Міністерство освіти і науки України
Запорізький національний технічний університет

МЕТОДИЧНІ ВКАЗІВКИ
до лабораторних робіт
з дисципліни
"ПРОЕКТУВАННЯ ТА РОЗРОБКА WEB-ДОДАТКІВ"
для студентів спеціальності
121 "Інженерія програмного забезпечення"
всіх форм навчання

2017

Методичні вказівки до лабораторних робіт з дисципліни "Проектування та розробка Web-додатків" для студентів спеціальності 121 "Інженерія програмного забезпечення" всіх форм навчання / Укл. Т.В. Федорончак. – Запоріжжя: ЗНТУ, 2015. – 38 с.

Автори: Т.В. Федорончак, к.т.н., доцент

Рецензент: А.О. Олійник, к.т.н., доцент, доцент

Відповідальний
за випуск: С.О. Субботін, д.т.н., професор

Затверджено
на засіданні кафедри
програмних засобів

Протокол №11
від "10" липня 2017 р.

ЗМІСТ

ВСТУП

Практична частина курсу "Проектування та розробка Web-додатків" складається з десяти лабораторних робіт та розрахунково-графічного завдання і призначена для отримання практичних навичок, необхідних для вирішення питань, пов'язаних із проектуванням та розробкою клієнт-серверних додатків у глобальній мережі Інтернет з використання сучасних інструментальних засобів, технологій, мов програмування та загальноприйнятих стандартів.

Лабораторні роботи виконуються студентами в якості командного проекту групами по 3-4 чоловіки.

Розрахунково-графічне завдання є індивідуальним та здається кожним студентом окремо.

Для виконання практичних завдань з курсу використовується наступне програмне забезпечення:

– XAMPP - багатоплатформова збірка Web-сервера, що містить Apache, MySQL, інтерпретатор скриптів PHP, мову програмування Perl та велику кількість додаткових бібліотек, що дозволяють запустити повноцінний Web-сервер;

– браузери Mozilla Firefox, Google Chrome, Opera, Internet Explorer зі встановленими інструментами для розробників (наприклад, FireBug);

– Komodo Edit 10 (чи інша IDE для веб-розробки, в тому числі демонстраційні версії платних застосунків, наприклад, phpStorm);

– MySQL Workbench - інструмент для візуального проектування баз даних, що інтегрує проектування, моделювання, створення й експлуатацію БД в єдине безшовне оточення для системи баз даних MySQL;

– phpMyAdmin - Web-додаток з відкритим кодом на мові PHP із графічним Web-інтерфейсом для адміністрування СКБД MySQL;

– GitGui - клієнт для системи керування версіями Git.

Розробка студентами Web-додатків виконується на основі безкоштовного MVC фреймворку CodeIgniter.

Загальна оцінка з курсу формується з наступних складових:

– виконання лабораторних робіт, тобто командний проект (50%);

– РГЗ (20%);

– письмовий екзамен (30%).

Лабораторна робота № 1
Постановка задачі проектування Web-додатку: розробка вимог до проекту

1.1 Мета роботи

Описати предметну область та розробити специфікацію вимог до Web-додатку.

1.2 Основні теоретичні відомості

Web-додаток - це клієнт-серверний додаток, в якому клієнтом виступає браузер, а сервером - Web-сервер. Логіка Web-додатку розподілена між сервером і клієнтом, збереження даних здійснюється переважно на сервері, обмін інформацією виконується мережею Інтернет. Однією з переваг такого підходу є той факт, що клієнти не залежать від конкретної операційної системи користувача, і тому Web-додатки є міжплатформними сервісами.

Границя між Web-сайтами та Web-додатками досить розмита.

Web-сайти переважно є інформаційними та служать для поширення інформації про компанію, продукт, послугу, або просто надають довідкову інформацію.

Якщо ж сайт є інтерактивним, тобто користувач не є пасивним споживачем, а є активним учасником сайту: шукає інформацію, натискає кнопки, заповнює форми, робить покупки та постійно працює з мишкою та клавіатурою - можна говорити про web-додаток. ,

Аналіз вимог [1] є першим етапом проектування та розробки будь-яких додатків, та є критичним для успішної розробки проекту. Вимоги мають бути задокументованими, вимірними, тестовними, пов'язаними з бізнес-потребами і описаними з рівнем деталізації достатнім для конструювання системи. Вимоги можуть бути архітектурними, структурними, поведінковими, функціональними, та не функціональними.

Специфікація вимог до програмного забезпечення (Software Requirements Specification, SRS) - це повний опис поведінки системи що розробляється. Вона включає множину прецедентів які описують всі взаємодії, які користувачі мають з програмним забезпеченням. Прецеденти також відомі як функціональні вимоги. На додачу до прецедентів SRS також включає нефункціональні (чи додаткові)

вимоги. Нефункціональні вимоги є вимогами які накладають обмеження на проект, чи реалізацію (такі як вимоги інженерії продуктивності, стандарти якості, чи обмеження проектування).

Детальна інформація щодо розробки Web-додатків [2] за допомогою мови PHP може бути знайдена за посиланнями [3-7].

1.3 Завдання на лабораторну роботу

1.3.1 Розбитися на команди по 3-4 студенти.

1.3.2 Обрати предметну область, визначити завдання на розробку Web-додатку та узгодити його з викладачем.

До прикладів розроблюваних web-додатків можна віднести наступні:

- система керування вмістом сайту(CMS);
- форум;
- Web-інтерфейс до бази даних;
- електронний магазин;
- система дистанційного навчання;
- система он-лайн опитування;
- система відеопрокату;
- система бронювання квитків кінотеатру,
- система розкладу занять кафедри.

1.3.3 Розробити специфікацію вимог до Web-додатку. Для опису програмної системи, що розробляється, та її функцій навести структурну діаграму, діаграми потоків даних, діаграми прецедентів, діаграми активності.

1.4 Зміст звіту

1.4.1 Тема та мета роботи.

1.4.2 Результати роботи.

1.4.3 Висновки, що відображують результати виконання роботи та їх критичний аналіз.

Лабораторна робота № 2
Проектування бази даних web-додатку

2.1 Мета роботи

Спроектувати та розробити базу даних Web-додатку за допомогою сучасних засобів web-програмування.

2.2 Основні теоретичні відомості

2.2.1 Проектування схеми бази даних web-додатку

Діаграми «сутність-зв'язок» (entity-relationship diagram, ERD) призначені для опису концептуальної схеми предметної області та розробки моделей даних. Ці діаграми забезпечують стандартний спосіб визначення даних та відношень між ними [8].

За допомогою ER-діаграм здійснюється деталізація сховищ даних проектованої системи, а також документуються об'єкти, важливі для предметної області (сутності), властивостей цих об'єктів (атрибути) та їх відносин з іншими об'єктами (зв'язки).

ER-діаграма дозволяє розглянути систему цілком і з'ясувати необхідні для її розробки вимоги, що стосуються зберігання інформації. Під час проектування баз даних відбувається перетворення ER-моделі в конкретну схему бази даних на основі обраної моделі даних (реляційної, об'єктної, мережевої, тощо).

Розробка ER-моделі предметної області включає наступні етапи.

Етап 1. Проводиться ідентифікація сутностей, їх первинних і альтернативних ключів, при необхідності основних атрибутів. Якщо отримана схема володіє надмірністю, то відбувається спрощення схеми шляхом нормалізації. Цей етап є визначальним при побудові моделі.

