Системи даного типу можуть дозволяти керувати вибором прізнакових описів і заходів подібності і мають додаткові можливості по поясненню результатів пошуку, наприклад, по візуалізації числові характеристик подібності.

Візуальні опису зображень для індексування становлять ознаки кольору, текстури, форми, просторового розташування, а також стійкі точки. Кольорові прізнаковие опису використовуються найбільш часто і в більшості випадків аналізуються в однорідних колірних просторах HSI (варіації HSL, HSV) і CIE L \* a \* b. Найпростішим і поширеним ознакою для представлення інформації про колір зображення є гістограма. Для обліку просторових характеристик розподілу кольору були запропоновані вектора когерентності кольору, коррелограмми, автокоррелограмми, зв'язкові коррелограмми.

Текстура описує просторовий розподіл кольору або яскравості на зображенні. Розрізняють три підходи до формулювання і обчисленню цієї ознаки: статистичний, під хід, ознаки Тамуров, засновані на відповідність сприйняття людини, і методи аналізу спектральних характеристик перетворень (вейв лети, фільтри Габора, розкладання в базисі Уолша і ін.).

**1.2 Порівняльний аналіз моделей пошуку зображень**

Зоровий апарат людини здатний швидко виділяти об'єкти сцени і оперувати ними як смисловими поняттями. Очевидно, що для підвищення ступеня «розуміння» зображень при пошуку краще використання локальних прізнакових описів. Цю необхідність підкреслює той факт, що зображення містять кілька об'єктів, що значно відрізняються один від одного по візуальному змістом. У цьому випадку порівняння зображень з глобальних ознаками може призводити до грубих помилок. Отже, досить важливою віхою в процесі подолання смислового розриву є можливість системи пошуку виконувати запити на рівні об'єктів.

В області комп'ютерного зору традиційно застосовують такі алгоритми сегментації: маркування зв'язкових компонент і центроїдного зв'язування, порогові методи і їх удосконалення, наприклад, мультіпороговие методи сегментації, що дозволяють проводити інтелектуальний аналіз відповідності інтервалів яскравості і покриття (розбиття) поля зору, помехоустоучівая порогова сегментація на основі аналізу гістограм в області гіперболічного вейвлет перетворення, сегментація через визначення меж, алгоритми злиття-розщеплення, в иделеніе однорідних за текстурою областей за допомогою марківських ланцюгів. Особливу увагу привертають перешкодостійкі методи сегментації. Зупинимося на комбінованих методах сегментації, які були розроблені для виділення областей при пошуку складних зображень з природними сценами. Розглянемо 7 найбільш поширених алгоритмів, які характеризуються «надмірної» деталізацією індукованого розбиття. Багато розробників нових систем пошуку зображень по областям не вводять новий метод сегментації, а посилаються на один з нижчеперелічених методів.

- Кластеризація методом к-середніх в деякому квантованим колірному просторі і потім - процедура маркування зв'язкових компонент.

- двоетапний кольорово-текстурний алгоритм JSEG . Висока якість і значні часові витрати.

- Алгоритм blobworld, вперше запропонований в реалізації однойменної системи. Виконується кластеризація за допомогою моделі гауссових сумішей і алгоритму максимізації математичного очікування, далі застосовується алгоритм маркування зв'язкових компонент для отримання кінцевого розбиття. Характеристики приблизно такі ж, як у JSEG.

- Метод графового розбиття NCut . Завдання сегментації трактується як завдання розбиття на графі: зображення - це зв'язний граф, в якому вершина - піксель, а ребра - це значення близькості між двома вершинами; результуючі підграфи відповідають областям зображення. Близькість визначають ознаки квіти і текстури. Якість середня, тимчасові витрати значні.

- Алгоритм зсуву медіани. Заснований на перетворенні пікселів зображення до певного простору ознак за допомогою оцінки ядра щільності і виконанні кластеризація в новому просторі. Середня якість і середні обчислювальні витрати.

- агломеративного ієрархічний алгоритм кластеризації, який використовує інформацію про колір і текстурі. Середня якість пошуку і високі обчислювальні витрати.

- графові метод Фелзеншвалба. Є комбінацією методу графового розбиття Ncut і алгоритму зсуву медіани. Відрізняється високою швидкодією і можливістю адаптувати рівень деталізації.

Аналіз сучасних систем пошуку і оглядових статей показав, що найбільш часто використовують алгоритм JSEG і Графова метод сегментації.

Отже, маємо уявлення зображень з бази даних і зображення запиту у вигляді сукупності однорідних в певному просторі ознак областей зв'язності. Візуальне опис будь-якого з зображення в цьому випадку складається з набору локальних прізнакових описів. Очевидна кореляція між природою цільового методу сегментації та набором видобутих візуальних ознак. Найбільш часто використовують ознаки кольору - домінантний колір, моменти, гістограма, текстури - фільтри Габора, вейвлети, а також просторового розташування - центр ваги, координати описує багатокутника і ін. Ймовірна неточність сегментації зображення в деяких випадках є причиною відмови від складання дескрипторів форми (в основному це площа і напрямок основних осей). Ця проблема зумовлює інтерес до таких заходів подібності розбиття поля зору, які б враховували відмінність в ступені деталізації сегментації.

Слід зазначити, що питання порівняння зображень на рівні областей є нетривіальним завданням. Традиційні заходи подібності, описані в попередньому підрозділі, можуть використовуватися як базові відстані між двома областями, але вони не дають можливість порівнювати прізнаковие опису сукупності областей. Існуючі підходи до вирішення цієї дилеми можна розділити на три групи:

- Незалежний пошук найбільшого збігу - для кожної з областей зображення запиту шукається найбільш схожа область зображення з бази даних, загальна величина подібності визначається зваженою сумою отриманих величин. Подібне порівняння виконується в системах NETRA, blobworld, IRMA, PicSOM, ImageRover, AMORE. Істотним недоліком є той факт, що, з огляду на незалежності порівняння областей, відразу кілька елементів розбиття поля зору зображення запиту може бути «пов'язане» з одним елементом розбиття зображення з бази даних.

- Порівняння «один-до-одного» - відрізняється від незалежного пошуку найбільшого збігу тим, що не допускається множинне «зв'язування» областей. Черговість вибору областей визначається їх розміром і відносним розташуванням. У системах Windsurf, Cortina, Istorama реалізований даний підхід. Очевидно, що область використання методів порівняння «один-до-одного» обмежена колекціями зображень, для яких є високоефективні алгоритми сегментації, які продукують приблизно однакову кількість елементів розбиття поля зору цих зображень.

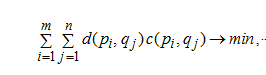
- порівняння «багато-до-багатьох» - при визначенні схожості враховується сукупна близькість порівнюваних наборів областей. Це дозволяє зменшити залежність порівняння від якості алгоритмів сегментації, конкретніше, від рівня деталізації, що використано в системах SIMPLIcity, PARAgrab, MuSIQUE. Зупинимося на методах порівняння даного типу більш детально.

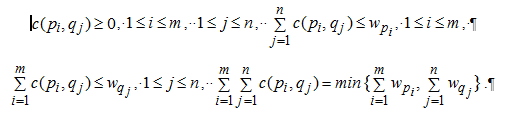
нехай - безліч прізнакових описів зображення з областями і вагою кожної області , рівним , і - набір подібних описів зображення з областями і вагою кожної області , рівним , Виконується умова і - базова міра близькості між ознаковими уявленнями двох областей.

Було запропоновані використовувати відстань Хаусдорфа для порівняння двох сукупностей розбиття поля зору:

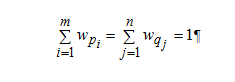
Ступінь близькості обчислюється шляхом визначення для кожної з областей порівнюваних зображень найбільш близькою в плані базової міри близькості області і вибору з отриманих величин найбільшою.

Останнім часом набула значного поширення міра подібності EMD (Earth Mover's Distance), вперше запропонована Рубнером і ін. для порівняння гістограм з різною кількістю відліків. Дана міра подібності при дотриманні певних умов, які будуть вказані нижче, є метрикою. Було показано, що EMD є варіацією метрики Маллоу, використовуваної в статистиці для порівняння імовірнісних розподілів: ступінь їх відмінності визначається мінімальною ціною перетворення одного в інше. Тоді обчислення EMD зводиться до вирішення наступного транспортної задачі:

 (1.1)



Відзначимо, що в обмежувальних нерівностях сформульованої задачі мінімізації передбачена можливість часткового порівняння областей, тобто міру EMD можна застосовувати для пошуку областей інтересу, що лягло в основу міри схожості RMD . Далі, якщо



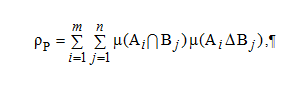
і - метрика, EMD суть метрика.

Метрика EMD широко використовується в системах пошуку зображень по областям . Так, в роботі запропонований спосіб визначення ваги областей і обмеження зверху базової міри схожості між областями для зменшення залежності цільової заходи від ступеня деталізації сегментації.

Очевидним недоліком EMD є великі обчислювальні витрати, необхідні на вирішення транспортної задачі. Запропонована в системі SIMPLIcity, міра подібності IRM (Integrated Region Matching) визначається рішенням завдання за допомогою жодного алгоритму. Забезпечуючи значне зниження складності обчислення, IRM не є метрикою, що кілька звужує область її використання.

Для оцінки двох розбиття поля зору можуть бути використані методи порівняння результатів кластеризації відстань Ван Донгена, метрика Мейл, метрика на розбитті. Тут схожість визначає форма і просторове розташування елементів розбиття, візуальні прізнаковие опису областей не враховуються. Припустимо, що , - розбиття поля зору першого і другого зображення відповідно, - розмірність зображень збігається.

Аналізуючи відстань Ван Донгена, метрику Мейл (варіацію інформації) і метрика на розбитті довільних вимірних множин :



де - позитивна, усюди певна міра, наприклад , - симетрична різниця, віддалимо перевагу останній в силу її універсальності і простоти обчислень.

Прелімінарного метричний аналіз зображень з баз даних на рівні областей створює передумови для створення ефективних індексних структур, спрямованих на скорочення числа операція порівняння на етапі виконання запиту.

**1.3 Аналіз методів індексації зображень**

Експоненціальне зростання кількості архівів мультимедіа даних і їх розмірів істотно звузив область практичного застосування таких систем пошуку зображень, які здійснюють повний перебір на етапі виконання запиту. Багатьма дослідниками наголошувалося на необхідності в методах індексації, тобто в методах попередньої обробки вмісту баз відеоданих, спрямованих на скорочення числа операцій порівняння. Однак лише в небагатьох системах пошуку, що мають практичне застосування, реалізована індексація.

У перших реалізаціях систем пошуку, де порівняння виконувалося на рівні витягнутих -мірних низькорівневих прізнакових описів, найбільшого поширення набули методи просторової індексації (SAM - spatial access methods), широко використовувані для індексації текстових даних: -дерева і їх модифікація -дерева, дерева, лінійні квадродерева і ін. Однак ефективність цих методів значно зменшується зі зростанням розмірності індексованих даних. Дана проблема була формалізована Вебером і отримала назву «прокляття розмірності» (curse of dimentionality): якщо розмірність простору ознак більше 20, то ефективність пошуку з використанням традиційних деревовидних індексних структур рівносильна повного перебору.

Одним з підходів до вирішення проблеми «прокляття розмірності» є попереднє скорочення розмірності індексованих даних. Перелічимо найбільш поширені методи:

- метод багатовимірного масштабування (multidimensional scaling) і його модифікація FastMap, де виконується перетворення елементів з мірного простору в дво- або тривимірне з максимально можливим збереженням величин і пропорцій відстаней;

- метод розкладання по сингулярним числах матриці, адаптований для динамічних баз даних;

- нейромережеві алгоритми, серед яких виділяється архітектура, де використані гібридні гетеро- і автоасоціативна нейтронні мережі.

Інший підхід полягає в адаптації існуючих методів для індексації даних саме високої розмірності. Серед нових методів особливо виділяються:

- X-дерева є удосконалення -дерев, де за допомогою мінімізації кількості розщеплення виключається можливість перетину між граничними областями.

