Міністерство освіти і науки України

Національний технічний університет України
„Київський політехнічний інститут імені Ігоря Сікорського”

А.В. Соломін

ВЕБ-ОРІЄНТОВАНА РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПРАКТИКУМ

Рекомендовано Методичною Радою КПІ ім.Ігоря Сікорського як навчальний посібник для студентів, що навчаються за спеціальністю 122 „Комп’ютерні науки та інформаційні технології” для всіх спеціалізацій (протокол №10 від 21.06.2018 р.)

Київ
КПІ ім.Ігоря Сікорського
2018

Рецензенти: Мартиш Є.В., д.ф.-м.н., КНУ ім.Тараса Шевченка
Зубчук В.І., к.т.н., НТУУ „КПІ ім.Ігоря Сікорського”

Відповідальний редактор Антонова-Рафі Ю.В., к.т.н.

Гриф надано Вченою радою ФБМІ НТУУ „КПІ ім.Ігоря Сікорського”
(протокол №10 від 30.05.2018 р.)
за поданням Методичної комісії ФБМІ (протокол №11 від 25.05.2018 р.)

Електронне мережеве навчальне видання

Соломін Андрій Вячеславович, к.ф.-м.н.

ВЕБ-ОРІЄНТОВАНА РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПРАКТИКУМ

Соломін А.В.
ВЕБ-ОРІЄНТОВАНА РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ. ПРАКТИКУМ
[Електронний ресурс] : навчальний посібник для студентів спеціальності 122 „Комп’ютерні науки та інформаційні технології” для всіх спеціалізацій / А.В.Соломін; КПІ ім.Ігоря Сікорського. – Електронні текстові данні (1 файл : 1,27 Мбайт). – Київ : КПІ ім.Ігоря Сікорського, 2018 – 131с.

Посібник призначений для викладачів, що проводять комп’ютерний практикум з дисципліни „Веб-орієнтована розробка програмного забезпечення”, та студентів спеціальності 122 „Комп’ютерні науки та інформаційні технології”. У посібнику розкриваються методичні та технологічні засади створення сучасних веб-застосувань з прикладами рішення типових завдань, що при успішному виконанні і засвоєнні матеріалу сприятиме формуванню вмінь оптимального та ефективного використання сучасних інформаційних технологій при проектуванні компонентів програмного забезпечення, набуванні здатностей до застосування теоретичних знань в практичній діяльності.

За редакцією укладача

ЗМІСТ

ВСТУП

Метою виконання комп'ютерного практикуму з дисципліни «Веб-орієнтована розробка програмного забезпечення» є формування вмінь оптимального і ефективного використання сучасних інформаційних технологій при проектуванні компонентів програмного забезпечення, набування здатностей до застосування теоретичних знань в практичній діяльності.

При цьому важливим є практичне ознайомлення із головними сервісами, їх протоколами, клієнтами, серверами, засобами налаштування, принципами експлуатації та основними технологіями.

У ході виконання практикуму студенти повинні:

- ознайомитись з поширеними службами Інтернету та відповідними до них програмами-клієнтами;
- вміти практично використовувати та налаштовувати програми-клієнти та сервери;
- вміти організовувати обмін інформацією між віддаленими комп'ютерами через різні служби Інтернету;
- навчитись основним засадам створення веб-сайту;
- навчитися проектувати структуру веб-сайту, правильно застосовувати службову та довідкову інформацію, враховувати сучасні вимоги до організації внутрішньої структури веб-сторінок, використовуючи сучасні технології.

Кожна практична робота містить такі розділи:

- теоретичні відомості, необхідні для виконання роботи;
- порядок виконання роботи;
- контрольні запитання.

Структура та оформлення звіту до практикуму

Звіт з комп'ютерного практикуму оформлюється на аркушах формату А4. Звіт складається відповідно до змісту і повинен містити такі розділи:

- титульна сторінка,
- мета роботи,
- опис етапів виконання роботи,
- опис отриманих результатів,
- висновки за результатами роботи.

Організація, контроль виконання та захист практикумів

До виконання комп'ютерного практикуму допускаються студенти, які мають теоретичні знання, що необхідні для виконання цієї роботи, а також знають та дотримуються правил безпеки.

Комп'ютерний практикум захищається безпосередньо після її виконання згідно з графіком, який встановлений робочою програмою курсу. Роботи, які захищені із запізненням, зараховуються зі штрафними балами. При захисті роботи студент демонструє результати виконаної роботи та відповідає на контрольні запитання за темою роботи.

1. ІНСТАЛЯЦІЯ WEB-СЕРВЕРА APACHE НА ОСНОВІ WAMP «ДЕНВЕР»

Мета роботи: Навчитися встановлювати програмне забезпечення, необхідне для роботи веб-сервера на локальному комп'ютері, у т.ч. WEB-сервер Apache, PHP та СКБД MySQL. Створити програмне середовище для подальшого виконання завдань практикумів, пов'язаних із серверними сценаріями.

1.1. Інформація для самостійної підготовки

Інсталяція та налаштування всіх складових веб-сервера, тобто виконання функцій адміністратора веб-вузла вимагає неабиякого досвіду. Але при виконанні багатьох завдань даного курсу, так само і при налаштуванні створюваного сайту в реальних умовах дуже важливо мати локальний веб-сервер. Тому спільнота розробників веб-сайтів створила інструментарії спрощеної інсталяції та автоматичного налаштування локального веб-сервера Apache разом з інтерпретатором PHP та системою керування базами даних MySQL. Такі інструментарії називаються скорочено WAMP, LAMP, MAMP, XAMP, де перша літера означає операційну систему (для Windows це літера W), а інші літери – відповідно Apache, MySQL, PHP. Серед пакетів WAMP, що підтримують кирилицю найбільш поширений пакет «Денвер», що означає „джентльменський набір веб-розробника".