Етап 2 служить для виявлення і визначення відносин між сутностями, визначенні атрибутів відношень за необхідності, а також для ідентифікації типів цих відносин (один-до-одного, один-до-багатьох, багато-до-багатьох). Деякі відносини на даному етапі можуть бути неспецифічними (багато-до-багатьох). Такі відносини вимагатимуть подальшої деталізації на етапі 3 .

Визначення відносин включає виявлення зв'язків, для цього відношення має бути перевірено в обох напрямках таким чином: вибирається примірник однієї з сутностей і визначається , скільки

різних примірників другої сутності може бути пов'язано з ним і навпаки.

Етап 3 полягає у вирішенні неспецифічних відносин. Для цього кожне неспецифічне відношення перетворюється на два специфічних з введенням нових (простих або асоціативних) сутностей.

На етапі 4 проводиться категоризація сутностей і визначення всіх необхідних атрибутів сутностей.

2.2.2 Середовища MySQL Workbench та phpMyAdmin

MySQL Workbench - інструмент для візуального проектування баз даних, що інтегрує проектування, моделювання, створення й експлуатацію БД в єдине безшовне оточення для системи баз даних MySQL.

phpMyAdmin - це web-додаток з відкритим кодом на мові PHP із графічним web-інтерфейсом для адміністрування СКБД MySQL. phpMyAdmin дозволяє через браузер здійснювати адміністрування сервера MySQL, запускати запити SQL, переглядати та редагувати вміст таблиць баз даних. Ця програма користується великою популярністю у web-розробників, оскільки дозволяє керувати СКБД MySQL без безпосереднього вводу SQL команд через дружній інтерфейс і з будь-якого комп'ютера під'єднаного до Інтернету без необхідності встановлення додаткового програмного забезпечення.

2.3 Завдання на лабораторну роботу

2.3.1 Розробити логічну модель даних для Web-додатку (ER-діаграму).

2.3.2 На основі логічної моделі даних спроектувати реляційну схему бази даних Web-додатку. Для проектування використати програму MySQL Workbench.

2.3.3 Створити розроблену базу даних в середовищі phpMyAdmin.

2.4 Зміст звіту

2.4.1 Тема та мета роботи.

2.4.2 Результати роботи.

2.4.3 Висновки, що відображують результати виконання роботи та їх критичний аналіз.

Лабораторна робота № 3
Розробка інтерфейсу користувача web-додатку за допомогою HTML та CSS

3.1 Мета роботи

Розробити користувацький інтерфейс Web-додатку за допомогою мови розмітки HTML та каскадних таблиць стилів CSS.

3.2 Основні теоретичні відомості

Розмітка (верстка) Web-сторінки - це процес створення Web-сторінки засобами мови HTML та CSS із попередньо створеного макету дизайну сайту, заздалегідь намальованого за допомогою графічних редакторів.

Таблична розмітка раніше була основним методом створення Web-сторінки (HTML3). Однак сьогодні вона використовується в основному лише для відображення табличних даних.

Розмітка за допомогою блоків (тег) та каскадних таблиць стилів (CSS), що описують їхнє оформлення, реалізує концепцію семантичної розмітки. При цьому розділяється зміст сторінки та її оформлення. Концепція семантичної розмітки найшла найбільше втілення в мові HTML5.

Всі макети сайтів, тобто типи розмітки сторінок можна розділити на три принципові групи:

- фіксовані (fixed),
- гумові (fluid),
- адаптивні (responsive).

Фіксований тип макету - дизайн, в якому ширину стовпця/рисунку задано точно в пікселах.

Переваги:

- легко розробляти;
- заздалегідь відомо як буде виглядати сайт.

Недоліки:

- для сайту існує лише один ідеальний розмір екрану;
- в деяких випадках може з'явитися скроллінг.

Гумовий тип макету – дизайн, в якому ширину стовпця/рисунку задано у відсотках від поточного розміру екрану чи батьківського елементу.

Переваги:

– сайт буде заповнювати весь простір браузера, що значно поліпшує його вигляд;

– сайт буде однаково виглядати на різних розмірах екрану.

Недоліки:

– розробка такого дизайну дуже складна;

– на великих екранах можлива поява дуже довгих строк.

Адаптивний тип макету - дизайн, якій підлаштовується (адаптується) під розмір екрану, в том числі може відбуватися переміщення блоків з одного місця на інше, або їхня заміна блоками, що відображаються лише при визначеній роздільній здатності. Адаптивна верстка прийшла на зміну ідеї створення спеціальних мобільних версій сайтів, розташованих на окремих піддоменах (наприклад, m.wikipedia.org).

Переваги:

– сайт буде відображатися на різних розмірах екрану так, як це найбільш зручно користувачеві.

Недоліки:

– потребує старанного опрацювання декількох макетів для різних розмірів екранів.

Також може використовуватися змішаний тип дизайну.

Шари (або ж блоки) представляють собою структурні елементи, які можна розміщувати на Web-сторінці шляхом накладання їх один на одного з точністю до піксела.

Скрипти (наприклад, на мові JavaScript) дозволяють динамічно змінювати параметри шарів. Шари можна переміщувати, ховати та відображати без перезавантаження всієї сторінки. Це надає можливість створювати на сторінці різні ефекти, такі як анімація, випадаюче меню, спливаючі підказки, плаваючі вікна тощо.

Валідність HTML-верстки – це її відповідність стандартам організації The World Wide Web Consortium (W3C). Відсутність помилок в розмітці документу - це один з головних показників якості розмітки. Перевірка розмітки на помилки та відповідність стандарту може бути проведена автоматично за допомогою он-лайн сервісу W3C чи за допомогою різних програм «валідаторів».

Детальну інформацію щодо мов HTML та CSS можна знайти за посиланням [9].

3.3 Завдання на лабораторну роботу

3.3.1 Навчитися позиціювати блочні елементи сторінок. Навчитися формувати макети Web-сторінок фіксованої ширини та такі, що розтягуються; макети з 1, 2 чи 3 колонок.

3.3.2 Для обраної предметної області розробити макет користувацького інтерфейсу Web-додатку, тобто визначити необхідні функціональні блоки, кількість колонок, керуючі елементи, розташування меню.

3.3.3 Розмітити макет сайту.

3.3.4 Перевірити відображення макету в різних браузерах та виправити за необхідності помилки. При роботі для відлагодження помилок скористатися вбудованими в браузери інспекторами коду.

3.3.5 Провести валідацію макету.

3.4 Зміст звіту

3.4.1 Тема та мета роботи.

3.4.2 Результати роботи.

3.4.3 Висновки, що відображують результати виконання роботи та їх критичний аналіз.

Лабораторна робота № 4
Знайомство з мовою PHP та MVC фреймворком CodeIgniter

4.1 Мета роботи

Отримати практичні навички Web-програмуванння за допомогою мови PHP та ознайомитися з фреймворком CodeIgniter.

4.2 Основні теоретичні відомості

4.2.1 Клієнт-серверна архітектура Web-додатків

Сучасні Web-додатки використовують клієнт-серверну архітектуру, яка за останні десятиліття стала одним із основних способів організації розподілених програмних систем.

При такій організації клієнтом виступає Web-браузер (Mozilla Firefox, Google Chrome, Opera, Internet Explorer, Apple Safari тощо), а у якості серверного додатку виступає Web-сервер (Apache HTTP Server, Microsoft Internet Information Services тощо).