- VA-файл є одночасно методом стиснення прізнакових векторів в так званий файл апроксимації вектора (VA-файл) на етапі прелімінарним обробки і фільтрації на етапі пошуку. Було запропоновано велику кількість удосконалень даного методу.

- i-Distance, де використовується комбінований метод кластеризації і індексування за допомогою дерев. На першому кроці алгоритму виконується розбиття і далі в кожному з отриманих кластерів створюється своя структура індексів за допомогою дерев.

У деяких системах пошуку зображень (Photobook, Istorama, Cortina), спрямованих на підтримку пошуку за категоріями, для зменшення витрат часу на виконання запитів використовують алгоритми кластеризації. Завдання розбиття множини на кластери дозволяє з одного боку оптимізувати пошук, з іншого - виконувати грубу класифікацію даних. Для виконання кластеризації переважно використання алгоритму *k* -середній зважаючи на його обчислювальної ефективності.

У тому випадку, коли міра подібності прізнакових уявлень порівнюваних зображень є метрикою, методи метричної індексації, засновані на аксіомі нерівності трикутника, дозволяють скорочувати кількість операцій порівняння на етапі виконання запиту. Серед найбільш поширених, цитованих і зарекомендували себе на практиці методів слід виділити:

- AESA і LAESA - методи, що зберігають в особливому вигляді обчислені відстані між екземплярами з бази даних, які використовуються на етапі виконання запиту для знаходження нижньої оцінки відстані;

- VP-дерева- метод, який продукує уявлення елементів колекцій у вигляді бінарних дерев, що дозволяє при обробці запиту не враховувати при розгляді зображення, що містяться в одній з гілок;

- MVP-дерева - модифікація VP-дерев, запропоновано використовувати -арні дерева;

- GNAT - метод, в якому на етапі попередньої обробки зображення з бази даних групуються за критерієм близькості навколо елементів обраного безлічі «ключових» об'єктів, утворюючи розбиття метричного простору, вводиться і обчислюється відстань між продукувати класами розбиття; на етапі пошуку визначається нижня оцінка відстані від об'єкта запиту до всіх класів;

- LC - метод «кластеризації» елементів метричного простору, в якому процедура розбиття є рекурсивної: з безлічі об'єктів вибирається довільне зображення, центр «кластера», до кластеру відносять всі об'єкти, які розташовані від «кластера" не далі ніж заданий фіксований радіус, для решти «віддалених» зображень процедура повторюється; на етапі пошуку використовується нижня і верхня оцінка відстані від об'єкта запиту до поточного кластера;

- M-дерево і його вдосконалення Slim-дерево - методи з динамічними структурами даних, адаптивними до операцій вставки і видалення.

Ефективність методів метричної індексації оцінюється за такими параметрами, як кількість звернень до пам'яті, обчислювальні витрати пошуку по індексного структурі і кількість операцій визначення відстані між поданням запиту і об'єктами бази даних. Загальна тенденція для методів пошуку зображень в базах даних полягає в тому, що прагнення до усунення семантичного конфлікту призводить до зростання обчислювальний складності методів обробки зображень і операцій обчислення відстані. Так, наприклад, операція порівняння зображень на рівні областей за допомогою EMD-метрики, що потребує вирішення транспортної задачі, на кілька порядків більш трудомістка, ніж операція обчисленням евклидова відстані між двома глобальними ознаковими описами. Отже, при створенні ефективної індексної структури для баз даних, що містять зображення, слід в першу чергу акцентувати увагу на методах, що забезпечують найменшу кількість операцій обчислення відстані.

Резюмуючи викладене, слід зазначити, що, на відміну від традиційних моделей розпізнавання, де прагнуть визначити групу позитивних результатів, в системах пошуку зображень потрібно виключити з розгляду якомога більше об'єктів для забезпечення можливості інтерактивного перегляду і уточнення результату. Методи метричної індексації в цьому аспекті є більш перспективним підходом до питання скорочення числа операцій обчислення відстані на етапі виконання запиту в порівнянні з методами просторової індексації.

**1.4 Теоретичні основи метричного пошуку зображень**

Завдання полягає в пошуку по заданому запиту , Який видається (або може приводитися) в одному із зазначених вище видів, елемента (ів) , Найкращим чином йому відповідного (их). Поняття «найкращим чином» будемо трактувати як мінімум відстані , . Нагадаємо, що неотрицательная функція є метрикою на безлічі , якщо виконуються аксіоми:

*а* ) - рефлексивність;

*б* ) - симетричність;

*в* ) - нерівність трикутника.

З огляду на варіативність завдання зображень, необхідно підкреслити, що якщо заданий набір метрик , То їх неотрицательная лінійна комбінація також є метрикою. Більш того, якщо деяка неотрицательная функція така, що і є опуклою , то - метрика.

Використання в якості критерію подібності метрики забезпечує, по-yoпервих, адекватність результату пошуку запитом (за умови правильного вибору метрики, що відповідає вимогам предметно-орієнтованої області), по-друге, облік нерівності трикутника створює передумови для виключення з розгляду цілих множин зображень без обчислення відстаней до них. Відзначимо, що пошук зі скороченим числом переборовши може здійснюватися двома шляхами: або використовуючи попередню кластеризацію в просторі зображень або ознак, або на основі методів, що аналізує значення заздалегідь обчисленої матриці відстаней між усіма елементами колекції відеокадрів. З іншого боку, всі алгоритми пошуку можна розділити на три групи: пошук найбільш схожих зображень, впорядкованих за подібністю, пошук зображень, що відрізняються від запиту не більше ніж на задане граничне обмеження , І комбінація цих підходів.

**1.5 Постановка завдання**

В даний час спостерігається багаторазове збільшення обсягів реєстрованого візуальної інформації. Як результат, все більш актуальною стає проблема ефективного зберігання, обробки і аналізу вмісту колекцій зображень, а також засобів пошуку та навігації. Найбільшою популярністю користується підхід, в якому запит задається зображенням-зразком, а схожість зображень визначається на рівні близькості видобутих прізнакових описів. Аналіз існуючих систем пошуку, трендів проведених досліджень показав, що намітився перехід від глобальних до локальних признаковая описами зображень. Синтез моделей порівняння зображень на рівні областей, які враховують просторовий схожість безвідносно рівня деталізації розбиття поля зору, є актуальним завданням, рішення якої дозволить, в тому числі, здійснювати пошук областей інтересу, корелюють з шуканими об'єктами.

Проведений аналіз існуючих методів індексації показав, що вибір методів метричної індексації, заснованих на оцінці відстаней за допомогою нерівності трикутника, найбільш раціональний для скорочення числа операцій порівняння при метричному пошуку зображень на рівні областей.

З вищевикладеного можна зробити висновок, що модель пошуку зображень, заснована на метричному аналізі та індексації результатів сегментації зображень із запитом за зразком, є досить перспективною для вирішення цілої низки практичних завдань.

**РОЗДІЛ 2 ІНСТРУМЕНТИ РОЗРОБКИ**

**2.1 JavaScript**

JavaScript - це порівняно нова мова для написання сценаріїв, розроблений компанією Netscape. За допомогою мови JavaScript ми можемо створювати інтерактивні web-сторінки найбільш зручним і ефективним способом. У цьому практичному посібнику наведені приклади, які, на думку автора, є найбільш важливими. Вони зможуть продемонструвати можливості JavaSript і принципи організації мови. У цьому посібнику наводяться приклади того, що можна зробити, використовуючи JavaScript, а також, що не менш важливо, розповідається, як це зробити.

JavaScript - це не те ж саме, що Java. Багато людей вважають, що мова JavaScript - це те ж саме, що мова Java, недарма вони носять однакові імена. Однак це невірно. Не будемо розбиратися в існуючих відмінностях, важливо лише пам'ятати, що JavaScript та Java - це різні мови, хоча в них є багато спільного.

Найчастіше мова JScript використовується для обробки подій на веб-сторінці таких як натиснення на кнопку відправки форми, вибір нового елементу зі списку, позиціонування курсору мишки над якимось елементом, завершення завантаження сторінки тощо. Призначення такого обробника подій - попередити користувача про можливу помилку в його діях чи створити більш зручну обстановку для роботи зі сторінкою. Проте іноді застосування мови скриптів може бути й більш прозаїчним : в момент завантаження виводити на сторінку HTML-код , що сформує той чи інший елемент сторінки після її завантаження чи провести певні обрахунки.

jQuery - популярна JavaScript-бібліотека з відкритим сирцевим кодом. Вона була представлена у січні 2006 року у BarCamp NYC Джоном Ресіґом (John Resig). Згідно з дослідженнями організації W3Techs, JQuery використовується понад половиною від мільйона найвідвідуваніших сайтів. jQuery є найпопулярнішою бібліотекою JavaScript, яка посилено використовується на сьогоднішній день.

Синтаксис jQuery розроблений, щоб зробити орієнтування у навігації зручнішим завдяки вибору елементів DOM, створенню анімації, обробки подій, і розробки AJAX-застосунків. jQuery також надає можливості для розробників, для створення плагінів у верхній частині бібліотеки JavaScript. Використовуючи ці об'єкти, розробники можуть створювати абстракції для низькорівневої взаємодії та створювати анімацію для ефектів високого рівня. Це сприяє створенню потужних і динамічних веб-сторінок.

**2.2 HTML**

HTML (від англ. Hypertext Markup Language - мова розмітки гіпертексту) - це стандартна мова розмітки документів у Всесвітній павутині. Всі веб-сторінки створюються за допомогою мови HTML (або XHTML). Мова HTML інтерпретується браузером і відображається у вигляді документа, зручному для людини. HTML є додатком SGML (стандартної узагальненої мови розмітки) і відповідає міжнародному стандарту ISO 8879.

HTML-документ є текстовим файлом розмічений за допомогою спеціальних (природно, текстових) команд. Текстовий формат представлення веб-документів був вибраний виходячи з основних вимог до веб-документу: простота, можливість безпосередньої інтерпретації в будь-якій операційній системі, мінімальний розмір файлу, зручність редагування і інтерпретації.

Мова розмітки гіпертекстових документів HTML дозволяє визначити різні типи елементів (у оригіналі element), що забезпечують функціональність документа: текстові фрагменти із заданими параметрами форматування, списки, таблиці, зображення, гіперпосилання і т.д. Елементи HTML оголошуються за допомогою команд розмітки, званих тегами (від англійського tag - ярлик). Усs HTML-теги, що зустрічаються в тексті документа інтерпретуються браузером при відображенні документа.

**2.3 CSS**

Cascading Style Sheets (каскадні таблиці стилів) - технологія опису зовнішнього вигляду документа, написаного мовою розмітки. CSS використовується переважно для оформлення HTML- і XHTML-документів, але іноді і для інших XML-структурованих документів (наприклад, в браузері Mozilla для оформлення елементів графічного інтерфейсу, XUL).

CSS використовується розробниками веб-сторінок для завдання кольорів, шрифтів, розташування і інших аспектів представлення документа. Основною метою розробки CSS було розділення вмісту (написаного на HTML або іншій мові розмітки) від представлення стилю документа. Це розділення може збільшити доступність документа, надати велику гнучкість і можливість управління його виглядом, а також зменшити складність і повторюваність в структурному вмісті. Крім того, CSS дозволяє представляти один і той же документ в різних стилях.

Стандарт CSS визначає пріоритети, у порядку яких застосовуються правила стилів, якщо для якогось елементу підходять деякі правила одночасно. Це називається "каскадом", в якому для правил розраховуються пріоритети або "ваги", що робить результати передбаченими.

Таблиця стилів складається з набору правил. Кожне правило, у свою чергу, складається з одного або декількох селекторів, розділених комами і блоку визначень.