Цей пакет дозволяє запустити повноцінний веб-сервер Apache з підтримкою PHP, Perl і сервер MySQL на локальній машині, що дозволяє веб-розробнику налаштовувати свій сайт без використання віддаленого хосту і лише після налаштування перемістити його. Для цілей даного курсу локальний сервер також необхідний при вивченні серверних технологій,

наприклад, PHP. Завдання наступного практикуму, пов’язаного з CMS Joomla, також неможливо буде виконати без локального веб-сервера.

В порівнянні з установкою повноцінного професійного веб-вузла Apache пакет «Денвер» має ряд переваг:

- Модульність і розширюваність. Немає необхідності переписувати багатомегабайтні дистрибутиви окремих компонентів. Базова версія Денвер-пакету Apache+PHP+Perl+MySQL має розмір усього 2—3 Мб і при цьому він цілком функціональний. У разі потреби можна переписати із сайта розробників додаткові ресурси, бібліотеки, що розширюють можливості PHP, у т.ч. забезпечують роботу з графікою, базами даних, відмінними від MySQL, і т. д. Більшість ресурсів є офіційно безкоштовними.
- Компоненти пакету вже зконфігуровані для роботи. Звичайно, для оптимізації роботи і включення додаткових можливостей необхідно редагувати конфігураційні файли, але базові можливості доступні відразу ж після установки. До складу пакета включений також установник, що значно спрощує процедуру конфігурування при перенесенні пакета в іншу директорію.

Щодо недоліків пакету «Денвер» порівняно з повноцінним професійним веб-вузлом Apache слід відзначити, в першу чергу, майже відсутню захищеність від нападів, тобто теоретично на базі «Денвер» можна створити і реальний веб-вузол, відкривши до нього зовнішній доступ, але практично це небезпечно.

Склад базового комплекту:

1. Apache: виконувані файли, дистрибутивні й адаптовані конфігураційні файли;

2. PHP: виконувані файли, модуль для веб-сервера Apache, дистрибутивний і адаптований конфігураційний файл, бібліотека підтримки графіки;

3. MySQL: виконувані файли, файли повідомлень про помилки на російській і англійській мовах, база даних mysql;

4. панель керування базою даних MySQL — phpMyAdmin, а також скрипт, що спрощує додавання нового користувача MySQL;

5. Perl: виконувані файли;

6. відлагоджувальна «заглушка» для sendmail (/usr/sbin/sendmail), що не відправляє листи, а записує їх у файл /tmp/!sendmail.txt;

7. система автоматичного пошуку віртуальних хостів і відновлення системного файлу hosts, а також конфігурації Apache. Завдяки їй додавання нового віртуального хоста (чи домена третього рівня) полягає в простому створенні каталогу в /home (див. за аналогією з вже існуючими хостами) і перезапуску комплексу. Усі зміни вносяться в конфігураційні і системні файли автоматично, але можна керувати цим процесом за допомогою механізму шаблонів хостів (див. /usr/local/apache/conf/httpd.conf за детальними роз'ясненнями).

На офіційному сайті доступні доповнення, що розширюють можливості базового комплекту.

Установка і налаштування.

Усі дистрибутиви поставляються у вигляді архівів, що саморозпаковуються, які містять також і інсталятор, що автоматично запускається. Процес установки полягає у відповіді на низку запитань у діалогових вікнах.

1. На перше запитання („Чи дійсно Ви бажаєте установити базовий комплект?") відповідайте позитивно — Так.

2. Відкриється вікно програми браузера з короткою інформацією про «Денвер». Для продовження установки закрийте його.

3. Подальша установка вестиметься у вікні консолі. Тут видаються інструкції по подальшій інсталяції і запрошується необхідна інформація.

4. В процесі установки потрібно відповісти на два питання: куди установити пакет і яку букву віртуального диска використовувати. Віртуальний диск з окремою буквою – це дуже зручно. Це окремий диск серед інших дисків, який цілком знаходиться у розпорядженні вашого сайту. Крім того в цьому випадку можна копіювати повністю веб-сервер на інший носій, наприклад, на флеш і навіть запускати сервер не на жорсткому диску, а на флеш. Можна на ньому створювати і сайти, а це додаткові переваги при виконанні завдань даного курсу і здачі результатів викладачу, враховуючи переміщення з домашнього комп'ютера на комп'ютери навчального закладу.

5. Тепер почнеться копіювання файлів дистрибутиву, після закінчення якого потрібно вказати спосіб запуску «Денвера»: чи буде він запускатися при завантаженні операційної системи або ж лише в ручному режимі.

6. Після закінчення процесу інсталяції сервера аби перевірити, чи він працює, слід перейти за адресою http://localhost/. Ви повинні потрапити на тестову сторінку системи. Вибравши ті чи інші посилання, можна протестувати кожен компонент окремо.

Якщо вікно з написом „Ура. Заработало!" не з'явилося, то «Денвер» працює неправильно. Майте на увазі, що він не функціонує спільно з програмою Skype. Аби працювати з цими програмами одночасно, необхідно відключити в Skype використання портів 80 і 443. Крім того, «Денвер» не запрацює, якщо у вас запущений інший веб-сервер, наприклад IIS (Microsoft) або увімкнений у налаштуваннях проксі-сервер.

У результаті, після установки новостворений диск Z міститеме 4 папки: denwer, home, tmp, usr (якщо при обиранні літери назви диску було обрано літеру існуючого диску, наприклад C:, то ці папки будуть розкидані серед інших вже існуючих на цьому диску; при цьому мобільність сервера буде збережена, тобто можна скопіювати ці папки на інший диск, наприклад, флеш і запустити веб-сервер вже на цьому диску). В папці denwer важливим є файл налаштування CONFIGURATION.txt, а також файли запуску веб-сервера (Run.exe), зупинки (Stop.exe) та перезапуску (Restart.exe). Папка home є кореневою для всіх сайтів. В папці tmp зберігаються тимчасові файли, а в папці usr – програмні модулі складових частин веб-вузла.

Створення віртуальних хостів.

Створити в папці /home директорію з ім’ям, що збігається з ім’ям віртуального хоста (наприклад test.ua).