Задачею Web-браузера є візуалізація Web-документів (HTML-сторінок) та мультимедійних об‘єктів, з яких вони складаються. В свою чергу Web-сервер виконує роль оброблювача запитів від віддалених Web-клієнтів. Обробкою може бути отримання статичної Web-сторінки безпосередньо із HTML-файлу, генерація динамічного вмісту із використанням спеціалізованих та універсальних мов програмування (PHP, Python, Java та інші), передача даних від користувача на Web-сервер, зокрема, файлів, а також оновлення визначеного фрагменту вже відображеної Web-сторінки (так звана Ajax-технологія).

В подібній ситуації протоколом обміну або способом формування запиту та генерації відповідей виступає HTTP (HyperText Transfer Protocol) – протокол обміну гіпертексту. Під гіпертекстом розуміють зв‘язані між собою за допомогою спеціальної адресації (URL) документи в мережі Internet. Основними рисами протоколу HTTP є наступне.

1. Обмін даними відбувається за схемою «запит»-«відповідь», отже кожну транзакцію обміну даними ініціює Web-клієнт, а Web-сервер генерує відповідь.

2. HTTP відноситься до верхнього (прикладного) рівня моделі відкритих систем OSI, тому його синтаксис найбільш наближений до

природної мови.

3. HTTP не зберігає історію попередніх транзакцій, тому збереження стану реалізують інші засоби (зокрема, на основі cookie-змінних).

4. Кожний ресурс, доступ до якого виконується завдяки HTTP, ідентифікується через URL (unified resource locator) – загальноприйнятий спосіб адресації в Internet.

5. Запит HTTP складається із рядка запиту («метод» «адреса ресурсу» «версія HTTP»), заголовків клієнта («назва»: «значення») і, опціонально, тіла запиту.

6. Відповідь HTTP складається із статусного рядка («версія HTTP», «код статусу», «опис коду статусу»), заголовків сервера («назва»: «значення») і, опціонально, тіла відповіді.

Обмін інформацією відбувається із використанням методів HTTP: GET, POST, PUT, HEAD, TRACE, OPTIONS, CONNECT, DELETE.

Заголовки HTTP уточнюють запит або відповідь і виконують службову роль при інтерпретації пакетів. Заголовки HTTP розрізняють наступним чином:

– заголовки клієнта (конфігурація клієнта, адреса сервера, тип, кодування, мова вмісту ресурсу, параметри кешування тощо);

– заголовки сервера (серверна конфігурація, перенаправлення, встановлення Cookie, параметри аутентифікації тощо);

– заголовки вмісту повідомлення (MIME-тип повідомлення, його розмір, параметри мови, кодування, ущільнення повідомлення тощо);

– загальні заголовки (дата відправлення/отримання повідомлення, сторінка, з якої виконувався перехід, управління кешуванням та з‘єднанням тощо).

Протокол HTTP за своєю природою не орієнтований на збереження передісторії попередніх станів обміну повідомленнями між клієнтом і сервером. Тому в цей протокол уведено два спеціальні заголовки (Set-Cookie та Cookie), які надають можливості серверу встановлювати («залишати») на клієнті невеликі обсяги даних (до 4096 байтів), а клієнту після цього з кожним запитом повертати збережені дані. Таким чином, сервер може однозначно і унікально ідентифікувати кожного клієнта. Фізично ці дані зберігаються у спеціальному конфігураційному файлі Web-браузера, а час

актуальності цих даних встановлює сервер за допомогою спеціального синтаксису заголовка Set-cookie.

Збереження стану характерне для таких Web-додатків як клієнти електронної пошти, Internet-магазини, інформаційні портали тощо.

Важливим для розробника Web-додатків аспектом протоколу HTTP є коди статусу обробки запиту клієнта (наприклад, 200 OK, 301 Moved permanently, 403 Forbidden, 404 Not found тощо).

Таким чином, використовуючи специфічні властивості протоколу HTTP (правила адресації, формат пакету, заголовки, методи та коди статусу), реалізуються різноманітні способи взаємодії Web-клієнта та Web-браузера із використанням спеціалізованих мов програмування та розмітки (PHP, Python, HTML, XML тощо).

4.2.2 Мова програмування PHP

Мова PHP (Personal Home Pages або Preprocessor Hypertext Preprocessor) за останнє десятиліття набула статусу найбільш вживаної для написання програмного коду з боку Web-сервера. Причиною тому є такі її переваги:

– простий синтаксис, в основному запозичений із традиційних мов програмування C та C++, об'єктно-орієнтовані можливості, а також слабка типізація;

– наявність механізму автоматичного збирання сміття, що спрощує виконання операції вивільнення пам'яті;

– наявність великої кількості бібліотечних модулів, інтегрованих у інтерпретатор (для роботи із файлами та мережею, обробки масивів та рядків; модулі шифрування, обробки графічних файлів, взаємодії з Internet-службами, Web-сервісами тощо);

– наявність засобів взаємодії з найбільш популярними СУБД (MySQL, Oracle, PostrgeSQL та іншими);

– наявність дистрибутивів для більшості сучасних операційних систем (Unix, Linux, Windows, MacOS) та спрощена інтеграція із Web-сервером Apache.

Код мовою PHP створюється у довільному текстовому редакторі, а файл програми зберігається із розширенням php (наприклад, index.php). Крім того, у код PHP може бути вбудований і HTML-код сторінки. В такому випадку HTML-інструкції будуть ігноруватись інтерпретатором і передаватимуться клієнту без змін. Для виділення PHP-коду, його необхідно огорнути у спеціальні теги:

<?php програмний код ?>.

Інформація щодо мови PHP можна знайти за посиланням [3-7].

4.2.3 Поняття PHP фреймворку

Останнім часом основною тенденцією при розробці Web-додатків є використання різноманітних програмних бібліотек, які прискорюють процес створення кінцевого продукту, а також надають можливості систематизувати програмний код за рахунок застосування шаблонів проектування, зокрема, «Модель-Подання-Контролер» (MVC).

Існує велика кількість подібних бібліотек, найбільш універсальні та потужні часто називають каркасами розробки або фреймворками (frameworks).

Фреймворк - в інформаційних системах структура програмної системи; програмне забезпечення, що полегшує розробку і поєднання різних компонентів великого програмного проекту. На відміну від бібліотек, які об'єднують набір підпрограм близької функціональності, фреймворк містить в собі велику кількість різних за призначенням бібліотек. Можна також говорити про каркасному підході як про підхід до побудови програм, де будь-яка конфігурація програми будується з двох частин: перша, постійна частина - каркас, що не змінюється від конфігурації до конфігурації і несе в собі гнізда, в яких розміщується друга, змінна частина - змінні модулі (або точки розширення).

PHP фреймворки стали базовою платформою для розробки Web-додатків. Вони забезпечують основну структуру програми. Використання PHP-фреймворків, дозволяє економити велику кількість часу, зменшити навантаження на процес розробки, позбутися проблеми повторюваного коду, і швидко створювати додатки.

Серед фреймворків для мови PHP слід відзначити такі як Zend Framework, CakePHP, Code Igniter, Symfony та Yii.

Zend Framework та Symfony, надають максимальні можливості порівняно з іншими, але є складними у вивченні та мають меншу швидкодію; інші – простіші у застосуванні, але надають менший набір функціональних можливостей.

Узагальнена характеристика фреймворків наведена нижче.