**2.4 PHP**

PHP (англ. PHP:Hypertext Preprocessor — PHP: гіпертекстовий препроцесор), попередньо: Personal Home Page Tools — скриптова мова програмування, була створена для генерації HTML-сторінок на стороні веб-серверу. PHP є однією з найпоширеніших мов, що використовуються у сфері веб-розробок (разом із Java, .NET, Perl, Python, Ruby). PHP підтримується переважною більшістю хостинг-провайдерів. Проект за яким був створений PHP — проект з відкритими програмними кодами.

Історія PHP починається з [1995](https://uk.wikipedia.org/wiki/1995) року, коли Расмус Лердорф ([англ.](https://uk.wikipedia.org/wiki/Англійська_мова) Rasmus Lerdorf) створив простий [застосунок](https://uk.wikipedia.org/wiki/Застосунок) мовою Perl, що аналізував відвідування користувачами його резюме на веб-сайті. Потім, коли цим [застосунком](https://uk.wikipedia.org/wiki/Застосунок) вже користувалися кілька чоловік, а число охочих одержати його постійно збільшувалося, Лердорф назвав своє творіння Особисті інструменти домашньої сторінки [англ.](https://uk.wikipedia.org/wiki/Англійська_мова) Personal Home Page Tools версія 1 і виставив для вільного завантаження. З цієї миті почався небувалий зліт популярності PHP.

Як це завжди буває, терміново було потрібне доопрацювання і нові доповнення. Для їхньої реалізації Расмус створює нову версію пакету, тепер уже написану на [С](https://uk.wikipedia.org/wiki/С). Отриманий таким чином інструмент набуває робочої назви [PHP/FI](https://uk.wikipedia.org/w/index.php?title=PHP/FI&action=edit&redlink=1) [Персональна Домашня сторінка / Інтерпретатор Форм](https://uk.wikipedia.org/w/index.php?title=Персональна_Домашня_сторінка_/_Інтерпретатор_Форм&action=edit&redlink=1) надалі він також буде відомий під назвою [PHP 2](https://uk.wikipedia.org/w/index.php?title=PHP_2&action=edit&redlink=1). Ця версія вже більшою мірою схожа на сьогоднішній PHP. Вона мала синтаксис і спосіб іменування змінних в стилі мови Perl, можливість вбудовування PHP операторів в [html](https://uk.wikipedia.org/wiki/Html)-код сторінки, автоматичну інтерпретацію форм, [інтеграцію](https://uk.wikipedia.org/wiki/Інтеграція) з базами даних

Закінчивши роботу Зеєв і Енді домовились з Расмусом про співпрацю в галузі розвитку та вдосконалення мови. З цієї миті з'являється PHP Group — група однодумців, що працюють над розвитком технології PHP. Одержаний продукт з'явився на світ у 1998 році під назвою PHP 3.

За час з [2000](https://uk.wikipedia.org/wiki/2000) по 2004 рік продовжувалися активні роботи з покращення 4 версії, але майже відразу [PHP Group](https://uk.wikipedia.org/w/index.php?title=PHP_Group&action=edit&redlink=1) приступила до продумування можливостей нової версії. В першу чергу було вирішено підсилити об'єктні можливості мови, що дозволяло використовувати його для реалізації масштабних проектів. Роботи із створення версії 5 велися тривалий час, в них брало участь рекордна кількість фахівців, зокрема Стерлінг Хьюз ([англ.](https://uk.wikipedia.org/wiki/Англійська_мова) Sterling Hughes) і Маркус Бергера ([англ.](https://uk.wikipedia.org/wiki/Англійська_мова) Marcus Boerger).

У липні [2004](https://uk.wikipedia.org/wiki/2004) року виходить офіційний [реліз](https://uk.wikipedia.org/wiki/Реліз_(програмне_забезпечення)) [PHP 5](https://uk.wikipedia.org/wiki/PHP_5). В першу чергу, як і планувалося, було перероблено весь механізм роботи з об'єктами. І якщо в попередніх версіях об'єктно-орієнтоване програмування на PHP було можливе в мінімальному ступені, а тому і використовувалося на практиці не часто, то PHP 5 володіє прекрасним потенціалом реалізації об'єктного програмування. Окрім цього, PHP збагатився рядом цінних розширень для роботи з [XML](https://uk.wikipedia.org/wiki/XML), різними джерелами даних, генерації [графіки](https://uk.wikipedia.org/wiki/Графіка) і інше.

З точки зору [системи типізації](https://uk.wikipedia.org/wiki/Система_типізації), PHP є мовою програмування з [динамічною типізацією](https://uk.wikipedia.org/wiki/Динамічна_типізація). Немає необхідності явного визначення типу змінних, хоча така можливість існує. В разі звернення до змінної, [інтерпретатор](https://uk.wikipedia.org/wiki/Інтерпретатор) PHP трактує її тип відповідно до контексту. За необхідності можливе приведення змінної до певного типу за допомогою відповідних конструкцій мови. Це може знадобитись, якщо зважити, що значення змінної можуть трактуватись по-різному в залежності від її типу. Також можливе визначення типу відповідної змінної на певному етапі виконання сценарію. Імена змінних чутливі до регістру символів.

**2.5 Symfony**

Symfony — відкритий [PHP](https://uk.wikipedia.org/wiki/PHP)-[фреймворк](https://uk.wikipedia.org/wiki/Фреймворк), що реалізує концепцію [модель-вид-контролер](https://uk.wikipedia.org/wiki/Модель-вид-контролер) ([MVC](https://uk.wikipedia.org/wiki/MVC)) та автоматизовує найзагальніші веб-задачі, являє собою широконалаштовну систему пов'язаних [класів](https://uk.wikipedia.org/wiki/Клас_(програмування)) і призначений для розробки та керування [веб-застосунками](https://uk.wikipedia.org/wiki/Веб-застосунок). Випускається під [МІТ](https://uk.wikipedia.org/wiki/Ліцензія_MIT) ліцензією. Symfony є [вільним програмним забезпеченням](https://uk.wikipedia.org/wiki/Вільне_програмне_забезпечення). [Веб-сайт](https://uk.wikipedia.org/wiki/Веб-сайт) першоі версії symfony-project.com був запущений [18](https://uk.wikipedia.org/wiki/18) [жовтня](https://uk.wikipedia.org/wiki/Жовтень) [2005](https://uk.wikipedia.org/wiki/2005) року. Symfony не варто плутати із [Symphony CMS](https://uk.wikipedia.org/w/index.php?title=Symphony_CMS&action=edit&redlink=1) відкритої [XML](https://uk.wikipedia.org/wiki/XML)/[XSLT](https://uk.wikipedia.org/wiki/XSL_Transformations) [CMS](https://uk.wikipedia.org/wiki/Система_керування_вмістом)

Symfony спрямований на прискорення створення та підтримки веб-застосунків, а також для уникнення витрат часу для розв'язування тривіальних задач у розробці (наприклад, написання валідаторів форм).

Для встановлення Symfony необхідний [Unix](https://uk.wikipedia.org/wiki/Unix), [Linux](https://uk.wikipedia.org/wiki/Linux), [Mac OS](https://uk.wikipedia.org/wiki/Mac_OS) чи [Windows](https://uk.wikipedia.org/wiki/Windows) із [веб-сервером](https://uk.wikipedia.org/wiki/Веб-сервер) та встановленим [PHP 5](https://uk.wikipedia.org/wiki/PHP). Symfony сумісний із такими [об'єктно-реляційними відображеннями](https://uk.wikipedia.org/wiki/Об'єктно-реляційне_відображення), як [Doctrine](https://uk.wikipedia.org/wiki/Doctrine_(PHP)) та [Propel](https://uk.wikipedia.org/wiki/Propel).

За допомогою [акселератора PHP](https://uk.wikipedia.org/wiki/Акселератор_PHP) Symfony збільшує продуктивність та зменшує навантаження на сервер.

Symfony ставить за мету дати розробникам повний контроль над конфігурацією: майже все можливо налаштувати, від структури каталогів до сторонніх [бібліотек](https://uk.wikipedia.org/wiki/Програмна_бібліотека).

Symfony використовує шаблон проектування [модель-вид-контролер](https://uk.wikipedia.org/wiki/Модель-вид-контролер). Розробників Symfony надихнули такі фреймворки, як [Ruby on Rails](https://uk.wikipedia.org/wiki/Ruby_on_Rails), [Django](https://uk.wikipedia.org/wiki/Django) та [Spring Framework](https://uk.wikipedia.org/wiki/Spring_Framework).

Використовуючи систему [плагінів](https://uk.wikipedia.org/wiki/Плагін), Symfony може підтримувати [JavaScript](https://uk.wikipedia.org/wiki/JavaScript) фреймворків та багато інших [PHP](https://uk.wikipedia.org/wiki/PHP) проектів:

Ще однією перевагою Symfony є наявність генераторів, за допомогою яких значно пришвидшується розробка.

Серед можливостей: інструменти для локалізації та інтернаціоналізації, unit-тестування, БД-абстракції, smart-URL, Debug Toolbar, development та production режими, form framework.

**РОЗДІЛ 3. ПРАКТИЧНА ЧАСТИНА**

**3.1 Структура веб-ресурсу**

Після того, як користувач заходить в середовище веб-ресурсу перед ним зявляється головне вікно програми (Рис. 3.1)

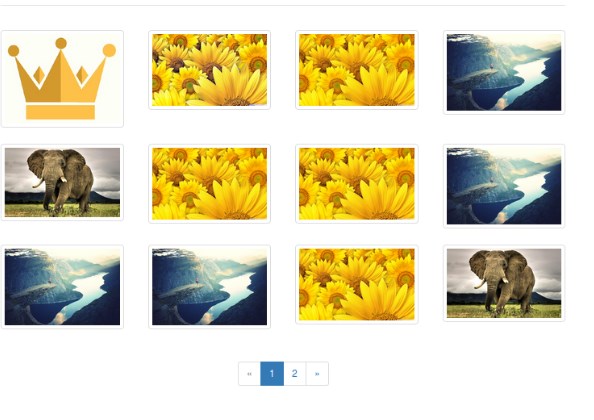


Рис. 3.1 – головне вікно програми

Як видно на зображення користувач може перейти на наступну сторінку або повернутися на попередню за допомогою нумерації сторінок внизу програми, а також за допомогою відповідних стрілок-вказівників.

Також користувач має можливість завантажити нове зображення. Для цього необхідно вибрати пункт меню «Upload», який знаходиться вверху екрану на головній сторінці (Рис. 3.2)

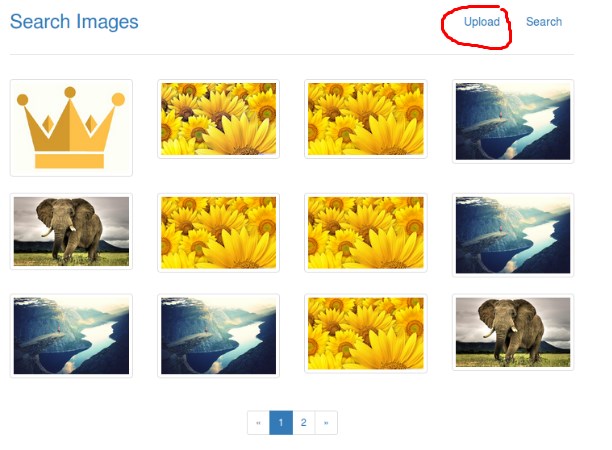


Рис. 3.2 - завантаження файлу

Після того як користувач вибере даний пункт меню перед ним з’явиться сторінка завантаження нового файлу (Рис. 3.3)

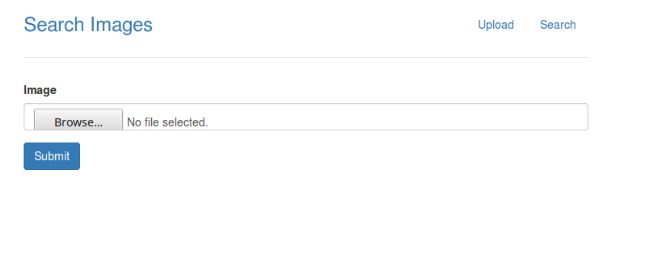


Рис. 3.3 – сторінка завантаження файлу

Після того користувач може обрати необхідний йому файл з свого комп’ютера. Для цього перед користувачем з’являється вікно завантаження файлу (Рис. 3.4)