Ця директорія буде зберігати директорії документів доменів третього рівня для test.ua. Наприклад, ім’я abc.test.ua зв’язується сервером з директорією /home/test.ua/abc/, а ім’я abc.def.test.ua — з /home/test.ua/abc.def/. Ну і, звичайно, піддиректорія www відповідає адресам www.test.ua і просто test.ua, оскільки фрагмент доменного імені www. вважається „за замовченням”. Треба створити папку www у директорії віртуального хоста, адже саме в ній будуть зберігатися його сторінки і скрипти. Тобто HTML-документи повинні знаходитися в директоріях /home/<ім’я_хоста>/www.

Після зміни структури файлів в папці home обов’язково треба перезапускати веб-сервер за допомогою командного файлу Restart.exe, щоб сервер „підхопив” нову структуру.

1.2. Порядок виконання роботи:

Устаткування, матеріали та інструменти

- Персональний комп'ютер.
- Інсталяційний пакет «Денвер».

Порядок виконання роботи

- Отримати у викладача або скачати з відкритих джерел інсталяційний пакет «Денвер».
- Встановити комплекс програм на комп'ютері.
- Створити хост для подальшої роботи.

За результатами виконання роботи в звіті повинні бути представлені:

1. Назва роботи
2. Мета роботи
3. Скрін-шоти сторінок створених сайтів.
4. Висновки

1.3. Питання до самоконтролю

1.3.1. Чому пакет називається „Денвер" і що таке WAMP?

1.3.2. У чому полягають переваги використання „Денвер" в порівнянні з повноцінною інсталяцією сервера Apach?

1.3.3. Які складові пакета „Денвер"?

1.3.4. В якому файлі пакету „Денвер" містяться налаштування сервера?

1.3.5. В якому файлі пакету „Денвер" містяться налаштування PHP?

1.3.6. Яка послідовність створення локального хоста?

1.3.7. Як створювати в „Денвер" піддомени 2-го та 3-го рівнів?

2. ІНСТАЛЯЦІЯ ТА БАЗОВІ ПРИНЦИПИ ВИКОРИСТАННЯ CMS JOOMLA

Мета роботи: Навчитися використовувати системи управління контентом (CMS), інсталювати CMS Joomla та створювати прості сайти на базі CMS.

2.1. Інформація для самостійної підготовки

CMS (англ. Content management system, система управління контентом) за самим змістом назви призначена, головним чином, для розділення функцій програмування сайту та наповнення його змістом (контентом). Це спрощує створення веб-ресурсів типу он-лайн газети, інформаційні портали, корпоративні сайти та т.ін., коли журналісти, наприклад, або співробітники корпорації не повинні вміти програмувати при наповненні сайту інформацією (контентом). Крім головного призначення CMS значно спрощують створення та підтримку таких сайтів та використання складних веб-технологій.

До нині було створено багато CMS. В даному практикумі детально розглядається одна з них, а саме CMS Joomla!

Система є кросплатформеною, тобто вона однаково працює під управлінням Windows, Linux, Mac OS, Free BSD і інших операційних систем. Це досягається за рахунок того, що при створенні Joomla! використовувалася лише мова PHP, яка, у свою чергу, функціонує на багатьох платформах. Тому точніше буде сказати, що Joomla! працює на будь-якому веб-сервері, який підтримує PHP.

Система безкоштовна.

Дозволяє створювати практично необмежену кількість сторінок сайту. Обмеження більше обумовлюються вживаними на сервері файловими системами і обмеженнями MySQL.

Дає можливість легко налаштовувати і змінювати зовнішній вигляд сайту за допомогою шаблонів. При цьому міняється лише зовнішнє оформлення сайту, а не його внутрішня структура.

За рахунок використання модулів можна істотно розширити функціональність сторінок сайту, наприклад, поставити лічильник відвідин, завантажувати і відображувати стрічки новин, галереї фото, відео.

Вживання компонентів дозволяє створити власні форуми, інтернет-магазини, фотоальбоми, опитування і багато іншого.

Можна підготувати для сайту матеріал і вказати певний час його опубликования.

Адміністратор може легко блокувати і розблоковувати будь-які сторінки, стежачи таким чином за вмістом сайту.

Для кожної динамічної сторінки можна задати список ключових слів, що дозволить пошуковим системам включати ці сторінки в рейтинги.

Система Joomla! складається з двох великих частин.

Адміністративна частина — частина сайту, доступна адміністраторові (в середовищі програмістів для цієї частини поширена назва back-end).

Сайт, або призначена для користувача частина, — частина сайту, яку бачить будь-який користувач, що увійшов на сайт (front-end).

Установка.

Для установки системи на сервері мають бути:

встановлений і запущений PHP версії не нижче 4.3.10;

встановлений і запущений MySQL;

включена підтримка XML;

прописана підтримка бібліотеки архівування zlib;

доступний для читання і запису каталог, в який розпаковуватиметься дистрибутив.

Зазначимо, що всі вищезгадані ресурси автоматично встановлюються за допомогою засвоєного на попередньому занятті пакета Денвер, якщо це стосується локального сервера. Саме такий варіант стосується даного практичного заняття. Якщо ж CMS треба буде розгортати на реальному віддаленому хості, то необхідно переконатись в присутності цих ресурсів на сервері.

Розглянемо установку системи покроково.

1. Створіть в кореневому каталозі сайту підкаталог joomla.

2. Перепишіть в нього архів Joomla_2.5.6-Stable-Full_Package_Russian_v3.tar.gz (або більш нову версію, завантажену з офіційного сайту Joomla).

3. Розпакуйте вміст файлу в каталог joomla. Після цього архів Joomla можна видалити. Для розпаковування архіву можна скористатися архіватором 7-Zip в Windows або tar в UNIX-подобных операційних системах.

4. Відкрийте будь-який браузер і наберіть в ньому адресу http://localhost/joomla — в результате відкриється сторінка установки Joomla! Надалі вважатимемо, що ця адреса є адресою нашого сайту.