1. Орієнтація на сучасні об‘єктно-орієнтовані підходи

організації Web-додатків, що надає змогу будувати масштабовані, контрольовані та надійні проекти.

2. Використання шаблону MVC, який надає можливості розділити програмний код взаємодії з даними (model), візуальне подання у вигляді шаблонів сторінок (view), та узагальнений алгоритм (бізнес-логіку) обробки запиту користувача (controller), який безпосередньо використовує моделі та подання проекту.

3. Наявність уніфікованого програмного інтерфейсу до реляційних баз даних, що надає можливість використовувати різні СКБД єдиним способом.

4. Наявність мови шаблонів, що надає можливості максимально розділити HTML-шаблон сторінки від даних, які наповнюють її інформаційним змістом.

5. Використання вбудованого механізму кешування, що забезпечує прискорення завантаження вже переглянутих сторінок.

6. Використання засобів валідації даних, переданих від користувача з HTML-форм.

7. Застосування механізму зручної для користувача адресації сторінок Web-додатку з можливістю відсікання небажаних або хибних адресів.

8. Автоматичне шифрування Cookie-змінних, які надсилаються з сервера, а також управління сесіями.

9. Наявність великої кількості бібліотечних функцій, які автоматизують найбільш вживані операції з даними Web-додатку.

10. Фільтрація вхідних даних від користувача для уникнення зловмисних дій.

4.2.4 Шаблон MVC

На сьогодні більшість PHP проектів побудовані за допомогою архітектури MVC. MVC - це архітектурний шаблон проектування, який дозволяє відокремити бізнес-логіку від користувальницького інтерфейсу, а так само виділити область логіки яка виробляє обмін інформацією між базою даних і призначеним для користувача інтерфейсом. Таким чином можна змінити логіку програми, не зачіпаючи інтерфейсної частини, або навпаки.

В шаблоні MVC (рис. 4.2) модель (model) – це та частина архітектури, яка взаємодіє з базою даних, подання (view) - представляє ту частину, яку безпосередньо бачить користувач, тобто

користувацький інтерфейс, і контролер (controller) - це область логіки, яка контролює і керує всіма її складовими і даними.

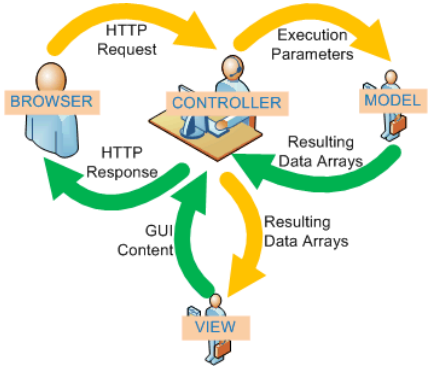

Рисунок 4.2 - Концепція PHP фреймворків

4.2.5 Особливості фреймворку CodeIgniter

CodeIgniter (CI) - популярний MVC PHP-фреймворк з відкритим вихідним кодом, призначений для розробки повноцінних Web-систем і додатків. Даний фреймворк розроблено компанією EllisLab.

Виконання програми засобами CI відбувається через єдиний інтерфейс – програмний файл index.php, який виконує ініціалізацію та застосовує налаштування проекту з конфігураційних файлів, перетворює вхідні дані від користувача на внутрішні структури даних CI та передає управління засобам маршрутизації.

Модуль маршрутизації аналізує вхідні дані HTTP-запиту, а саме: метод HTTP, URL-адресу та заголовки. Якщо сторінка з такими ж вхідними даними вже існує у внутрішньому кеші CI, то її вміст повертається без подальшої обробки безпосередньо користувачеві (через Web-сервер). В іншому випадку наступним кроком є фільтрація вхідних даних з метою уникнення несанкціонованих дій з боку можливого зловмисника: екранування даних, вилучення небажаних тегів тощо. Після цього в залежності від URL запит надходить до визначеного контролера додатку. Контролер, в свою чергу, обмінюється даними з моделями (виконує запити до бази даних, файлів тощо), за необхідності викликає бібліотечні та допоміжні функції і, наприкінці, передає дані у подання, де формується кінцева HTML-сторінка, яка передається користувачеві.

Графічно дану схему виконання додатку з використанням фреймворку CI ілюструє рисунок 4.3, на якому напівжирним шрифтом виділено ті елементи структури проекту, які кодуються програмістом, при цьому решта – вбудовані модулі CI.

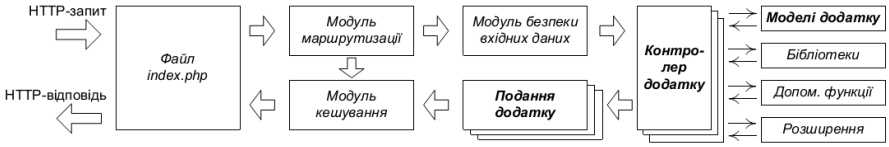

Рисунок 4.3 - Схема виконання програми CodeIgniter

Комплект файлів CI об‘єднується в групи ієрархічно поєднаних каталогів в залежності від функціонального призначення. Головним каталогом Web-додатку є Application, а ядро системи розташоване у System. На рисунку 4.4 представлено файлову структуру проекту з використанням фреймворку CI.

З рисунку 4.4 видно, що різноманітні налаштування проекту групуються в каталозі config, зокрема, налаштування бази даних зберігаються у database.php, маршрутизація URL – у routes.php тощо. Це дозволяє змінювати різноманітні налаштування безпосередньо в процесі функціонування Web-додатку, не змінюючи склад файлової системи та не запускаючи додаткових командних файлів, що часто буває критичним при використанні орендованого дискового простору на сервері для Web-проектів.

Найголовнішими блоками проекту, які реалізовує безпосередньо програміст є:

– контролери (controllers), які є точками входу при виконанні обробки відповідного запиту та виконують функцію управління ходом обробки запиту від користувача.;

– моделі (models), які уніфікують доступ до зовнішніх даних (баз даних, файлів, мережних служб тощо);

– подання (views), які зберігають шаблони Web-сторінок, відокремлюючи дизайн від інформаційного вмісту HTML-документу.

За замовчуванням каталоги controllers та views містять тестові файли welcome.php та welcome_message.php відповідно. Ці файли можна використовувати як зразок для створення власного проекту.