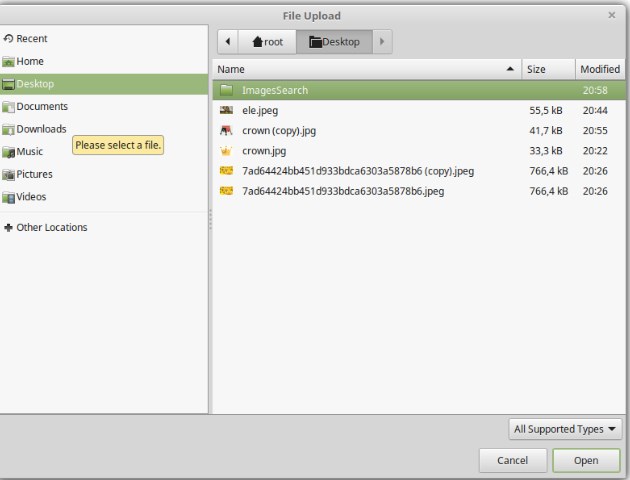


Рис. 3.4 – вікно завантаження файлу

Після цього, якщо файл зображення успішно завантажено, то з’являється сторінка успішного завантаження картинки (Рис. 3.5)

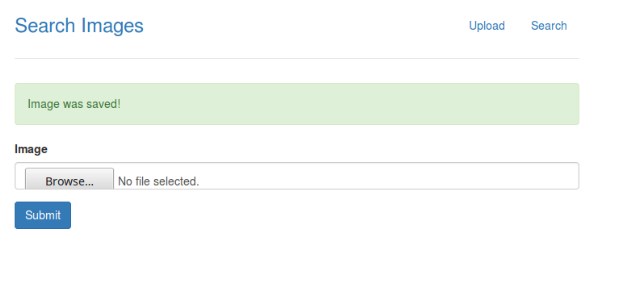


Рис. 3.5 – сторінка успішного завантаження

Крім цього, користувач для здійснення пошуку може завантажувати картинки користуючись таким самим принципом як описано вище.

Завантаживши вибрану картинку для пошуку її в базі для користувача виникають два можливих варіанти. Перший – це пошук не дав результатів, такого зображення нема (Рис. 3.6)

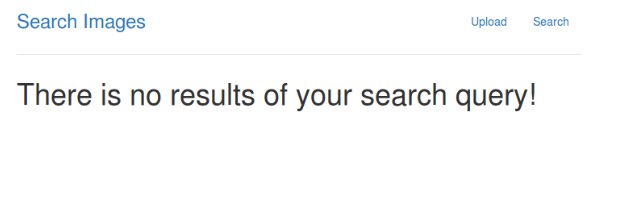


Рис. 3.6 – пошук не дав результатів

Другим варіантом є вдалий пошук, якщо така картинка дійсно знаходиться в базі (Рис. 3.7)

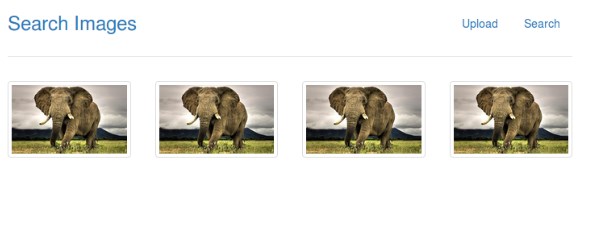


Рис. 3.7 – результати пошуку

Після цього користувач може вибрати знайдену картинку і перед ним з’явиться вікно перегляду (Рис. 3.8)



Рис. 3.8 – вікно перегляду зображення

**3.2 Етапи розробки проекту**

Послідовна розробка проекту дозволяє чітко відокремити кожен, з етапів реалізації тієї чи іншої функції. Така розмежованість надає змогу без затрат часу на вивчення коду вносити зміни тільки в ту частину проекту, котру дійсно потрібно, а не змінювати весь веб-додаток.

Першим етапом розробки було створення головної сторінки, де показані всі зображення, які є в базі розробленого ресурсу. Тут користувач отримує можливість переходити зі сторінки на сторінку та повертатись на попередню.

Після закінчення реалізації оформлення головної сторінки та функції переходу до наступних елементів ресурсу, розпочалась розробка рівнів. Першочергово було розроблено функції пошуку та завантаження зображень для пошуку. Далі розроблено можливість вибирати одну із запропонованих картинок для окремого детального перегляду. Оскільки основна ідея роботи ресурсу не потребує якихось додаткових елементів на зразок коментарів, то ці деталі було упущено для того, щоб зберегти лаконічний інтерфейс

Наприкінці розпочався етап візуального оформлення, до складу якого входить вступна частина та візуальні блоки.

Реалізація цих етапів цілком і повністю покладена на каскадну модель програмування, котра дозволяє покроково виконувати весь функціонал проекту та розпочинати наступний етап тільки після того, як попередній був закінчений. Такий підхід до організації роботи дозволяє наперед визначити інструментарій та ресурсі, які будуть використовуватись при розробці веб – ресурсу.

**3.3. Огляд аналогів**

Зараз існує чимало програм для пошуку зображень. Для їх функціонування достатньо всього-лиш мати доступ до Інтернету. Найбільш відомою та популярною є методика пошуку зображення за допомогую функції пошуковика Google, яка і має відповідну назву «Google зображення ». Працює воно дуже просто. Достатньо в пошуковику ввести «Google зображення» і перед користувачем виникає наступне вікно (Рис. 3.9)

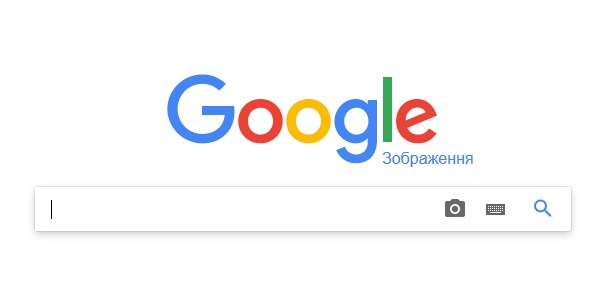


Рис. 3.9 – сторінка пошуку зображення

Потім необхідно вибрати пошук і один з методів завантаження зображення: посилання на нього або безпосередньо з комп’ютера (Рис. 3.10)

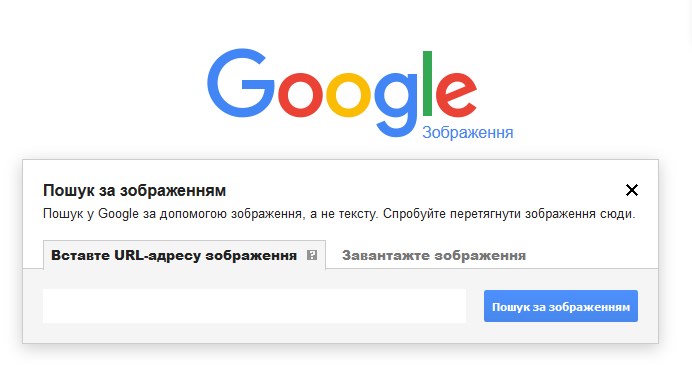


Рис. 3.10 – завантаження зображення

**Висновки**

Актуальність даного завдання обумовлена зростанням розмірів колекцій мультимедіа даних в різних сферах людської діяльності, що призводить до численних питань забезпечення ефективного зберігання, стиснення, інтерпретації і, в тому числі, забезпечення швидкого пошуку. Метою є дослідження алгоритмів пошуку зображень за зразком в базах даних і синтез методів індексації в метричному просторі для забезпечення шви-якого пошуку зображень. Об'єктом дослідження є система управління базами даних зображень. Предметом дослідження є методи метричної індексації вмісту баз відеоданих на базі метричного порівняння результатів сегментації зображень. Проблема пошуку набору елементів, які близькі до даного, запитуваного, елементу за деякими критеріями подібності є на сьогоднішній день надзвичайно актуальною. Для зберігання інформації в усіх сучасних аналізаторах зображень передбачено використання бази даних. База даних візуальної інформації перед-ставлять собою програмне забезпечення, призначене для накопичення, зберігання, архівування інформації, її систематизації і подальшого швидкого пошуку зображень подібних заданому. Це дозволяє вирішувати різні прикладні завдання, пов'язані з обробкою і розпізнаванням зображень.

В умовах постійно зростаючих обсягів реєстрованих відеоданих необхдно синтез нових, які відповідають висунутим вимогам по валідності і швидкодії методів і моделей пошуку зображень в базах даних. Аналіз поточного стану досліджень в області пошуку і розроблених програмних комплексів показав, що перспективним і вже зараз реалізовуються на практиці підходом є реалізація запитів за зразком з порівнянням зображень у вигляді результатів їх сегментації.

В рамках даної роботи були досліджені методи і моделі пошуку зображень в базах даних за зразком. Для порівняння зображень було вибрано метричний спосіб порівняння результатів сегментації зображень. На цій основі синтезовані і досліджені методи індексації, названі метричними, які дозволяють скорочувати кількість операцій порівняння на етапі пошуку, а саме: метод на повній матриці відстаней, на «розрідженій» матриці відстаней, метод, який створює дерева пошуку, метод, заснований на розбитті ( кластеризації) даних.

Для дослідження цільової моделі пошуку зображень і синтезованих методів метричної індексації був створений дослідний програмний комплекс. Результати імітаційного моделювання, результати експериментів по скороченню кількості операцій порівняння на етапі виконання запиту в системі пошуку зображень за допомогою методів метричної індексації, порівняльний аналіз з існуючими системами-аналогами підтвердили ефективність запропонованого підходу.

**Список використаних джерел**

1. Chan K., Liu F., Purnomo R., Xiong X. Content-based image retrieval using regional representation // International Workshop on Theoretical Foundations of Computer Vision: Multi-Image Analysis / Klette R. et al. (Eds.). – Lecture Notes In Computer Science. – London, UK: Springer-Verlag. – Vol. 2032. – 2000. – P. 238250.

2. Bozkaya T., Ozsoyoglu M. Indexing large metric spaces for similarity search queries // ACM Trans. on Database Systems (TODS). – Vol. 24, No. 3. – 1999. – P. 361404.

3. Гонсалес Р., Вудс Р. Цифровая обработка изображения: Пер. с англ. – М.: Техносфера. – 2005. – 1070 с.

4. Chang N.S., and Fu K.S. Query by pictorial example // IEEE Trans. on Software Engineering.  Vol. 6, No. 6. – 1980. – P. 519524.

5. Smeulders A.W., Worring M., Santini S., Gupta A., Jain R. Content-based image retrieval at the end of the early years // IEEE Trans. on Pattern Analysis and Machine Intelligence.  Vol. 22, No. 12.  2000. – P. 1349–1380.

6. Форсайт Д., Понс Ж. Компьютерное зрение: современный подход: Пер. с англ..  М.: Издательский дом «Вильямс». – 2004. – 928 с.

7. Шапиро Л., Стокман Дж. Компьютерное зрение: Пер. с англ. – М.: Бином. – 2006. – 752 с.

8. Rui Y., Huang T., Chang S.-F. Image retrieval: Current techniques, promising directions and open issues // Journal of Visual Communication and Image Representation. – Vol. 10, No. 4.  1999. – P. 39–62.

9. Veltkamp R.C., Hagendoorn M. State-of-the-art in shape matching // Principles of visual information retrieval. – 2000. – P. 87119.

10. Zhang D., Lu G. Review of shape representation and description techniques // Pattern Recognition. – Vol. 37. – 2003.  P. 119.

11. Mikolajczyk K., Schmid C. A performance evaluation of local descriptors // IEEE Trans. on Pattern Analysis and Machine Intelligence. – Vol. 27, No. 10. – 2005.  P. 16151630.

12. Liu Y., Zhanga D., Lua G., Ma W.-Y. A survey of content-based image retrieval with high-level semantics // Pattern Recognition. – Vol. 40, No. 1. – 2007. – P. 262–282.