Перший крок при установці системи — вибір мови. Вкажіть російську мову і нажмите кнопку Далі. Зверніть увагу, що тут ви вибрали лише мову установки. Русифікація або налаштування для інших мов системи Joomla! реалізується окремим додатком, але в даному випадку версія CMS вже русифікована.

5. Переконайтеся, що конфігурація вашої системи задовольняє мінімальним вимогам Joomla!

У розділі Перевірка Joomla! всі пункти мають бути помічені зеленим знаком.

Якщо в розділі Перевірка Joomla! всі пункти відмічені зеленим, можна переходити до наступного кроку установки, натиснувши кнопку Далі.

Якщо якісь параметри відмічені Немає, необхідно припинити установку системи, усунути проблеми і після цього натискувати кнопку Повторити перевірку.

6. У наступному розділі потрібно підтвердити згоду з ліцензією GNU/GPL.

7. Необхідно налаштувати сервер баз даних, який використовуватиме Joomla! у своїй роботі. Як зазначалося раніше, Joomla! зберігає свої дані на сервері баз даних. У ролі сервера БД використовується MySQL.

На сторінці налаштувань доступу до MySQL є наступні розділи.

Тип бази даних — встановіть MySQL.

Назва хосту — вкажіть IP-адресу або назву сервера, на якому запущено MySQL (той, до якого ви маєте доступ). В нашому випадку – localhost.

Ім'я користувача і Пароль — вкажіть ім'я користувача і пароль, які дозволять створювати бази, таблиці в них і мати доступ до записів в останніх.

Розширені установки — цим розділом варто скористатися, якщо у вас вже є база даних в MySQL з тим ім'ям, яке ви хочете задати. В цьому випадку клацніть на назві даного розділу — в результаті відкриються додаткові параметри налаштування:

- видалити існуючі таблиці — якщо при установці нової версії системи імена існуючих таблиць в базі даних зпівпадуть з іменами новостворюваних таблиць, старі таблиці будуть видалені;

- створити резервну копію старих таблиць — цей параметр аналогічний попередньому, але в цьому випадку існуючі таблиці не видалятимуться, а перейменуються з префіксом old_; якщо ж в базі вже є резервні копії таблиці, резервні копії замінюватимуться на нові;

- префікс таблиць — префікс системних таблиць Joomla! Цей префікс потрібний для того, щоб гарантувати неспівпадання імен системних таблиць Joomla! з уже існуючими робочими таблицями в базі даних. У новостворюваній системі це великого значення не має, проте щоб уникнути проблем в майбутньому краще задати префікс. Префікс генерується за випадковими правилами.

Після вибору і перевірки налаштувань MySQL на цій сторінці натискуйте кнопку Далі.

8. Необхідно налаштувати параметри доступу по FTP.

На UNIX-подібних системах (Linux, Free BSD і ін.) потрібно вказати ім'я користувача, пароль і кореневий каталог, в якому знаходиться проект Joomla! Це необхідно для того, щоб Joomla! могла дістати доступ до своїх конфігураційних файлів і каталогів. Дане налаштування не суттєве, якщо ви інсталюєте Joomla! на сервер під управлінням ОС Windows.

Після налаштування конфігурації FTP натискуйте кнопку Далі.

9. На наступній сторінці потрібно налаштувати головну конфігурацію.

Необхідно задати назву свого сайту. Обов'язково вкажіть електронну адресу адміністратора сайту в рядку Ваш e-mail групи E-mail і пароль адміністратора аби Joomla! і користувачі сайту могли повідомляти про всі проблеми роботи сайту по електронній пошті. Нарешті, встановіть пароль доступу адміністратора до системи управління сайтом.

Наступний розділ — Завантаження демо-даних/Міграция і відновлення з резервной копії. Цей розділ призначається для того, щоб користувач зміг обрати вигляд установки. Оскільки ми створюємо

тестовий учбовий сайт, натискуйте кнопку Встановити демо-дані. Зверніть увагу, що напис на кнопці зміниться на текст Демо-дані успішно встановлені.

При установці системи для реальної роботи цю кнопку натискати не слід.

Після того, як в цьому вікні налаштовані всі параметри, натискайте кнопку Далі у верхній частині сторінки.

10. На сторінці Завершення установки ви отримаєте повідомлення, що установка завершена і можна перейти або до адміністрування сайту (натискайте кнопку Адмін), або безпосередньо на сайт (натискайте кнопку Сайт).

Перш ніж приступити до роботи в Joomla!, необхідно видалити каталог installation в каталозі, в який був розпакований її дистрибутив. Це слід зробити обов'язково, оскільки в каталозі installation після установки залишається інформація, яка може бути доступна для потенційних зломщиків сайту. У подальшій роботі з сайтом цей каталог не потрібний.

На цьому практичному занятті для демонстрації успішного виконання треба показати свій варіант сайту, створеного на базі демо-сайту, який було встановлено при інсталяції, з іншим контентом, заголовками, ілюстраціями, пунктами меню. Тому слід розібратися з принципами керування, розглядаючи основні елементи демо-сайту, і зробити свої індивідуальні зміни до нього.

2.2. Порядок виконання роботи:

Устаткування, матеріали та інструменти

- Персональний комп'ютер.
- Інсталяційні пакети «Денвер» та «Joomla!».

Порядок виконання роботи

- Отримати у викладача або скачати з відкритих джерел Інсталяційні пакети «Денвер» та «Joomla!».
- Встановити комплекс програм на комп'ютері.
- Створити змінений варіант демо-сайту з іншим контентом, заголовками, ілюстраціями, пунктами меню.

За результатами виконання роботи в звіті повинні бути представлені:

1. Назва роботи
2. Мета роботи
3. Скрін-шоти сторінок створених сайтів.
4. Висновки

2.3. Питання до самоконтролю

2.3.1. У чому сутність та переваги використання CMS?

2.3.2. Які Ви знаєте найпоширеніші CMS?

2.3.3. Які кроки треба виконати для інсталяції CMS Joomla?

2.3.4. Які особливості налаштування СУБД при інсталяції Joomla?

2.3.5. Навіщо при інсталяції Joomla доцільно іноді встановлювати демо-дані та як це робити?