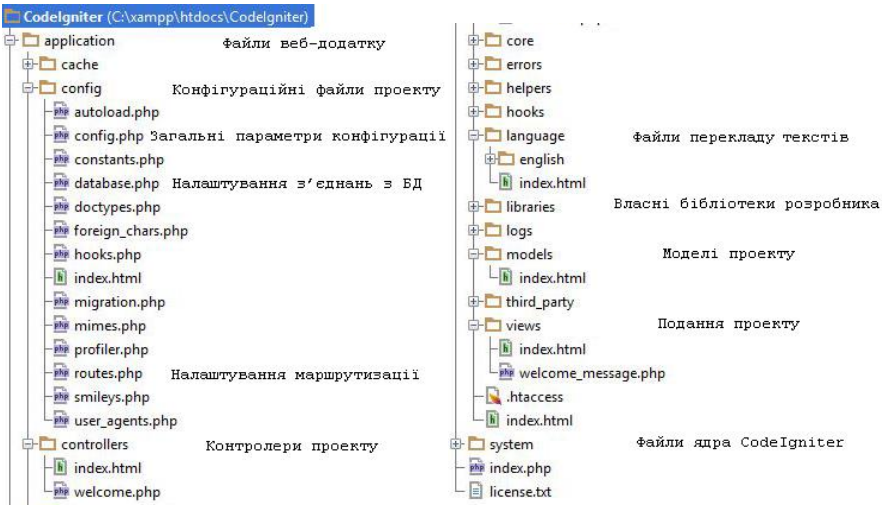

Рисунок 4.4 – Файлова структура проекту CodeIgniter

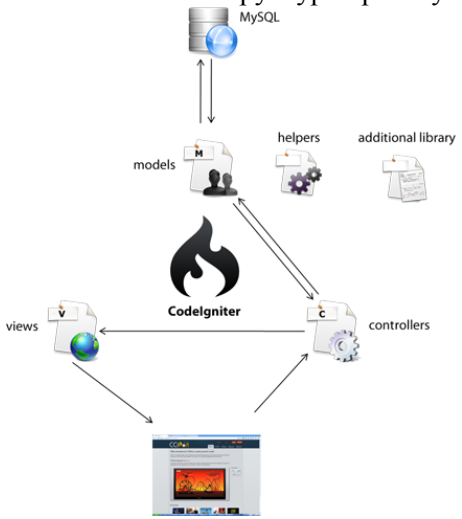

Рисунок 4.5 – MVC структура Web-додатку

Розглянемо детальніше хід виконання програми на прикладі. http://localhost/index.php/.

Алгоритм обробки полягає у наступному. Web-сервер, отримавши запит, передає його на виконання файлу index.php, який знаходиться безпосередньо в htdocs. Оскільки в URL не міститься

жодного уточнення після index.php, то застосовуються налаштування за умовчанням, вказані в конфігураційному файлі routes.php:

```
$route['default_controller'] = 'welcome';
```

Завдяки цьому, управління передається у **контролер** Welcome, що знаходиться в файлі application/controllers/welcome.php (за домовленістю назва файлу і класу має співпадати, але назва класу записується з великої літери). Код файлу welcome.php має вигляд:

```
<?php if ( ! defined('BASEPATH')) exit('No direct script
access allowed');
class Welcome extends CI_Controller {
function __construct(){
parent::__construct();
// Your own constructor code
}

function index(){
$data['page_title'] = 'Welcompe page title';
$this->load->view('welcome_message', $data);
}
}
```

Основу файлу welcome.php складає клас Welcome, успадкований від CI_Controller – базового класу для всіх контролерів проекту. Суттєвим методом в ньому є index(), оскільки саме цей метод викликається за замовчуванням у випадку, коли URL не містить додаткових елементів, крім …/index.php. В цьому методі записано код:

```
$this->load->view('welcome_message', $data);
```

який означає, що необхідно передати управління компоненту подання (view) з назвою файла welcome_message.php з метою генерації сторінки HTML. В подання також передається змінна $data, яка представляє собою масив. В поданні можуть бути використанні змінні, що відповідають ключам масиву (в прикладі це $page_title).

Як правило, файли подання містять код HTML з найпростішими вставками мовою PHP: операторами та функціями виведення (echo, print тощо), а також циклів (for, foreach та іншими). Це обумовлено власне метою подання – об‘єднання статичного коду HTML та динамічних даних, отриманих в результаті взаємодії програми з базою даних, файлами тощо.

Вибір **контролера**, якому передається управління при обробці запиту користувача, визначається URL-адресою запиту. Типова структура URL-адреси має наступний вигляд :

http://example.com/[controller-class]/[controller-method]/[arguments].

URL-адреса має фіксований формат, як показано на рисунку 1.4. Першим параметром є назва контролера проекту, якому передається управління, далі метод у класі контролера, і остання частина – параметри, які отримає метод контролера.

Наприклад:

http://example.com/portfolio/latest/5 - клас контролера portfolio, метод контролера latest, 5 - параметр метода.

Слід зауважити, що на рис.4.5 після назви сайту не вказано index.php. Як правило, на практиці позбуваються використання index.php засобами Web-сервера. Наприклад, для Web-сервера Apache необхідно в кореневому каталозі проекту створити файл .htaccess і вписати в цей файл наступні інструкції:

```
RewriteEngine on
RewriteCond $1 !^(index\.php|images|robots\.txt)
RewriteRule ^(.*)$ /index.php/$1 [L]
```

При цьому завдяки використанню модулю modrewrite встановлюються правила обробки запиту перед передачею його Web-додатку (тут CodeIgniter). А саме: RewriteCond вказує, які файли пропускати без обробки (index.php, images, robots.txt), а RewriteRule задає правило, що будь-який запит, крім вказаного в RewriteCond необхідно модифікувати шляхом додавання перед ним рядку index.php. Таким чином, CodeIgniter розпізнає вірний формат URL.

Моделі призначені для об'єднання функцій взаємодії із зовнішніми даними (файлами, базами даних) у єдиний клас. Для використання моделі необхідно створити файл PHP визначеної структури в каталозі /application/models. При чому назви файлу та класу мають співпадати, а назва класу записується з великої літери. Структура класу включає назву класу, ознаку успадкування від універсальної моделі (CI_Model), а також конструктор _constructor(), в якому викликається код батьківського класу. Завантаження моделі з контролера відбувається за допомогою команди:

```
$this->load->model('Model_name');
```

де Model_name - назва класу моделі, а виклик функції відбувається за допомогою команди $this->model_name->function().

```
<?php
class Testmodel extends CI_Model {
function _construct() {
parent::__construct();
```

```
//завантаження бібліотеки бд
$this->load->database();
}
function get data($id) {
$sql = "SELECT fieldl FROM tablel WHERE id = ".$id;
$q = $this->db->query($sql);
$row = $q->row();
return $row->fieldl;
}}
```

В подальшому виклик методів класу моделі відбувається наступним чином:

```
$this->Testmodel->get_data(10);
```

тобто модель стає об'єктом базового класу CI.

Для автоматичного завантаження моделі при старті проекту необхідно у конфігураційному файлі /application/config/autoload.php вписати назву моделі в відповідний масив:

```
$autoload['model] = array('model_name');
```

Моделі традиційно відповідають за взаємодію із базою даних, тому відповідний клас прикладу містить функцію порівняння даних про користувача із вмістом бази даних. Слід зауважити, що CI виконує під‘єднання автоматично під час завантаження об‘єкта database, тому код класу моделі зазвичай містить лише команди доступу до таблиць бази даних.

Зазначимо, що оскільки CI є об‘єктно-орієнтованою бібліотекою, то результати, що повертаються з бази даних, мають вигляд об‘єктів класу запиту, де члени класу відповідають полям, визначеним у запиті select.

Подання у Codelgniter - це повна HTML-сторінка або її фрагмент (наприклад: заголовок, меню, основний вміст, нижня частина). Основне призначення подання - відділити логіку отримання даних від візуального вигляду. Як правило, подання містить мінімум PHP-коду: лише той, що призначено для виведення даних (echo, print тощо). Крім того, допустимим є використання циклів PHP (for, foreach тощо) для виведення даних, що мають вигляд списків, дерев та інших складних структур.

Крім того, в CI для швидкої розробки додатків передбачені **бібліотеки** та **помічники**. Програміст може розширювати існуючі та створювати власні бібліотеки та помічники.