13. Дуда Р., Харт П. Распознавание образов и анализ сцен: Пер. с англ.  М.: Мир. – 1976. – 512 с.

14. Прэтт У. Цифровая обработка изображений.  М.: Мир. – 1982. – 792 с

15. Путятин Е.П., Аверин С.И. Обработка изображений в робототехнике.  М.: Машиностроение. – 1990. – 320 с.

16. Ту Дж., Гонсалес Р. Принципы распознавания образов: Пер. с англ. – М.: Мир. – 1978. – 416 с.

17. Deng Y., Manjunath B. Unsupervised segmentation of color-texture regions in images and video // IEEE Trans. on Pattern Analisys and Machine Intelligence. – Vol. 23, No 8. – 2001. – P. 800810.

18. Carson C., Belongie S., Greenspan H., Malik J. Blobworld: image segmentation using expectation-maximization and its application to image querying // IEEE Trans. on Pattern Analysis and Machine Intelligence. Vol. 24, No. 8. – 2002. – P. 1026–1038.

19. Shi J., Malik J. Normalized cuts and image segmentation // IEEE Trans. on Pattern Analysis and Machine Intelligence. – Vol. 22, No 8. – 2000. – P. 888905.

20. Rubner Y., Tomasi C., Guibas L. The Earth Mover’s Distance as a metric for image retrieval // International Journal of Computer Vision / Springer Netherlands. – Vol. 40, No 2. – 2000. – P. 99121.

21. Long F., Zhang H., Feng D. Fundamentals of content-based image retrieval // Multimedia Information Retrieval and Management  Techonological Fundamentals and Applications / D. Feng, W. Sie. (Eds.).  Springer.  2003. – P. 132.

22. Ko B., Byun H. Integrated region-based image retrieval using region’s spatial relationships // Proc. of the 16th International Conference on Pattern Recognition. – Vol. 1. – 2002. – P. 196199.

23. Jiang X., Marti C., Irniger C., Bunke H. Image segmentation evaluation by techniques of comparing clusterings // Image Analysis and Processing – ICIAP 2005 / Fabio R., Sergio V. (Eds.). – Lecture Notes in Computer Science. – Vol. 3617. – 2005. – P. 344351.

24. Meila M. Comparing clusterings by the variation of information // Proc. of the 6th Annual Conference on Learning Theory. – 2003. – P. 173–187.

25. Kinoshenko D., Mashtalir V., Shlyakhov V. A partition metric for clustering features analysis // International Journal Information Theories and Applications. –Vol. 14, No. 3. – 2007. – Р. 230236.

26. Bohm C., Berchtold S., Keim D.A. Searching in high-dimensional spaces: Index structures for improving the performance of multimedia databases // ACM Computing Surveys (CSUR). – Vol. 33, No. 3. – 2001. – P. 322373.

27. Vidal E. An algorithm for finding nearest neighbours in (approximately) constant average time // Pattern Recognition Letters. – Vol. 4, No. 3. – 1986. – P. 145157.

28. Mico M.L., Oncina J., Vidal E. A new version of the nearest-neighbour approximating and eliminating search algorithm (AESA) with linear preprocessing time and memory requirements // Pattern Recognition Letters. – Vol. 15, No. 1. – 1994. – P. 917.

29. Uhlmann J. Satisfying general proximity/similarity queries with metric trees // Information Processing Letters. – Vol. 40. – 1991. – P. 175179.

30. Brin S. Near neighbor search in large metric spaces // Proc. of the 21st International Conference on Very Large Data Bases / Dayal U., Gray P., Nishio S. (Eds.).  Very Large Data Bases. – 1995.  P. 574584.

**Додаток 1 Лістинг програми**

|  |  |
| --- | --- |
| <?php |  |
|  |  |
|  | namespace AppBundle\Controller; |
|  |  |
|  | use AppBundle\Entity\Image; |
|  | use Symfony\Bundle\FrameworkBundle\Controller\Controller; |
|  | use Sensio\Bundle\FrameworkExtraBundle\Configuration\Route; |
|  | use Symfony\Component\Config\Definition\Exception\Exception; |
|  | use Symfony\Component\HttpFoundation\Request; |
|  | use Symfony\Component\HttpFoundation\Response; |
|  | use Symfony\Component\HttpFoundation\File\UploadedFile; |
|  |  |
|  | class ImageController extends Controller |
|  | { |
|  | /\*\* |
|  | \* @Route("/upload") |
|  | \*/ |
|  | public function uploadAction(Request $request) |
|  | { |
|  | if ($request->isMethod('post')) { |
|  | try { |
|  | /\*\* @var Symfony\Component\HttpFoundation\File\UploadedFile $imageFile \*/ |
|  | $imageFile = $request->files->get('image'); |
|  | $fileName = md5(uniqid()) . '.' . $imageFile->guessExtension(); |
|  | $imageFile->move($this->getParameter('images\_dir'), $fileName); |
|  | $image = new Image(); |
|  | $image->setPath($this->getParameter('public\_image\_path') . '/' . $fileName); |
|  | $manger = $this->getDoctrine()->getManager(); |
|  | $manger->persist($image); |
|  | $manger->flush(); |
|  | $this->addFlash('success', 'Image was saved!'); |
|  | } catch (Exception $e) { |
|  | $this->addFlash('danger', $e->getMessage()); |
|  | } |
|  | } |
|  | return $this->render('AppBundle:Image:create.html.twig'); |
|  | } |
|  |  |
|  | /\*\* |
|  | \* @Route("/images/{page}/{limit}") |
|  | \*/ |
|  | public function listAction($page = 1, $limit = 12) |
|  | { |
|  | $images = $this->getDoctrine()->getManager()->getRepository('AppBundle:Image')->getImages($page, $limit); |
|  | $maxPages = ceil($images->count() / $limit); |
|  | $thisPage = $page; |
|  | $images = $images->getIterator(); |
|  | return $this->render('AppBundle:Image:list.html.twig', compact('images', 'maxPages', 'thisPage', 'limit')); |
|  | } |
|  |  |
|  | /\*\* |
|  | \* @Route("/search/{page}/{limit}") |
|  | \*/ |
|  | public function searchAction(Request $request, $page = 1, $limit = 12) |
|  | { |
|  | if ($request->isMethod('post')) { |
|  | $imagePath = $this->\_uploadSearched($request); |
|  | $this->get('session')->set('search\_path', $imagePath); |
|  | } |
|  | $searchPath = $this->get('session')->get('search\_path'); |
|  | if ($searchPath) { |
|  | $images = $this->getDoctrine()->getManager()->getRepository('AppBundle:Image')->searchByImage($page, $limit, $searchPath); |
|  | if(!$images){ |
|  | return $this->redirectToRoute('app\_image\_noresults'); |
|  | } |
|  | $maxPages = ceil($images->count() / $limit); |
|  | $thisPage = $page; |
|  | $images = $images->getIterator(); |
|  | return $this->render('AppBundle:Image:list.html.twig', compact('images', 'maxPages', 'thisPage', 'limit')); |
|  | } |
|  |  |
|  | return $this->redirectToRoute('app\_image\_imagessearch'); |
|  | } |
|  |  |
|  | /\*\* |
|  | \* @Route("/") |
|  | \*/ |
|  | public function indexAction() |
|  | { |
|  | return $this->redirectToRoute('app\_image\_list'); |
|  | } |
|  |  |
|  | /\*\* |
|  | \* @Route("/images-search") |
|  | \*/ |
|  | public function imagesSearchAction() |
|  | { |
|  | return $this->render('@App/Image/images-search.html.twig'); |
|  | } |
|  |  |
|  | /\*\* |
|  | \* @Route("/no-results") |
|  | \*/ |
|  | public function noResultsAction() |
|  | { |
|  | return $this->render('@App/Image/no-results.html.twig'); |
|  | } |
|  |  |
|  | /\*\*Upload Searched Image And Return Path |
|  | \* @param Request $request |
|  | \* @return string |
|  | \*/ |
|  | private function \_uploadSearched(Request $request) |
|  | { |
|  | $imageFile = $request->files->get('image'); |
|  | $fileName = md5(uniqid()) . '.' . $imageFile->guessExtension(); |
|  | $url = $request->getSchemeAndHttpHost() . '/' . $this->getParameter('searched\_images\_url') . $fileName; |
|  | $imageFile->move($this->getParameter('searched\_images\_dir'), $fileName); |
|  | return $url; |
|  | } |
|  |  |
|  | } |

|  |  |
| --- | --- |
| <?php |  |
|  |  |
|  | namespace AppBundle\Entity; |
|  |  |
|  | use Doctrine\ORM\Mapping as ORM; |
|  |  |
|  | /\*\* |
|  | \* Image |
|  | \* |
|  | \* @ORM\Table(name="image") |
|  | \* @ORM\Entity(repositoryClass="AppBundle\Repository\ImageRepository") |
|  | \*/ |
|  | class Image |
|  | { |
|  | /\*\* |
|  | \* @var int |
|  | \* |
|  | \* @ORM\Column(name="id", type="integer") |
|  | \* @ORM\Id |
|  | \* @ORM\GeneratedValue(strategy="AUTO") |
|  | \*/ |
|  | private $id; |
|  |  |
|  | /\*\* |
|  | \* @var string |
|  | \* |
|  | \* @ORM\Column(name="path", type="string", length=255) |
|  | \*/ |
|  | private $path; |
|  |  |
|  | /\*\* |
|  | \* Get id |
|  | \* |
|  | \* @return int |
|  | \*/ |
|  | public function getId() |
|  | { |
|  | return $this->id; |
|  | } |
|  |  |
|  | /\*\* |
|  | \* Set path |
|  | \* |
|  | \* @param string $path |
|  | \* |
|  | \* @return Image |
|  | \*/ |
|  | public function setPath($path) |
|  | { |
|  | $this->path = $path; |
|  |  |
|  | return $this; |
|  | } |
|  |  |
|  | /\*\* |
|  | \* Get path |
|  | \* |
|  | \* @return string |
|  | \*/ |
|  | public function getPath() |
|  | { |
|  | return $this->path; |
|  | } |
|  | } |