2.3.6. Які можливості CMS Joomla Вас зацікавили?

3. ВИКОРИСТАННЯ РОЗШИРЕНИХ МОЖЛИВОСТЕЙ CMS JOOMLA ДЛЯ СТВОРЕННЯ СКЛАДНИХ САЙТІВ

Мета роботи: Навчитися використовувати розширені можливості системи управління контентом (CMS Joomla) для створення складних сайтів.

3.1. Інформація для самостійної підготовки

Адміністрування в CMS Joomla!

Аби дістати доступ до адміністрування сайту (серед програмістів розділ сайту називається back-end на відміну від розділу front-end, котрий доступний користувачам), потрібно ввести в адресний рядок браузеру назви головного каталога сайту і підкаталогу administrator до нього. Наприклад, в нашому випадку головним каталогом сайту є http://localhost/joomla, тому адресою адміністративної частини сайту буде http://localhost/joomla/administrator. Після введення цієї адреси відкриється сторінка реєстрації користувача-адміністратора в адміністративній частині сайту. Для входу в режим адміністрування сайту необхідно ввести ім'я користувача і пароль доступу адміністратора.

1. У полі Username вкажіть admin (це ім'я користувача для адміністратора за умовчанням).

2. У полі Password введіть пароль, який Ви вказували при установці Joomla! на сторінці Головна конфігурація.

3. У полі Language можна обрати мову інтерфейсу адміністративної частини сайту. За умовчанням встановлена англійська мова.

Тепер натискуйте кнопку Login — в результаті з'явиться головна сторінка адміністрування системи.

Головна сторінка адміністрування розділяється на декілька частин: головне меню (у верхній частині сторінки), поле вибору найчастіше

виконуваних операцій і інформаційна панель (праворуч). На головну сторінку завжди можна попасти, вибравши команду меню Site Control Panel.

Користувачі і групи

Як і в будь-якій системі, що вимагає розмежування доступу, в Joomla! є можливість призначити користувачам і групам різні права. Для надання доступу користувачам до сторінки управління виконайте наступні дії.

1. Увійдіть до адміністративної частини сайту і зареєструйтеся в ній з правами адміністратора, як описувалося вище.

2. Виберіть в меню Сайт Пользователи або на панелі управління значок Пользователи — в результаті відкриється сторінка менеджера користувачів.

Розглянемо групи, які надає CMS Joomla!

Super Administrator — група, користувачі якої мають доступ до будь-якої частини сайту і до всіх налаштувань. Користувачів даної групи не можна видалити або перемістити в іншу групу. При установці Joomla! створюється користувач admin, що належить цій групі. Будьте уважні, розміщуючи в цю групу користувачів. Краще обмежитися одним користувачем admin.

Administrator — група, користувачі якої мають дещо урізані права в порівнянні з користувачами групи Super Administrator. Зокрема, вони не можуть:

змінювати глобальні налаштування сайту в меню „Сайт Общие настройки”;

проводити масові розсилки по електронній пошті;

редагувати і додавати користувачів групи Super Administrator;

додавати і змінювати шаблони сайту;

додавати і змінювати мовні налаштування.

Manager — група, користувачі якої можуть лише управляти вмістом сайту. Їм забороняється:

встановлювати модулі і компоненти;

адмініструвати користувачів;

користувачі групи Super Administrator можуть заборонити їм доступ до деяких компонентів.

Registered — група, користувачі якої можуть заходити на сайт з головної сторінки сайту, дістаючи доступ до якої-небудь додаткової інформації, вказаної адміністратором.

Author (Автор) — група, користувачі якої успадковують права групи Registered, але додатково до неї вони можуть змінювати інформацію, додану ними ж. Наприклад, користувач може помістити на сайт свою статтю і редагувати її надалі, при цьому інші користувачі даної групи цього зробити не зможуть.

Editor — група, користувачі якої можуть додавати і змінювати інформацію, створену будь-яким користувачем (у цю групу включаються редактори сайту).

Publisher — група, користувачі якої можуть робити те ж, що і користувачі групи Editor, проте вони можуть також розміщувати на сайті будь-яку інформацію.

На сайті можна розмістити форму реєстрації нових користувачів. При цьому кожен новий користувач автоматично потрапляє в групу Registered. Аби увімкнути або вимкнути реєстрацію користувачів, виконайте наступні дії.

1. Увійдіть до режиму адміністрування сайту з правами користувача групи Super Administrator.

2. Виберіть меню „Сайт Общие настройки”, за умовчанням повинна відкритися вкладка Сайт.

3. Перейдіть на вкладку Система.

4. В області Налаштування користувача встановіть перемикач проти поля „Дозволити реєстрацію користувачів” в положення Так. Параметр „Права нового користувача” дозволяє встановити, до якої групи будуть автоматично зараховуватись нові користувачі.

Часто буває, що зареєстрованого користувача потрібно активувати. Наприклад, при створенні електронної пошти необхідно, аби кожен новий користувач активувався. Це робиться для того, щоб запобігти разповсюдженню спаму на ваш сайт. В той же час, наприклад, інтернет-магазину може і не потрібно активація користувача для доступу до корзини замовлень. Тому в системі Joomla! передбачається можливість вмикати і вимикати активацію користувачів, що реєструються. За це відповідає параметр Активація нового користувача. Якщо перемикач встановити в положення Так, то кожен новий користувач повинен активуватися. Ця процедура полягає в тому, що після введення своїх параметрів (імені користувача, логіна, пароля і електронної адреси) користувачеві висилається по электронной пошті лист із записом для активації свого облікового запису. Йому потрібно виконати необхідні дії. Якщо перемикач встановити в положення Ні, то будь-який користувач, що реєструється, відразу дістає доступ до додаткової інформації сайту без активації.

5. Після змін облікових записів користувачів збережіть їх, натискаючи кнопку Застосувати або Зберегти. Для відміни змін і виходу з режиму зміни параметрів на панелі інструментів вкладки Система натискуйте кнопку Закрити.