Помічники та бібліотеки дозволяють вирішувати задачі. Різниця між ними в тому, що кожний файл помічника - це просто об'єднана в

одну категорію колекція функцій, кожна з яких виконує одну специфічну задачу незалежно від інших функцій. Помічники пишуться не в ООП-форматі, а є простими процедурними функціями.

Є помічник URL, що допомагає створювати посилання, помічник Form, який допомагає створювати елементи форми, помічник Text з операціями форматування тексту, помічник Cookie для встановлення та зчитування куків, помічник File для роботи з файлами тощо. Файли помічников завантажуються за допомогою команди $this->load->helper('name');, де name - це імя помічника без розширення .php та постфікса _helper. Після завантаження поміника його функції стають доступні як звичайні функції PHP.

Серед бібліотек слід виділити бібліотеки для роботи з вхідними та вихідними даними додатку, електронною поштою, фтп, створення сесій, засоби двонаправленного шифрування даних, валідації форм, пагінації, бібліотеки для завантаження файлів та для роботи з зображеннями, язикову бібіліотеку для локалізації додатку тощо.

Для їхнього завантаження використовується команда

```
$this->load->library('classname');
```

де classname - це імя класу, який треба завантажити. Для використання бібліотеки використовується команда

```
$this->someclass->some_function();
```

Детальна документація по фреймворку CodeIgniter може бути знайдена у дистрибутиві фреймворку та за посиланнями [10].

4.3 Завдання на лабораторну роботу

4.3.1 Розгорнути та налаштувати PHP-фреймворк CodeIgniter. Підключити його до розробленої база даних.

4.3.2 Ознайомитися зі структурою фреймворку.

4.3.3 Створити простий модуль, що відображає статичні сторінки Web-додатку, та модуль новин сайту. При виконанні завдання бажано не користуватися помічниками фреймворку та шаром абстракції бази даних Active Record.

4.4 Зміст звіту

4.4.1 Тема та мета роботи.

4.4.2 Результати роботи.

4.4.3 Висновки, що відображують результати виконання роботи та їх критичний аналіз.

Лабораторна робота № 5
Проектування архітектури Web-додатку з використанням CodeIgniter

5.1 Мета роботи

Спроектувати структуру Web-додатку з використанням фреймворку CodeIgniter. Розподілити роботу по реалізації додатку серед членів команди.

5.2 Основні теоретичні відомості

Приступаючи до роботи над архітектурою додатку, необхідно пам'ятати про основні принципи проектування. Це допоможе створити архітектуру, яка буде слідувати перевіреним підходам, забезпечить мінімізацію витрати, простоту обслуговування, зручність користування та розширюваність.

Розглянемо ці основні принципи.

Розподіл функцій. Розділіть додаток на окремі компоненти по можливості з мінімальним перекриттям функціональності. Важливим фактором є максимальне зменшення кількості точок дотику, що забезпечить високу зв'язність (high cohesion) і слабку зв'язаність (low coupling). Невірне розмежування функціональності може привести до високої пов'язаності і складнощів взаємодії, навіть незважаючи на слабке перекриття функціональності окремих компонентів.

Принцип одиничності відповідальності. Кожен окремо взятий компонент або модуль повинен відповідати тільки за одну конкретну властивість, функцію або сукупність пов'язаних функцій.

Принцип мінімального знання (також відомий як Закон Деметера (Law of Demeter, LoD)). Компоненту або об'єкту не повинні бути відомі внутрішні деталі інших компонентів або об'єктів.

Не повторюйте (Don't repeat yourself, DRY). Намір повинний бути означений тільки один раз. У застосуванні до проектування програми це означає, що певна функціональність повинна бути реалізована тільки в одному компоненті і не повинна дублюватися ні в одному іншому компоненті.

Мінімізуйте проектування наперед. Проектуйте тільки те, що необхідно. У деяких випадках, коли вартість розробки або витрати в разі невдалого дизайну дуже високі, може знадобитися повне

попереднє проектування і тестування. В інших випадках, особливо при гнучкій розробці, можна уникнути масштабного проектування наперед (big design upfront, BDUF). Якщо вимоги до додатка чітко не визначені, або існує ймовірність зміни дизайну з часом, намагайтеся не витрачати багато сил на проектування завчасно. Цей принцип називають YAGNI («You ain't gonna need it»).

Мета архітектора при проектуванні програми або системи - максимальне спрощення дизайну через його розбиття на функціональні області. Наприклад, користувальницький інтерфейс (user interface, UI), виконання бізнес-процесів або доступ до даних - все це різні функціональні області. Компоненти в кожній з цих областей повинні реалізовувати дану конкретну функціональність і не повинні змішувати в собі код різних функціональних областей. Так в компонентах UI не повинно бути коду прямого доступу до джерела даних; для вилучення даних в них повинні використовуватися або бізнес-компоненти, або компоненти доступу до даних.

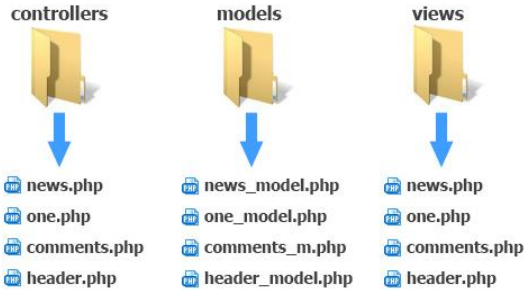

Рисунок 5.1 – Розташування файлів модулів Web-додатку

5.3 Завдання на лабораторну роботу

5.3.1 У складі створених команд розробити проект загальної структури (архітектури) Web-додатку з використанням фреймворку CI. Для цього визначити необхідні функціональні модулі додатку: визначити контролери, моделі та подання, які необхідно створити, їх призначення та необхідні методи.

5.3.2 Структуру додатку представити у вигляді UML діаграм класів.

5.3.3 Розподілити роботу по реалізації додатку серед членів

команди та погодити її з викладачем. Робота кожного студента повинна включати написання методів контролерів та моделей, роботу з поданнями у вигляді створення форм, написання запитів до бази даних, роботу з регулярними виразами, роботу з файлами чи зображеннями, валідацію даних на стороні сервера тощо.

5.4 Зміст звіту

5.4.1 Тема та мета роботи.

5.4.2 Результати роботи.

5.4.3 Висновки, що відображують результати виконання роботи та їх критичний аналіз.

Лабораторна робота № 6
Інтеграція створеного макету до Web-додатку

6.1 Мета роботи

Інтегрувати створений макет Web-сайту до розробленого Web-додатку.

6.2 Основні теоретичні відомості

Документація по фреймворку CodeIgniter пропонує формувати Web-сторінки з незмінними частинами, такими як хедер чи футер, наступним чином:

```
$this->load->view('header');
$this->load->view('template');
$this->load->view('footer');
```

Однак така стратегія не є досить гнучкою та не дозволяє явно відобразити структуру сторінки.

Для того, щоб скористатися можливістю використання шаблону розмітки, пропонуємо скористатися бібліотекою Layout, яку можна знайти за посиланням [13].

Шаблон розмітки описує цілу Web-сторінку як шаблон з блоками для хедеру (шапки сайту), меню, контенту (змісту сторінки), футеру (підвалу сторінки) тощо. Під час візуалізації ці блоки заповнюються даними.