|  |  |
| --- | --- |
| <?php |  |
|  |  |
|  | namespace AppBundle\Repository; |
|  |  |
|  | use Doctrine\ORM\Tools\Pagination\Paginator; |
|  | use AppBundle\Utils\ImagesComparer as ImagesComparer; |
|  |  |
|  | /\*\* |
|  | \* ImageRepository |
|  | \* |
|  | \* This class was generated by the Doctrine ORM. Add your own custom |
|  | \* repository methods below. |
|  | \*/ |
|  | class ImageRepository extends \Doctrine\ORM\EntityRepository |
|  | { |
|  | /\*\* |
|  | \* Paginator Helper |
|  | \* |
|  | \* Pass through a query object, current page & limit |
|  | \* the offset is calculated from the page and limit |
|  | \* returns an `Paginator` instance, which you can call the following on: |
|  | \* |
|  | \* $paginator->getIterator()->count() # Total fetched (ie: `5` posts) |
|  | \* $paginator->count() # Count of ALL posts (ie: `20` posts) |
|  | \* $paginator->getIterator() # ArrayIterator |
|  | \* |
|  | \* @param Doctrine\ORM\Query $dql DQL Query Object |
|  | \* @param integer $page Current page (defaults to 1) |
|  | \* @param integer $limit The total number per page (defaults to 5) |
|  | \* |
|  | \* @return \Doctrine\ORM\Tools\Pagination\Paginator |
|  | \*/ |
|  | public function paginate($dql, $page = 1, $limit = 16) |
|  | { |
|  | $paginator = new Paginator($dql); |
|  |  |
|  | $paginator->getQuery() |
|  | ->setFirstResult($limit \* ($page - 1))// Offset |
|  | ->setMaxResults($limit); // Limit |
|  |  |
|  | return $paginator; |
|  | } |
|  |  |
|  | /\*\* |
|  | \*Get Images With Pagination |
|  | \*/ |
|  | public function getImages($currentPage = 1, $limit = 16) |
|  | { |
|  | // Create our query |
|  | $query = $this->createQueryBuilder('image') |
|  | ->orderBy('image.id', 'DESC') |
|  | ->getQuery(); |
|  |  |
|  | // No need to manually get get the result ($query->getResult()) |
|  |  |
|  | $paginator = $this->paginate($query, $currentPage, $limit); |
|  |  |
|  | return $paginator; |
|  | } |
|  |  |
|  | /\*\* |
|  | \* @param int $currentPage |
|  | \* @param int $limit |
|  | \* @param string $imagePath |
|  | \* @return Paginator|false |
|  | \*/ |
|  | public function searchByImage($currentPage = 1, $limit = 16, $imagePath) |
|  | { |
|  | $ids = $this->getImagesIds($imagePath); |
|  | if(empty($ids)){ |
|  | return false; |
|  | } |
|  | // Create our query |
|  | $queryBuilder = $this->createQueryBuilder('image'); |
|  | $query = $queryBuilder |
|  | ->where($queryBuilder->expr()->in('image.id', $ids)) |
|  | ->orderBy('image.id', 'DESC') |
|  | ->getQuery(); |
|  |  |
|  | // No need to manually get get the result ($query->getResult()) |
|  |  |
|  | $paginator = $this->paginate($query, $currentPage, $limit); |
|  |  |
|  | return $paginator; |
|  | } |
|  |  |
|  | /\*\*Get similar images ids |
|  | \* |
|  | \* @param string $imagePath |
|  | \* @return array |
|  | \*/ |
|  | protected function getImagesIds($imagePath) |
|  | { |
|  | $ids = array(); |
|  | $comparer = new ImagesComparer(); |
|  | $images = $this->findAll(); |
|  | foreach ($images as $image) { |
|  | $result = $comparer->compare($\_SERVER['DOCUMENT\_ROOT'] . $image->getPath(), $imagePath); |
|  | if ($result <= 20) { |
|  | $ids[] = $image->getId(); |
|  | } |
|  | } |
|  |  |
|  | return $ids; |
|  | } |
|  | } |

|  |  |
| --- | --- |
| {% extends "::layout.html.twig" %} |  |
|  |  |
|  | {% block title %}Upload Image{% endblock %} |
|  | {% block css %} |
|  | <link href="{{ asset('css/bootstrap.min.css') }}" rel="stylesheet"> |
|  | {% endblock %} |
|  | {% block body %} |
|  | {% for type, flashes in app.session.flashbag.all %} |
|  | {% for flash in flashes %} |
|  | <div class="alert alert-{{ type }} fade in"> |
|  | {{ flash }} |
|  | </div> |
|  | {% endfor %} |
|  | {% endfor %} |
|  | <form method="post" enctype="multipart/form-data"> |
|  | <div class="form-group"> |
|  | <label for="image">Image</label> |
|  | <input type="file" required name="image" id="image" class="form-control" accept="image/jpeg, image/png"> |
|  | </div> |
|  | <button type="submit" class="btn btn-primary">Submit</button> |
|  | </form> |
|  | {% block javascripts %}{% endblock %} |
|  | {% endblock %} |

|  |  |
| --- | --- |
| {% extends "::layout.html.twig" %} |  |
|  |  |
|  | {% block title %}Search Images{% endblock %} |
|  | {% block css %} |
|  | <link href="{{ asset('css/bootstrap.min.css') }}" rel="stylesheet"> |
|  | {% endblock %} |
|  | {% block body %} |
|  | {% for type, flashes in app.session.flashbag.all %} |
|  | {% for flash in flashes %} |
|  | <div class="alert alert-{{ type }} fade in"> |
|  | {{ flash }} |
|  | </div> |
|  | {% endfor %} |
|  | {% endfor %} |
|  | <form method="post" action="{{ path('app\_image\_search') }}" enctype="multipart/form-data"> |
|  | <div class="form-group"> |
|  | <label for="image">Image</label> |
|  | <input type="file" required name="image" id="image" class="form-control" accept="image/jpeg, image/png"> |
|  | </div> |
|  | <button type="submit" class="btn btn-primary">Submit</button> |
|  | </form> |
|  | {% block javascripts %}{% endblock %} |
|  | {% endblock %} |

|  |  |
| --- | --- |
| {% extends '::layout.html.twig' %} |  |
|  |  |
|  | {% block body %} |
|  | {% set i = 0 %} |
|  | {% for image in images %} |
|  | {% set i = i+1 %} |
|  | {% if i == 1 %} |
|  | <div class="row"> |
|  | {% endif %} |
|  | <div class="col-lg-3 col-md-4 col-xs-6 thumb"> |
|  | <a class="thumbnail" href="#"> |
|  | <img class="img-responsive" src="{{ asset(image.getPath()) }}" alt="image"> |
|  | </a> |
|  | </div> |
|  | {% if i == 4 or loop.last %} |
|  | {% set i = 0 %} |
|  | </div> |
|  | {% endif %} |
|  | {% endfor %} |
|  |  |
|  | <!-- Modal --> |
|  | <div class="modal fade" id="myModal" tabindex="-1" role="dialog" aria-labelledby="myModalLabel"> |
|  | <div class="modal-dialog" role="document"> |
|  | <div class="modal-content"> |
|  | <div class="modal-header"> |
|  | <button type="button" class="close" data-dismiss="modal" aria-label="Close"><span |
|  | aria-hidden="true">&times;</span></button> |
|  | </div> |
|  | <div class="modal-body"> |
|  | <img class="img-responsive" id="modal-image" src="" alt="image"> |
|  | </div> |
|  | </div> |
|  | </div> |
|  | </div> |
|  | <script type="text/javascript"> |
|  | $(document).ready(function () { |
|  | var modal = $('#myModal'); |
|  | var modalImage = $('#modal-image'); |
|  |  |
|  | $('.img-responsive').click(function (e) { |
|  | e.preventDefault(); |
|  | modalImage.attr('src', $(this).attr('src')); |
|  | modal.modal('show'); |
|  | }); |
|  | }); |
|  | </script> |
|  | {% include '@App/Image/paginator.html.twig' %} |
|  | {% endblock %} |
|  | {% block title %} |
|  |  |
|  | {% endblock %} |

|  |  |
| --- | --- |
| {% extends '::layout.html.twig' %} |  |
|  |  |
|  | {% block body %} |
|  | <h1>There is no results of your search query!</h1> |
|  | {% endblock %} |
|  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| {% block paginator %} | |  | |
|  | | {% if maxPages > 1 %} | |
|  | | {% set route = app.request.get('\_route') %} | |
|  | | <div class="text-center"> | |
|  | | <ul class="pagination pagination-sm"> | |
|  | | {# `«` arrow #} | |
|  | | <li {{ thisPage == 1 ? 'class="disabled"' }}> | |
|  | | <a href="{{ path(route, {page: thisPage-1 < 1 ? 1 : thisPage-1, limit: limit}) }}">«</a> | |
|  | | </li> | |
|  | |  | |
|  | | {# Render each page number #} | |
|  | | {% for i in 1..maxPages %} | |
|  | | <li {{ thisPage == i ? 'class="active"' }}> | |
|  | | <a href="{{ path(route, {page: i, limit: limit}) }}">{{ i }}</a> | |
|  | | </li> | |
|  | | {% endfor %} | |
|  | |  | |
|  | | {# `»` arrow #} | |
|  | | <li {{ thisPage == maxPages ? 'class="disabled"' }}> | |
|  | | <a href="{{ path(route, {page: thisPage+1 <= maxPages ? thisPage+1 : thisPage, limit: limit}) }}">»</a> | |
|  | | </li> | |
|  | | </ul> | |
|  | | </div> | |
|  | | {% endif %} | |
|  | | {% endblock %}  <?phpnamespace AppBundle\Tests\Controller;use Symfony\Bundle\FrameworkBundle\Test\WebTestCase;class ImageControllerTest extends WebTestCase{ public function testCreate() { $client = static::createClient(); $crawler = $client->request('GET', '/image/create'); }} | |
| <?php |  | |  |
|  | namespace AppBundle\Utils; | |  |
|  |  | |  |
|  | /\*\* | |  |
|  | \* Class ImagesComparer | |  |
|  | \* @package AppBundle\Utils | |  |
|  | \*/ | |  |
|  | class ImagesComparer | |  |
|  | { | |  |
|  | private function mimeType($i) | |  |
|  | { | |  |
|  | $mime = getimagesize($i); | |  |
|  | $return = array($mime[0], $mime[1]); | |  |
|  |  | |  |
|  | switch ($mime['mime']) { | |  |
|  | case 'image/jpeg': | |  |
|  | $return[] = 'jpg'; | |  |
|  | return $return; | |  |
|  | case 'image/png': | |  |
|  | $return[] = 'png'; | |  |
|  | return $return; | |  |
|  | default: | |  |
|  | return false; | |  |
|  | } | |  |
|  | } | |  |
|  |  | |  |
|  | private function createImage($i) | |  |
|  | { | |  |
|  | $mime = $this->mimeType($i); | |  |
|  |  | |  |
|  | if ($mime[2] == 'jpg') { | |  |
|  | return imagecreatefromjpeg($i); | |  |
|  | } else if ($mime[2] == 'png') { | |  |
|  | return imagecreatefrompng($i); | |  |
|  | } else { | |  |
|  | return false; | |  |
|  | } | |  |
|  | } | |  |
|  |  | |  |
|  | private function resizeImage($i, $source) | |  |
|  | { | |  |
|  | $mime = $this->mimeType($source); | |  |
|  |  | |  |
|  | $t = imagecreatetruecolor(8, 8); | |  |
|  |  | |  |
|  | $source = $this->createImage($source); | |  |
|  |  | |  |
|  | imagecopyresized($t, $source, 0, 0, 0, 0, 8, 8, $mime[0], $mime[1]); | |  |
|  |  | |  |
|  | return $t; | |  |
|  | } | |  |
|  |  | |  |
|  | private function colorMeanValue($i) | |  |
|  | { | |  |
|  | $colorList = array(); | |  |
|  | $colorSum = 0; | |  |
|  | for ($a = 0; $a < 8; $a++) { | |  |
|  |  | |  |
|  | for ($b = 0; $b < 8; $b++) { | |  |
|  |  | |  |
|  | $rgb = imagecolorat($i, $a, $b); | |  |
|  | $colorList[] = $rgb & 0xFF; | |  |
|  | $colorSum += $rgb & 0xFF; | |  |
|  |  | |  |
|  | } | |  |
|  |  | |  |
|  | } | |  |
|  |  | |  |
|  | return array($colorSum / 64, $colorList); | |  |
|  | } | |  |
|  |  | |  |
|  | private function bits($colorMean) | |  |
|  | { | |  |
|  | $bits = array(); | |  |
|  |  | |  |
|  | foreach ($colorMean[1] as $color) { | |  |
|  | $bits[] = ($color >= $colorMean[0]) ? 1 : 0; | |  |
|  | } | |  |
|  |  | |  |
|  | return $bits; | |  |
|  |  | |  |
|  | } | |  |
|  |  | |  |
|  | public function compare($a, $b) | |  |
|  | { | |  |
|  | $i1 = $this->createImage($a); | |  |
|  | $i2 = $this->createImage($b); | |  |
|  |  | |  |
|  | if (!$i1 || !$i2) { | |  |
|  | return false; | |  |
|  | } | |  |
|  |  | |  |
|  | $i1 = $this->resizeImage($i1, $a); | |  |
|  | $i2 = $this->resizeImage($i2, $b); | |  |
|  |  | |  |
|  | imagefilter($i1, IMG\_FILTER\_GRAYSCALE); | |  |
|  | imagefilter($i2, IMG\_FILTER\_GRAYSCALE); | |  |
|  |  | |  |
|  | $colorMean1 = $this->colorMeanValue($i1); | |  |
|  | $colorMean2 = $this->colorMeanValue($i2); | |  |
|  |  | |  |
|  | $bits1 = $this->bits($colorMean1); | |  |
|  | $bits2 = $this->bits($colorMean2); | |  |
|  |  | |  |
|  | $hammeringDistance = 0; | |  |
|  |  | |  |
|  | for ($a = 0; $a < 64; $a++) { | |  |
|  |  | |  |
|  | if ($bits1[$a] != $bits2[$a]) { | |  |
|  | $hammeringDistance++; | |  |
|  | } | |  |
|  |  | |  |
|  | } | |  |
|  |  | |  |
|  | return $hammeringDistance; | |  |
|  | } | |  |
|  | } | |  |