Компоненти

Компонент — це підпрограма, яка формує динамічну сторінку сайту. Про те, який компонент треба показати на сторінці, система Joomla! узнає з адресного рядка.

Наприклад, рядок http://localhost/joomla/index.php?option=com_sample повідомляє систему, що необхідно вивести на сторінку компонент com_sample.

У CMS Joomla! компоненти служать для організації основної частини сайту. Вони можуть організовувати форуми, інтернет-магазини, новинні портали, форми зв'язку з адміністрацією сайту і багато що інше. На сторінках сайту компоненти розташовуються на їх центральній частині і грають головну роль. Аби компонент попав на сторінку сайту, його потрібно встановити, прив'язати до меню і опублікувати.

Для установки нових компонентів виконайте наступні дії.

1. Відкрийте меню Розширення Установить/Удалить — в результаті з'явиться сторінка Менеджер розширень.

2. Якщо компонент поставляється у вигляді zip-архива, натискуйте кнопку Огляд в області Завантажити файл пакету і виберіть файл, в якому знаходиться компонент.

3. Натискайте кнопку Завантажити файл & Встановити — у результаті компонент завантажиться на сайт, розпакується і встановиться в систему.

4. Якщо компонент поставляється у вигляді набору файлів, що знаходяться в каталозі, введіть повний шлях (локальний на вашому комп'ютері) до каталога з компонентом в області Встановити з теки і натисніть кнопку Встановити.

5. Якщо ви бажаєте встановити компонент з Інтернету, скористайтеся областю Встановити з URL. У полі URL введіть повну адресу, звідки система викачає і встановить компонент, і натискайте кнопку Встановити.

Далі з'явиться повідомлення про те, що компонент успішно встановився. Тепер дістати доступ до адміністрування компонента можна через меню „Компоненти Имя компонента”. Якщо встановити компонент не удалося, система оповістить вас про це.

На сторінці видалення компонентів можна також вмикати і вимикати компоненти, натискаючи, відповідно, на зелену галочку або на червоний хрестик.

Модулі

Модуль — це підпрограма, написана на PHP або JavaScript. Візуально вона виглядає прямокутником, який може розміщуватися практично в любой частині сайту, — там, де забажає адміністратор або автор сторінки сайту. Модулі дозволяють показувати баннерную рекламу, анонси новин, організовують форми входу користувачів на сайт, в пошту і багато іншого. Модуль сам по собі існувати не може: він завжди прив'язується або до меню, або до сторінки сайту.

Ці підпрограми бувають двох типів: модулі сайту і модулі адміністративної частини сайту. Модулі сайту публікуються на сторінках сайту, до них мають доступ користувачі сайту. Адміністративні модулі призначаються для розширення функціональності адміністративної частини сайту. До модулів можна відкривати і закривати доступ так само, як і до компонентів. Можна дозволити доступ до модулів окремим групам користувачів. Налаштування модулів зберігаються, як правило, в xml-файлах, які є складовими модуля. На одній сторінці може знаходитися декілька модулів.

Установка модулів ідентична установці компонентів.

Адміністрування модулів відбувається на сторінці менеджера модулів аналогічно адмініструванню компонентів.

Для налаштування параметрів модуля клацніть в стовпці Назва модуля по імені модуля — в результаті відкриється сторінка налаштування параметрів модуля.

Шаблони

Шаблон — це набір стильових таблиць (CSS), малюнків і html-сторінок, в якому визначаються виклики php-функцій, що змінюють загальний вигляд сайту. Оформлення сайту, а саме: колірне рішення, фонові малюнки, положення меню, деяких елементів управління — визначається саме в шаблоні. Головна функція шаблонів — управління зовнішнім виглядом сайту без зміни вихідних кодів компонентів, модулів і плагінів, тобто розділення програмної і інтерфейсної частин сайту. Після інсталяції Joomla! у системі за умовчанням вбудовуються кілька шаблонів, один із яких „beez". Установка шаблонів в систему відбувається аналогічно установці компонентів, а адміністрування – за посередництвом менеджера шаблонів, що знаходиться в меню „Розширення".

Плагіни.

Плагин (мамбот, або фільтр) — це підпрограма, яка на вході отримує код сторінки, обробляє його і на виході видає інший код. Це дуже зручний інструмент, за допомогою якого можна обробити будь-яку сторінку перед показом її користувачеві. Наприклад, можна організувати автоматичний переклад сайту іншою мовою, вилови і видалення нецензурних виразів на сторінках (зокрема, на форумах).

Установка плагінів нічим не відрізняється від установки компонентів, а управління плагінами аналогічне управлінню модулями.

Отже, CMS Joomla! пропонує широкий спектр розширень для завдань вашого сайту, що дозволяє гнучко і дуже швидко набудовувати його практично під будь-які потреби. Чітка модульна структура дає можливість істотно змінювати зовнішній вигляд сайту.

Як приклад, наведемо покроково процес інсталяції плагіна Simple Image Gallery, що слугує для створення галерей зображень в спливаючому вікні на сайті. Його перевагою є те, що галереї можна показувати на будь-якому місці усередині статті, і він дуже простий у використанні.

Установка і налаштування Simple Image Gallery на Joomla 2.5.

КРОК 1. Установка проводиться через стандартний менеджер розширень Joomla так само, як і інших розширень.

Крок 2. Включити Simple Image Gallery - «Розширення» - «Менеджер плагінів», налаштування – тут вказана тека, де зберігаються зображення, розміри ікон, параметри відображення, можна також вказати назву зображень.

КРОК 3. Створити теку зі своїми зображеннями в теці image.

КРОК 4. Вказати на будь-якому місці статті, де ви хочете розмістити галерею ось такий вираз, де Paris – назва вашого альбому:

{ gallery}Paris{/gallery } без пропусків!

3.2. Порядок виконання роботи:

Устаткування, матеріали та інструменти

- Персональний комп'ютер.
- Інсталяційні пакети «Денвер» та «Joomla!».