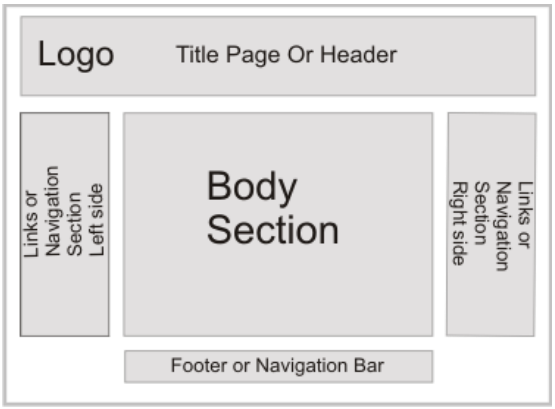

Рисунок 6.1 – Приклад шаблону розмітки

6.3 Завдання на лабораторну роботу

6.3.1 Підключити бібліотеку Layout.

6.3.2 Макет додатку, створений в лабораторній роботі №3, інтегрувати в подання створюваного Web-додатку.

6.4 Зміст звіту

6.4.1 Тема та мета роботи.

6.4.2 Результати роботи.

6.4.3 Висновки, що відображують результати виконання роботи та їх критичний аналіз.

Лабораторна робота № 7
Реалізація функціонального модуля Web-додатку

7.1 Мета роботи

Отримати практичні навички з реалізації функціональних модулів Web-додатків мовою PHP та командної розробки програмних систем.

7.2 Основні теоретичні відомості

Git – це вільна розподілена система керування версіями файлів та спільної роботи. Git є однією з найефективніших, надійних і високопродуктивних систем керування версіями, що надає гнучкі засоби нелінійної розробки, що базуються на відгалуженні і злитті гілок.

Проект створив Лінус Торвальдс для управління розробкою ядра Linux, а сьогодні підтримується Джуніо Хамано.

Git є розподіленою, тобто децентралізованою системою контролю версій. Це означає, що робочий процес як правило виглядає наступним чином.

1. На власному комп'ютері створюється репозиторій (шляхом клонування існуючого репозиторію або створення нового).
2. В робочій директорії цього репозиторію змінюються/додаються/видаляються файли.
3. Виконується фіксація (commit) змін в даний репозиторій (тобто в локальний репозиторій на власному комютері).
4. Етапи 2 и 3 повторюються необхідну кількість разів.
5. За необхідності виконується синхронізація змін з іншими репозиторіями: забираються чужі набори змін (pull) та/або віддаються власні (push).

Таким чином вся повсякденна робота проходить в локальному репозиторії, а коли виникає необхідність виконується відправлення результатів власної роботи в один або декілька інших репозиторіїв,

7.3 Завдання на лабораторну роботу

7.3.1 Встановити та налаштувати систему керування версіями. Створити командою репозиторій за допомогою сервісу Bitbucket та додати до нього файли проекту Web-додатку, отримані в результаті

виконання лабораторної роботи №6.

7.3.2 Виконати роботу, заплановану в лабораторній роботі №5.

7.3.3 Кожен член команд клонує репозиторій та виконує свою частину роботи на власному комп'ютері.

7.3.4 Наприкінці кожного заняття студент повинен виконати фіксацію результатів своєї роботи до центрального репозиторію.

7.3.5 В результаті виконання лабораторної роботи студенти команди повинні продемонструвати робочій варіант власного Web-додатку.

7.4 Зміст звіту

7.4.1 Тема та мета роботи.

7.4.2 Результати роботи.

7.4.3 Висновки, що відображують результати виконання роботи та їх критичний аналіз.

Лабораторна робота № 8
Удосконалення користувацького інтерфейсу Web-додатку за допомогою JavaScript та технології AJAX

8.1 Мета роботи

Удосконалити користувацький інтерфейс розробленого модуля за допомогою JavaScript та технології AJAX.

8.2 Основні теоретичні відомості

JavaScript [9, 12] – мова написання сценаріїв для Web-сторінок. При використанні в рамках технології DHTML JavaScript код включається в HTML-код сторінки і виконується інтерпретатором, вбудованим в браузер.

Ajax (асинхронний Javascript) – це підхід до створення інтерактивних Web-додатків, згідно з яким обмін даними між Web-браузером та Web-сервером відбувається у фоновому режимі, що надає можливості оновлювати лише частину Web-сторінки. Такий підхід зменшує мережний трафік та дозволяє будувати додатки, наближені до «віконних», які є більш звичними для користувача.

Головним чином підхід базується на використанні засобів мови JavaScript. При цьому з серверного боку жодних змін виконувати не потрібно у порівнянні з традиційним підходом. На клієнті за Ajax відповідає об‘єкт Javascript XMLHttpRequest. Передача даних за допомогою цього об‘єкта може відбуватись як у синхронному, так і у асинхронному режимі. Складність безпосереднього використання цього об‘єкта зумовлена неповною сумісністю реалізації різними браузерами. Тому, як правило, використовують крос-платформні бібліотеки, які уніфікують доступ до даного об‘єкта. Однією з таких бібліотек є JQuery. Серед її особливостей можна відзначити широкий спектр можливостей щодо пошуку та модифікації елементів Web-сторінки із застосуванням синтаксису мов XPath та CSS, наявність засобів управління подіями елементів сторінки, візуальні ефекти тощо. З огляду на задачі AjaxJQuery пропонує наступні можливості:

1. Завантаження даних з сервера безпосередньо у елемент сторінки (наприклад, у параграф).

2. Завантаження та виконання програмного коду мовою Javascript.

3. Завантаження даних із сервера у форматі JSON, що спрощує перетворення даних у об‘єкти Javascript.

4. Відправлення даних на сервер.

5. Обробка помилок, що виникли при передачі даних.

6. Серіалізація даних HTML-форми для подальшої передачі на сервер.

8.3 Завдання на лабораторну роботу

8.3.1 Погодити з викладачем завдання на лабораторну роботу, яке містить елементи інтерактивності при взаємодії клієнту з додатком: керування елементами користувацького інтерфейсу сторінки на стороні клієнту, оновлення даних без перезавантаження сторінки, валідацію даних форм на стороні клієнта, побудову графіків тощо.

8.3.2 Підключити до Web-додатку бібліотеку jQuery та інші необхідні бібліотеки, наприклад jQuery UI, HighCharts, Uploadify, DataTables, FullCalendar тощо.

8.3.3 Розробити скрипт на мові JavaScript, що реалізує необхідну функціональність.

8.3.4 Під час зневадження скрипту скористатися вбудованими інспекторами браузерів.

8.3.5 Наприкінці кожного заняття студент повинен внести до центрального репозиторію проекту результати своєї роботи.

8.4 Зміст звіту

8.4.1 Тема та мета роботи.

8.4.2 Результати роботи.

8.4.3 Висновки, що відображують результати виконання роботи та їх критичний аналіз.

Лабораторна робота № 9
Аутентифікація і персоналізація користувачів Web-додатку

9.1 Мета роботи

Отримати практичні навички з аутентифікації і персоналізації користувачів Web-додатку.

9.2 Основні теоретичні відомості

HTTP-cookie, кукі або реп'яшки - поняття, яке використовується для опису інформації у вигляді текстових або бінарних даних, отриманих від Web-сайту на Web-сервері, яка зберігається у клієнта, тобто браузера, а потім відправлена на той же сайт, якщо сайт повторно відвідати.

Таким чином Web-сервер помічає браузер користувача при відвідуванні. Куки створюються за ініціативою скриптового сценарію на стороні Web-браузера. При наступному візиті сервер буде знати, що користувач вже тут був. За допомогою кукі-технології можна вивчити вподобання відвідувача. Кукі є одним із найточніших засобів визначення унікального користувача.