|  |  |
| --- | --- |
| <?php |  |
|  |  |
|  | namespace AppBundle; |
|  |  |
|  | use Symfony\Component\HttpKernel\Bundle\Bundle; |
|  |  |
|  | class AppBundle extends Bundle |
|  | { |
|  | } |

|  |  |
| --- | --- |
| .footer { |  |
|  | margin-top: 30px; |
|  | position: absolute; |
|  | bottom: 0; |
|  | } |

|  |  |
| --- | --- |
| # Use the front controller as index file. It serves as a fallback solution when |  |
|  | # every other rewrite/redirect fails (e.g. in an aliased environment without |
|  | # mod\_rewrite). Additionally, this reduces the matching process for the |
|  | # start page (path "/") because otherwise Apache will apply the rewriting rules |
|  | # to each configured DirectoryIndex file (e.g. index.php, index.html, index.pl). |
|  | DirectoryIndex app.php |
|  |  |
|  | # By default, Apache does not evaluate symbolic links if you did not enable this |
|  | # feature in your server configuration. Uncomment the following line if you |
|  | # install assets as symlinks or if you experience problems related to symlinks |
|  | # when compiling LESS/Sass/CoffeScript assets. |
|  | # Options FollowSymlinks |
|  |  |
|  | # Disabling MultiViews prevents unwanted negotiation, e.g. "/app" should not resolve |
|  | # to the front controller "/app.php" but be rewritten to "/app.php/app". |
|  | <IfModule mod\_negotiation.c> |
|  | Options -MultiViews |
|  | </IfModule> |
|  |  |
|  | <IfModule mod\_rewrite.c> |
|  | RewriteEngine On |
|  |  |
|  | # Determine the RewriteBase automatically and set it as environment variable. |
|  | # If you are using Apache aliases to do mass virtual hosting or installed the |
|  | # project in a subdirectory, the base path will be prepended to allow proper |
|  | # resolution of the app.php file and to redirect to the correct URI. It will |
|  | # work in environments without path prefix as well, providing a safe, one-size |
|  | # fits all solution. But as you do not need it in this case, you can comment |
|  | # the following 2 lines to eliminate the overhead. |
|  | RewriteCond %{REQUEST\_URI}::$1 ^(/.+)/(.\*)::\2$ |
|  | RewriteRule ^(.\*) - [E=BASE:%1] |
|  |  |
|  | # Sets the HTTP\_AUTHORIZATION header removed by Apache |
|  | RewriteCond %{HTTP:Authorization} . |
|  | RewriteRule ^ - [E=HTTP\_AUTHORIZATION:%{HTTP:Authorization}] |
|  |  |
|  | # Redirect to URI without front controller to prevent duplicate content |
|  | # (with and without `/app.php`). Only do this redirect on the initial |
|  | # rewrite by Apache and not on subsequent cycles. Otherwise we would get an |
|  | # endless redirect loop (request -> rewrite to front controller -> |
|  | # redirect -> request -> ...). |
|  | # So in case you get a "too many redirects" error or you always get redirected |
|  | # to the start page because your Apache does not expose the REDIRECT\_STATUS |
|  | # environment variable, you have 2 choices: |
|  | # - disable this feature by commenting the following 2 lines or |
|  | # - use Apache >= 2.3.9 and replace all L flags by END flags and remove the |
|  | # following RewriteCond (best solution) |
|  | RewriteCond %{ENV:REDIRECT\_STATUS} ^$ |
|  | RewriteRule ^app\.php(?:/(.\*)|$) %{ENV:BASE}/$1 [R=301,L] |
|  |  |
|  | # If the requested filename exists, simply serve it. |
|  | # We only want to let Apache serve files and not directories. |
|  | RewriteCond %{REQUEST\_FILENAME} -f |
|  | RewriteRule ^ - [L] |
|  |  |
|  | # Rewrite all other queries to the front controller. |
|  | RewriteRule ^ %{ENV:BASE}/app.php [L] |
|  | </IfModule> |
|  |  |
|  | <IfModule !mod\_rewrite.c> |
|  | <IfModule mod\_alias.c> |
|  | # When mod\_rewrite is not available, we instruct a temporary redirect of |
|  | # the start page to the front controller explicitly so that the website |
|  | # and the generated links can still be used. |
|  | RedirectMatch 302 ^/$ /app.php/ |
|  | # RedirectTemp cannot be used instead |
|  | </IfModule> |
|  | </IfModule> |