Файл «cookies» (невеликий файл з налаштуваннями профілів) полегшує користування Web-сайтом, записуючи дані, необхідні для входу в систему та збору статистики. Застосовується для збереження даних, специфічних для даного користувача, і використовуваних Web-сервером для різних цілей, серед яких:

- аутентифікація користувача,
- збереження персональних налаштувань користувача,
- відстеження стану сеансу/сесії доступу користувача;
- ведення статистики про користувачів.

Однак у кукі є недолік: їх легко перехопити та підмінити (наприклад, для отримання доступу до облікового запису), якщо користувач використовує нешифроване з'єднання з сервером.

Аутентифікація – це перевірка справжності пред'явленого користувачем ідентифікатора. Аутентифікація зазвичай потрібна при доступі до таких інтернет-сервісів як:

- електронна пошта;
- Web-форум;
- соціальні мережі;

– інтернет-банкінг;
– платіжні системи;
– корпоративні сайти;
– інтернет-магазини.

Позитивним результатом аутентифікації (крім встановлення довірчих відносин і вироблення сесійного ключа) є авторизація користувача, тобто надання йому прав доступу до ресурсів, визначених для виконання його завдань.

9.3 Завдання на лабораторну роботу

9.3.1 Погодити з викладачем завдання на лабораторну роботу, пов'язане з розмежуванням доступу користувачів різних типів до різних частин сайту. Наприклад, це може бути розробка модулю аутентифікації та авторизації для користувачів сайту.

9.3.2 Розподілити роботу по реалізації серед членів команди.

9.3.3 Виконати поставлене завдання.

9.4 Зміст звіту

9.4.1 Тема та мета роботи.

9.4.2 Результати роботи.

9.4.3 Висновки, що відображують результати виконання роботи та їх критичний аналіз.

Лабораторна робота № 10
Генерація персоніфікованих документів засобами PHP

10.1 Мета роботи

Отримати практичні навички з генерації персоніфікованих документів засобами PHP.

10.2 Основні теоретичні відомості

На сайтах, орієнтованих на надання послуг, досить часто необхідно створювати персоніфіковані документи, які генеруються у відповідь на інформацію, що вводиться користувачами. Така функціональність може служити для надання автоматично заповнюваних форм або для генерації таких документів як контракти, листи або сертифікати в форматах RTF чи PDF, а також табличні дані у форматах CSV.

Rich Text Format (RTF, формат збагаченого тексту) - пропрієтарний міжплатформовий формат зберігання розмічених текстових документів, запропонований корпорацією Microsoft і іншими розробниками. Він призначений для використання в якості формату обміну під час передачі документів між різними програмами. Для передачі інформації про форматування в ньому використовується синтаксис та ключові слова і тому він досить легко читається людиною. Таким чином робота з такими файлами досить проста і виконується як зі звичайними текстовими файлами.

Створення PDF-документів трохи складніше. Для цього необхідно скористатися додатковими бібліотеками, як наприклад PDFLib.

CSV (comma-separated values, значення, розділені комою) - файловий формат для представлення табличних даних, у якому поля відокремлюються символом коми та переходу на новий рядок. Поля, що містять коми, декілька рядків, або лапки (позначаються подвійними лапками), мають обмежуватися з обох боків лапками.

Формат CSV використовується для перенесення даних між базами даних та програмами-редакторами електронних таблиць.

10.3 Завдання на лабораторну роботу

10.3.1 Погодити з викладачем завдання на лабораторну роботу.

10.3.2 Розподілити роботу по реалізації серед членів команди.
10.3.3 Виконати поставлене завдання.

10.4 Зміст звіту

10.4.1 Тема та мета роботи.
10.4.2 Результати роботи.
10.4.3 Висновки, що відображують результати виконання роботи та їх критичний аналіз.

Розрахунково-графічне завдання: верстання макету Web-додатку за допомогою HTML та CSS

Мета роботи

Отримати практичні навички з верстання складних макетів Web-додатків за допомогою HTML та CSS з елементами інтерактивності.

Завдання

1. Використовуючи пошукові системи обрати один з сучасних безкоштовних дизайнів Web-сайтів в форматі PSD. Погодити обраний дизайн з викладачем. Дизайн повинен включати 3-4 сторінки.
2. Визначити тип макету, функціональні блоки дизайну, кількість колонок, керуючі елементи, розташування меню. Надати графічне подання структури макету.
3. В графічному редакторі виконати нарізку графічних елементів дизайну.
4. За допомогою мов HTML та CSS розмітити макет сайту та виконати його наповнення графічними елементами та зразками тексту. Під час макетування скористатися сучасними можливостями мови HTML5 та CSS3. Так званий текст-рибу для наповнення сторінок можна взяти за посиланням [14].
5. Перевірити відображення макету в різних браузерах та виправити за необхідності помилки.
6. Для створення інтерактивних можливостей макету підключити бібліотеку jQuery та інші необхідні та створити JS скрипт для керування елементами графічного інтерфейсу (наприклад, випадаюче меню, галереї зображень тощо).
7. Провести валідацію макету.

Зміст звіту

1. Тема та мета роботи.
2. Результати роботи.
3. Висновки, що відображують результати виконання роботи та їх критичний аналіз.

ЛІТЕРАТУРА

1. Халл Элизабет. Разработка и управление требованиями. Практическое руководство пользователя / Элизабет Халл, Кен Джексон, Джереми Дик. – Telelogic, 2005. – 229 с.

2. Casteleyn S., Daniel F., Dolog P., Matera M. Engineering Web Applications. – Berlin: Springer-Verlag, 2009. – 363p.

3. Томсон Лаура. Разработка веб-приложений с помощью PHP и MySQL / Лаура Томсон, Люк Веллинг. – Вильямс, 2010. – 848 с.

4. Колисниченко Д.Н. PHP и MySQL. Разработка веб-приложений. – СПб.: БХВ-Петербург, 2015. – 592 с.

5. Никсон Р. Создаем динамические web-сайты с помощью PHP, MySQL и JavaScript. Спб.: ПИТЕР, 2011. – 496 с.

6. Пауэрс Д. PHP. Создание динамических страниц. – М.: Рид Групп, 2012. – 640 с.

7. Зандстра Мэт. PHP. Объекты, шаблоны и методики программирования / Мэт Зандстра. – Вильямс, 2011. – 560 с.

8. Карпова И П. Базы данных. Учебное пособие. - СПб.: Питер, 2013. - 240 с.

9. Роббинс Дженнифер. HTML5, CSS3 и JavaScript. Исчерпывающее руководство / Дженнифер Роббинс. – 4-е издание. – Эксмо, 2014. – 516 с.

10. David Upton. CodeIgniter for Rapid PHP Application Development. – Packt Publishing, 2007. – 244 p.

11. Дейв Крейн и др. Ajax в действии. – М.: Вильямс, 2006. – 640 с.

12. Бранденбау Д. JavaScript: сборник рецептов, – СПб.: Питер, 2000. – 416 с.

13. Using Layout Pattern With Codeigniter [Електронний ресурс]. – Режим доступу: http://a32.me/2012/09/using-layout-pattern-with-codeigniter.

14. Lorem Ipsum – текст-"риба", що використовується в друкарстві та дизайні [Електронний ресурс]. – Режим доступу: http://www.lipsum.com.