|  |  |
| --- | --- |
| <?php |  |
|  |  |
|  | /\* |
|  | \* \*\*\*\*\*\*\*\*\*\*\*\*\*\* CAUTION \*\*\*\*\*\*\*\*\*\*\*\*\*\* |
|  | \* |
|  | \* DO NOT EDIT THIS FILE as it will be overridden by Composer as part of |
|  | \* the installation/update process. The original file resides in the |
|  | \* SensioDistributionBundle. |
|  | \* |
|  | \* \*\*\*\*\*\*\*\*\*\*\*\*\*\* CAUTION \*\*\*\*\*\*\*\*\*\*\*\*\*\* |
|  | \*/ |
|  |  |
|  | if (!isset($\_SERVER['HTTP\_HOST'])) { |
|  | exit('This script cannot be run from the CLI. Run it from a browser.'); |
|  | } |
|  |  |
|  | if (!in\_array(@$\_SERVER['REMOTE\_ADDR'], array( |
|  | '127.0.0.1', |
|  | '::1', |
|  | ))) { |
|  | header('HTTP/1.0 403 Forbidden'); |
|  | exit('This script is only accessible from localhost.'); |
|  | } |
|  |  |
|  | require\_once dirname(\_\_FILE\_\_).'/../var/SymfonyRequirements.php'; |
|  |  |
|  | $symfonyRequirements = new SymfonyRequirements(); |
|  |  |
|  | $majorProblems = $symfonyRequirements->getFailedRequirements(); |
|  | $minorProblems = $symfonyRequirements->getFailedRecommendations(); |
|  | $hasMajorProblems = (bool) count($majorProblems); |
|  | $hasMinorProblems = (bool) count($minorProblems); |
|  |  |
|  | ?> |
|  | <!DOCTYPE html> |
|  | <html> |
|  | <head> |
|  | <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/> |
|  | <meta name="robots" content="noindex,nofollow" /> |
|  | <title>Symfony Configuration Checker</title> |
|  | <style> |
|  | /\* styles copied from symfony framework bundle \*/ |
|  | html { |
|  | background: #eee; |
|  | } |
|  | body { |
|  | font: 11px Verdana, Arial, sans-serif; |
|  | color: #333; |
|  | } |
|  | .sf-reset, .sf-reset .block, .sf-reset #message { |
|  | margin: auto; |
|  | } |
|  | img { |
|  | border: 0; |
|  | } |
|  | .clear { |
|  | clear: both; |
|  | height: 0; |
|  | font-size: 0; |
|  | line-height: 0; |
|  | } |
|  | .clear-fix:after { |
|  | content: "\0020"; |
|  | display: block; |
|  | height: 0; |
|  | clear: both; |
|  | visibility: hidden; |
|  | } |
|  | .clear-fix { |
|  | display: inline-block; |
|  | } |
|  | \* html .clear-fix { |
|  | height: 1%; |
|  | } |
|  | .clear-fix { |
|  | display: block; |
|  | } |
|  | .header { |
|  | padding: 30px 30px 20px 30px; |
|  | } |
|  | .header-logo { |
|  | float: left; |
|  | } |
|  | .search { |
|  | float: right; |
|  | padding-top: 20px; |
|  | } |
|  | .search label { |
|  | line-height: 28px; |
|  | vertical-align: middle; |
|  | } |
|  | .search input { |
|  | width: 195px; |
|  | font-size: 12px; |
|  | border: 1px solid #dadada; |
|  | background: #fff url(data:image/gif;base64,) repeat-x left top; |
|  | padding: 5px 6px; |
|  | color: #565656; |
|  | } |
|  | .search input[type="search"] { |
|  | -webkit-appearance: textfield; |
|  | } |
|  | #content { |
|  | width: 970px; |
|  | margin: 0 auto; |
|  | } |
|  | #content pre { |
|  | white-space: normal; |
|  | font-family: Arial, Helvetica, sans-serif; |
|  | } |
|  |  |
|  | /\* |
|  | Copyright (c) 2010, Yahoo! Inc. All rights reserved. |
|  | Code licensed under the BSD License: |
|  | http://developer.yahoo.com/yui/license.html |
|  | version: 3.1.2 |
|  | build: 56 |
|  | \*/ |
|  | .sf-reset div,.sf-reset dl,.sf- input,.sf-reset textarea,.sf-reset select{font-size:100%;}.sf-reset legend{color:#000;} |
|  | .sf-reset abbr { |
|  | border-bottom: 1px dotted #000; |
|  | cursor: help; |
|  | } |
|  | .sf-reset p { |
|  | font-size: 14px; |
|  | line-height: 20px; |
|  | padding-bottom: 20px; |
|  | } |
|  | .sf-reset strong { |
|  | color: #313131; |
|  | font-weight: bold; |
|  | } |
|  | .sf-reset a { |
|  | color: #6c6159; |
|  | } |
|  | .sf-reset a img { |
|  | border: none; |
|  | } |
|  | .sf-reset a:hover { |
|  | text-decoration: underline; |
|  | } |
|  | .sf-reset em { |
|  | font-style: italic; |
|  | } |
|  | .sf-reset h2, |
|  | .sf-reset h3 { |
|  | font-weight: bold; |
|  | } |
|  | .sf-reset h1 { |
|  | font-family: Georgia, "Times New Roman", Times, serif; |
|  | font-size: 20px; |
|  | color: #313131; |
|  | word-wrap: break-word; |
|  | } |
|  | .sf-reset li { |
|  | padding-bottom: 10px; |
|  | } |
|  | .sf-reset .block { |
|  | -moz-border-radius: 16px; |
|  | -webkit-border-radius: 16px; |
|  | border-radius: 16px; |
|  | margin-bottom: 20px; |
|  | background-color: #FFFFFF; |
|  | border: 1px solid #dfdfdf; |
|  | padding: 40px 50px; |
|  | word-break: break-all; |
|  | } |
|  | .sf-reset h2 { |
|  | font-size: 16px; |
|  | font-family: Arial, Helvetica, sans-serif; |
|  | } |
|  | .sf-reset li a { |
|  | background: none; |
|  | color: #868686; |
|  | text-decoration: none; |
|  | } |
|  | .sf-reset li a:hover { |
|  | background: none; |
|  | color: #313131; |
|  | text-decoration: underline; |
|  | } |
|  | .sf-reset ol { |
|  | padding: 10px 0; |
|  | } |
|  | .sf-reset ol li { |
|  | list-style: decimal; |
|  | margin-left: 20px; |
|  | padding: 2px; |
|  | padding-bottom: 20px; |
|  | } |
|  | .sf-reset ol ol li { |
|  | list-style-position: inside; |
|  | margin-left: 0; |
|  | white-space: nowrap; |
|  | font-size: 12px; |
|  | padding-bottom: 0; |
|  | } |
|  | .sf-reset li .selected { |
|  | background-color: #ffd; |
|  | } |
|  | .sf-button { |
|  | display: -moz-inline-box; |
|  | display: inline-block; |
|  | text-align: center; |
|  | vertical-align: middle; |
|  | border: 0; |
|  | background: transparent none; |
|  | text-transform: uppercase; |
|  | cursor: pointer; |
|  | font: bold 11px Arial, Helvetica, sans-serif; |
|  | } |
|  | .sf-button span { |
|  | text-decoration: none; |
|  | display: block; |
|  | height: 28px; |
|  | float: left; |
|  | } |
|  | .sf-button .border-l { |
|  | text-decoration: none; |
|  | display: block; |
|  | height: 28px; |
|  | float: left; |
|  | padding: 0 0 0 7px; |
|  | background: transparent url(data:image/png; ///4ME5wAAAABJRU5ErkJggg==) no-repeat top left; |
|  | } |
|  | .sf-button .border-r { |
|  | padding: 0 7px 0 0; |
|  | background: transparent url(data:image/png;base64, /U7AAQYACJ2vxVdJW4eQAAAABJRU5ErkJggg==) right top no-repeat; |
|  | } |
|  | .sf-button .btn-bg { |
|  | padding: 0 14px; |
|  | color: #636363; |
|  | line-height: 28px; |
|  | background: transparent url(data:image/png;AAAAAuQmCC) repeat-x top left; |
|  | } |
|  | .sf-button:hover .border-l, |
|  | .sf-button-selected .border-l { |
|  | background: transparent url(data:image/png;base64, FTkSuQmCC) no-repeat top left; |
|  | } |
|  | .sf-button:hover .border-r, |
|  | .sf-button-selected .border-r { |
|  | background: transparent url(data:image/png;base64, rkJggg==) right top no-repeat; |
|  | } |
|  | .sf-button:hover .btn-bg, |
|  | .sf-button-selected .btn-bg { |
|  | color: #FFFFFF; |
|  | text-shadow:0 1px 1px #6b9311; |
|  | background: transparent url(data:image/png; repeat-x top left; |
|  | } |
|  |  |
|  | /\* styles copied from bundles/sensiodistribution/webconfigurator/css/install.css \*/ |
|  | body { |
|  | font-size: 14px; |
|  | font-family: "Lucida Sans Unicode", "Lucida Grande", Verdana, Arial, Helvetica, sans-serif; |
|  | } |
|  | .sf-reset h1.title { |
|  | font-size: 45px; |
|  | padding-bottom: 30px; |
|  | } |
|  | .sf-reset h2 { |
|  | font-weight: bold; |
|  | color: #FFFFFF; |
|  | /\* Font is reset to sans-serif (like body) \*/ |
|  | font-family: "Lucida Sans Unicode", "Lucida Grande", Verdana, Arial, Helvetica, sans-serif; |
|  | margin-bottom: 10px; |
|  | background-color: #aacd4e; |
|  | padding: 2px 4px; |
|  | display: inline-block; |
|  | text-transform: uppercase; |
|  | } |
|  | .sf-reset ul a, |
|  | .sf-reset ul a:hover { |
|  | background: url(data:image/png; ==) no-repeat right 7px; |
|  | padding-right: 10px; |
|  | } |
|  | .sf-reset ul, ol { |
|  | padding-left: 20px; |
|  | } |
|  | .sf-reset li { |
|  | padding-bottom: 18px; |
|  | } |
|  | .sf-reset ol li { |
|  | list-style-type: decimal; |
|  | } |
|  | .sf-reset ul li { |
|  | list-style-type: none; |
|  | } |
|  | .sf-reset .symfony-blocks-install { |
|  | overflow: hidden; |
|  | } |
|  | .sf-reset .symfony-install-continue { |
|  | font-size: 0.95em; |
|  | padding-left: 0; |
|  | } |
|  | .sf-reset .symfony-install-continue li { |
|  | padding-bottom: 10px; |
|  | } |
|  | .sf-reset .ok { |
|  | color: #fff; |
|  | font-family: "Lucida Sans Unicode", "Lucida Grande", Verdana, Arial, Helvetica, sans-serif; |
|  | background-color: #6d6; |
|  | padding: 10px; |
|  | margin-bottom: 20px; |
|  | } |
|  | .sf-reset .ko { |
|  | background-color: #d66; |
|  | } |
|  | .sf-reset p.help { |
|  | padding: 12px 16px; |
|  | word-break: break-word; |
|  | } |
|  | .version { |
|  | text-align: right; |
|  | font-size: 10px; |
|  | margin-right: 20px; |
|  | } |
|  | .sf-reset a, |
|  | .sf-reset li a { |
|  | color: #08C; |
|  | text-decoration: none; |
|  | } |
|  | .sf-reset a:hover, |
|  | .sf-reset li a:hover { |
|  | color: #08C; |
|  | text-decoration: underline; |
|  | } |
|  | .sf-reset textarea { |
|  | padding: 7px; |
|  | } |
|  | </style> |
|  | </head> |
|  | <body> |
|  | <div id="content"> |
|  | <div class="header clear-fix"> |
|  | <div class="header-logo"> |
|  | </div> |
|  |  |
|  | <div class="search"> |
|  | <form method="get" action="http://symfony.com/search"> |
|  | <div class="form-row"> |
|  |  |
|  | <label for="search-id"> |
|  |  |
|  |  |
|  |  |
|  | <input name="q" id="search-id" type="search" placeholder="Search on Symfony website" /> |
|  |  |
|  | <button type="submit" class="sf-button"> |
|  | <span class="border-l"> |
|  | <span class="border-r"> |
|  | <span class="btn-bg">OK</span> |
|  | </span> |
|  | </span> |
|  | </button> |
|  | </div> |
|  | </form> |
|  | </div> |
|  | </div> |
|  |  |
|  | <div class="sf-reset"> |
|  | <div class="block"> |
|  | <div class="symfony-block-content"> |
|  | <h1 class="title">Configuration Checker</h1> |
|  | <p> |
|  | This script analyzes your system to check whether is |
|  | ready to run Symfony applications. |
|  | </p> |
|  |  |
|  | <?php if ($hasMajorProblems): ?> |
|  | <h2 class="ko">Major problems</h2> |
|  | <p>Major problems have been detected and <strong>must</strong> be fixed before continuing:</p> |
|  | <ol> |
|  | <?php foreach ($majorProblems as $problem): ?> |
|  | <li><?php echo $problem->getTestMessage() ?> |
|  | <p class="help"><em><?php echo $problem->getHelpHtml() ?></em></p> |
|  | </li> |
|  | <?php endforeach; ?> |
|  | </ol> |
|  | <?php endif; ?> |
|  |  |
|  | <?php if ($hasMinorProblems): ?> |
|  | <h2>Recommendations</h2> |
|  | <p> |
|  | <?php if ($hasMajorProblems): ?>Additionally, to<?php else: ?>To<?php endif; ?> enhance your Symfony experience, |
|  | it’s recommended that you fix the following: |
|  | </p> |
|  | <ol> |
|  | <?php foreach ($minorProblems as $problem): ?> |
|  | <li><?php echo $problem->getTestMessage() ?> |
|  | <p class="help"><em><?php echo $problem->getHelpHtml() ?></em></p> |
|  | </li> |
|  | <?php endforeach; ?> |
|  | </ol> |
|  | <?php endif; ?> |
|  |  |
|  | <?php if ($symfonyRequirements->hasPhpIniConfigIssue()): ?> |
|  | <p id="phpini">\* |
|  | <?php if ($symfonyRequirements->getPhpIniConfigPath()): ?> |
|  | Changes to the <strong>php.ini</strong> file must be done in "<strong><?php echo $symfonyRequirements->getPhpIniConfigPath() ?></strong>". |
|  | <?php else: ?> |
|  | To change settings, create a "<strong>php.ini</strong>". |
|  | <?php endif; ?> |
|  | </p> |
|  | <?php endif; ?> |
|  |  |
|  | <?php if (!$hasMajorProblems && !$hasMinorProblems): ?> |
|  | <p class="ok">All checks passed successfully. Your system is ready to run Symfony applications.</p> |
|  | <?php endif; ?> |
|  |  |
|  | <ul class="symfony-install-continue"> |
|  | <?php if ($hasMajorProblems || $hasMinorProblems): ?> |
|  | <li><a href="config.php">Re-check configuration</a></li> |
|  | <?php endif; ?> |
|  | </ul> |
|  | </div> |
|  | </div> |
|  | </div> |
|  | <div class="version">Symfony Standard Edition</div> |
|  | </div> |
|  | </body> |
|  | </html> |

|  |  |
| --- | --- |
| <!DOCTYPE html> |  |
|  | <html> |
|  | <head> |
|  | <meta charset="UTF-8" /> |
|  | <title>{% block title %}Welcome!{% endblock %}</title> |
|  | {% block stylesheets %}{% endblock %} |
|  | <link rel="icon" type="image/x-icon" href="{{ asset('favicon.ico') }}" /> |
|  | </head> |
|  | <body> |
|  | {% block body %}{% endblock %} |
|  | {% block javascripts %}{% endblock %} |
|  | </body> |
|  | </html> |

|  |  |
| --- | --- |
| {% block footer %} |  |
|  | <footer class="footer"> |
|  | <p>&copy; 2017 Search Images by Myroslav Matrafailo</p> |
|  | </footer> |
|  | {% endblock %} |

|  |  |
| --- | --- |
| {% block header %} |  |
|  | <div class="header clearfix"> |
|  | <nav> |
|  | <ul class="nav nav-pills pull-right"> |
|  | <li role="presentation"><a href="{{ path('app\_image\_upload') }}">Upload</a></li> |
|  | <li role="presentation"><a href="{{ path('app\_image\_imagessearch') }}">Search</a></li> |
|  | </ul> |
|  | </nav> |
|  | <h3 class="text-muted"><a href="{{ path('app\_image\_list') }}">Search Images</a></h3> |
|  | </div> |
|  | {% endblock %} |

|  |  |
| --- | --- |
| {% block header %} |  |
|  | <div class="header clearfix"> |
|  | <nav> |
|  | <ul class="nav nav-pills pull-right"> |
|  | <li role="presentation"><a href="{{ path('app\_image\_upload') }}">Upload</a></li> |
|  | <li role="presentation"><a href="{{ path('app\_image\_imagessearch') }}">Search</a></li> |
|  | </ul> |
|  | </nav> |
|  | <h3 class="text-muted"><a href="{{ path('app\_image\_list') }}">Search Images</a></h3> |
|  | </div> |
|  | {% endblock %} |

|  |  |
| --- | --- |
| <!DOCTYPE html> |  |
|  | <html lang="en"> |
|  | <head> |
|  | <meta charset="utf-8"> |
|  | <meta http-equiv="X-UA-Compatible" content="IE=edge"> |
|  | <meta name="viewport" content="width=device-width, initial-scale=1"> |
|  | <!-- The above 3 meta tags \*must\* come first in the head; any other head content must come \*after\* these tags --> |
|  | <meta name="description" content=""> |
|  | <meta name="author" content=""> |
|  | <link rel="icon" href="{{ asset('apple-touch-icon.png') }}"> |
|  |  |
|  | <title>Images Search</title> |
|  |  |
|  | <!-- Bootstrap core CSS --> |
|  | <link href="{{ asset('css/bootstrap.min.css') }}" rel="stylesheet"> |
|  |  |
|  | <!-- IE10 viewport hack for Surface/desktop Windows 8 bug --> |
|  | <link href="{{ asset('css/ie10-viewport-bug-workaround.css') }}" rel="stylesheet"> |
|  |  |
|  | <!-- Custom styles for this template --> |
|  | <link href="{{ asset('css/jumbotron-narrow.css') }}" rel="stylesheet"> |
|  | <link href="{{ asset('css/main.css') }}" rel="stylesheet"> |
|  | <script type="text/javascript" src="{{ asset('js/jquery-3.1.1.min.js') }}"></script> |
|  | <script type="text/javascript" src="{{ asset('js/bootstrap.min.js') }}"></script> |
|  | {% block css %}{% endblock %} |
|  | </head> |
|  |  |
|  | <body> |
|  |  |
|  | <div class="container"> |
|  | {% include '::header.html.twig' %} |
|  | {% block body %}{% endblock %} |
|  | {%include '::footer.html.twig' %} |
|  | </div> <!-- /container --> |
|  |  |
|  | {% block javascripts %}{% endblock %} |
|  | </body> |
|  | </html> |