Порядок виконання роботи

- Установити на своєму комп'ютері локальний сервер Денвер.
- Установити CMS Joomla на локальному хості.
- Розробити в CMS Joomla сайт з такими параметрами:
    a. шаблон сайту повинен відрізнятися від шаблону Joomla, тобто в шапці сайту повинні бути інший фоновий малюнок, лого та назва сайту;
    b. видалити рекламні матеріали в нижній частині сайту Joomla;
    c. створити кілька сторінок з текстом і картинками;

d. використати категорії матеріалів для структурування контенту, наприклад, категорія вищого рівня – „захворювання”, категорія нижчого рівня – „серцеві захворювання”; повинні бути сторінки з матеріалами відповідних категорій;
e. форма зворотнього зв’язку для можливості користувачу надсилати листа до системного адміністратора сайту;
f. присвоєння відповідних повноважень доступу користувачам різних груп;
g. десь на сторінці використати галерею фотографій; посилання на якесь відео з YouTube;
h. розмістити на сайті якусь стрічку новин RSS.

Підказки:

Формат галереї фото та посилання на YouTube:

{gallery}kpi{/gallery}

{youtube}1EoUneNYCIQ{/youtube}

(тут kpi – це папка з фото, яка повинна бути створена у медіа-менеджері в Joomla;

1EoUneNYCIQ – ідентифікаційний номер якогось відео з YouTube)

За результатами виконання роботи в звіті повинні бути представлені:

1. Назва роботи
2. Мета роботи
3. Скрін-шоти сторінок створених сайтів.
4. Висновки

3.3. Питання до самоконтролю

3.3.1. Що таке компоненти, модулі, плагіни в CMS Joomla?

3.3.2. Які Ви знаєте найпоширеніші компоненти, модулі, плагіни в CMS Joomla?

3.3.3. Що таке шаблон CMS Joomla, як ним користуватись?

3.3.4. Як здійснюється розширення функціональних можливостей CMS Joomla?

3.3.5. Чи можна створити сайт віртуального магазину на базі CMS Joomla? А соціальну мережу?

3.3.6. Як створити галерею зображень в CMS Joomla?

4. СТВОРЕННЯ СТРУКТУРИ ОБ'ЄКТІВ СТОРІНОК САЙТУ ЗАСОБАМИ HTML

Мета роботи: Вдосконалити вміння використовувати технології гіпертекстової розмітки в дизайні сайту.

4.1. Інформація для самостійної підготовки

Базові поняття HTML зараз вивчають уже в школі. З використанням виключно тільки мови HTML зараз сайти не розробляють. Але мова гіпертекстової розмітки відіграє дуже важливу роль і в сучасних веб-технологіях, зокрема при створенні DOM (document object model) веб-сторінок, в дизайні сайту, наприклад, при розробці шаблонів Joomla.

Тому на даному практичному занятті зосередимось на кількох складних темах, пов'язаних з використанням HTML, маючи на увазі загалом володіння слухачами базовими технологіями гіпертекстової розмітки. У випадку виникнення потреби детальних пояснень щодо якихось елементів технологій HTML рекомендується звернутись до відповідних літературних джерел, наприклад [1].

4.2. Порядок виконання роботи:

Устаткування, матеріали та інструменти

- Персональний комп'ютер зі встановленим браузером та текстовим редактором типу Блокнот або WordPad.
- Кілька файлів малюнків в форматі .jpg.

Порядок виконання роботи

Завдання практикуму складається з 4-х частин, що пов'язані з різними темами застосування HTML.

1 завдання.

Треба створити WEB-сторінку з складним списком, що повинен виглядати в браузері таким чином

1. Коржі торту
    - Мука
    - Яйця
    - Молоко
2. Крем
    - Масло
    - Згущене молоко
3. Глазур зверху

2 завдання.

Треба створити дві WEB-сторінки. На першій з них повинен бути відображений малюнок, який є одночасно гіперпосиланням на другу WEB-сторінку, причому при наведенні миші на малюнок повиннен з’являтись спливаючий напис „Ви перейдете на іншу сторінку” (як зображено на рис. 4.1):

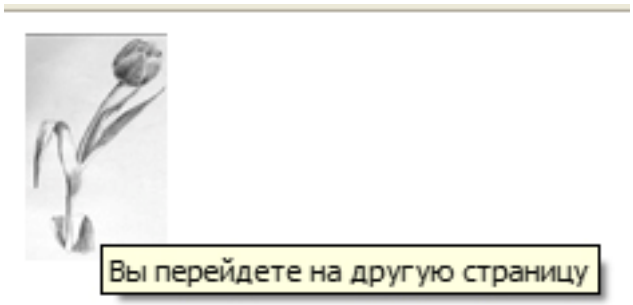

Рис. 4.1. Приклад малюнку, що слугує гіперпосиланням

А при натисканні клавіші миші повинен відбуватись перехід на другу нашу WEB-сторінку ” (як зображено, наприклад, на рис. 4.2):

Рис. 4.2. Приклад переходу за гіперпосиланням на іншу сторінку

<u>3 завдання</u>.

Треба створити WEB-сторінку з таблицею, як зображено на рис. 4.3:

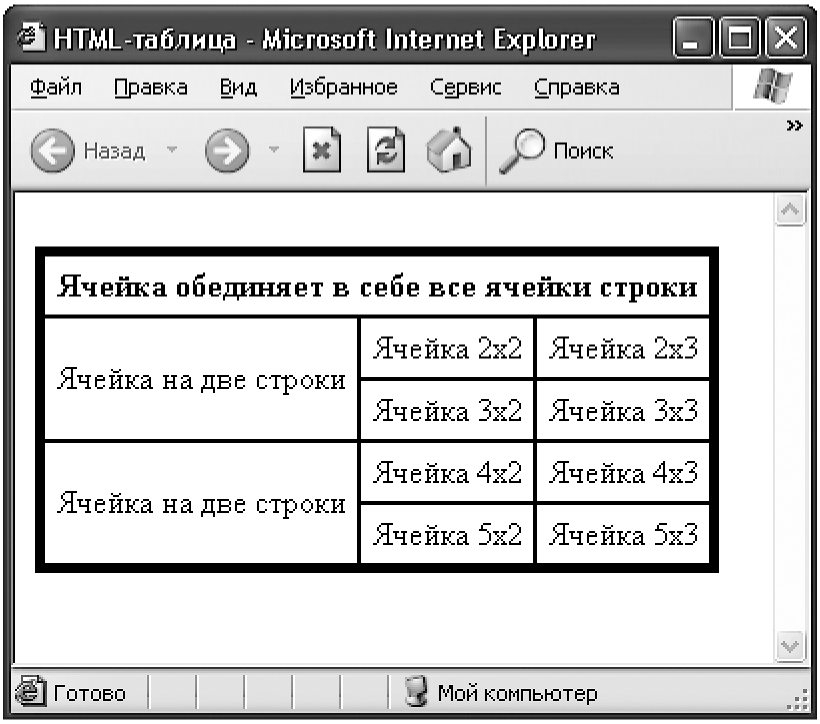

Рис. 4.3. WEB-сторінка з таблицею

<u>4 завдання</u>.

Завдання полягає у створенні WEB-сторінки з досить складною таблицею, приклад якої зображено на рис. 4.4:

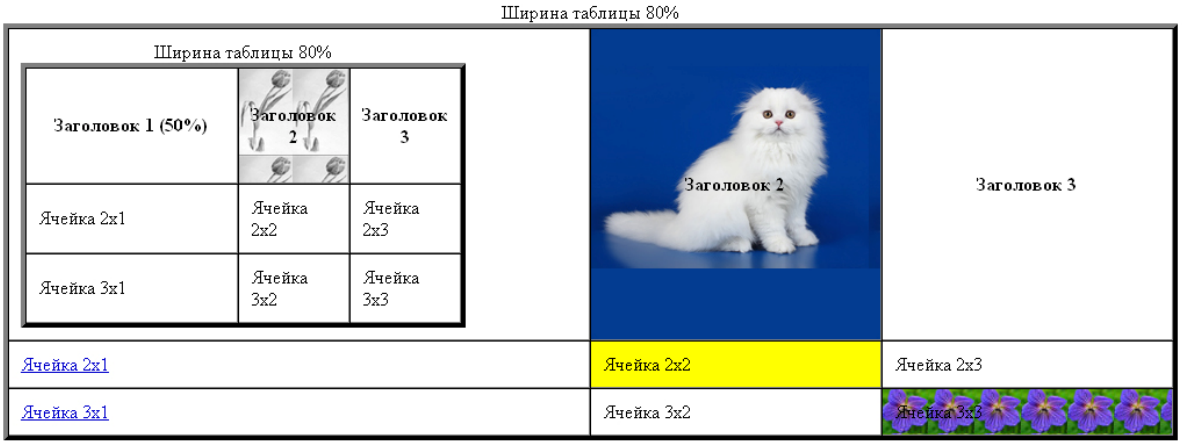

Рис. 4.4. Приклад WEB-сторінки зі складною таблицею

Малюнки, звичайно, повинні бути іншими, більш того у всіх студентів різні і розміщені в різних комірках таблиці. Тексти в комірках будь-які.

Обов’язкові параметри наступні:

- велика таблиця з 3-х рядків і 3-х стовбців, що за шириною повинна займати 80% ширини вікна браузера;
- в верхній лівій комірці розміщена маленька таблиця з 3-х рядків і 3-х стовбців, що за шириною повинна займати 80% ширини комірки;
- лівий ствбчик маленької таблиці складає за шириною 50% всієї ширини цієї таблиці;
- в деяких комірках і великої, і малої таблиці повинні бути розміщені малюнки (будь-які, і у всіх студентів різні), а деякі комірки заповнені фоновим кольором;
- в деяких комірках замість тексту повинні бути гіперпосилання на якісь інші сторінки сайту;
- WEB-сторінка повинна містити всі елементи, що вимагаються стандартом, в тому числі DOCTYPE, заголовок з необхідними метатегами, коментарі.

За результатами виконання роботи в звіті повинні бути представлені:

1. Назва роботи

2. Мета роботи

3. Скрін-шоти сторінок створених сайтів.

4. Висновки

4.3. Питання до самоконтролю

4.3.1. Що таке DOCTYPE? Опишіть структуру.

4.3.2. Що таке метатеги? Які метатеги Ви знаєте?

4.3.3. Які опції тегів гіперпосилань Ви знаєте? Їх призначення.

4.3.4. Як будується таблиця засобами HTML?

4.3.5. Яке призначення опції alt для зображень на веб-сторінці?

5. ВИКОРИСТАННЯ КАСКАДНИХ ТАБЛИЦЬ СТИЛІВ В ДИЗАЙНІ САЙТУ

Мета роботи: Набути та вдосконалити вміння використовувати технології каскадних таблиць стилів (CSS) в дизайні сайту.

5.1. Інформація для самостійної підготовки

Технологія CSS в базовому варіанті зараз широко відома і використовується майже в кожному сайті. Поряд з HTML її вивчають, починаючи зі школи.

Тому на даному практичному занятті зосередимось на кількох складних темах, пов'язаних з використанням CSS, маючи на увазі загалом володіння слухачами базовими технологіями. У випадку виникнення потреби нагадувань щодо якихось елементів технологій рекомендується звернутись до відповідних літературних джерел, наприклад [1]. При формулюванні конкретних завдань далі в тексті будуть надані деякі пояснення, що стосуються найскладніших аспектів застосування CSS.

5.2. Порядок виконання роботи:

Устаткування, матеріали та інструменти

- Персональний комп'ютер зі встановленим браузером та текстовим редактором типу Блокнот або WordPad.
- Кілька файлів малюнків в форматі .jpg.

Порядок виконання роботи

Завдання практикуму складається з 2-х частин, що пов'язані з різними темами застосування CSS.

1 завдання.

Завдання полягає в створенні WEB-сторінки з таблицею у вигляді шахівниці, як зображено нижче на рис. 5.1:

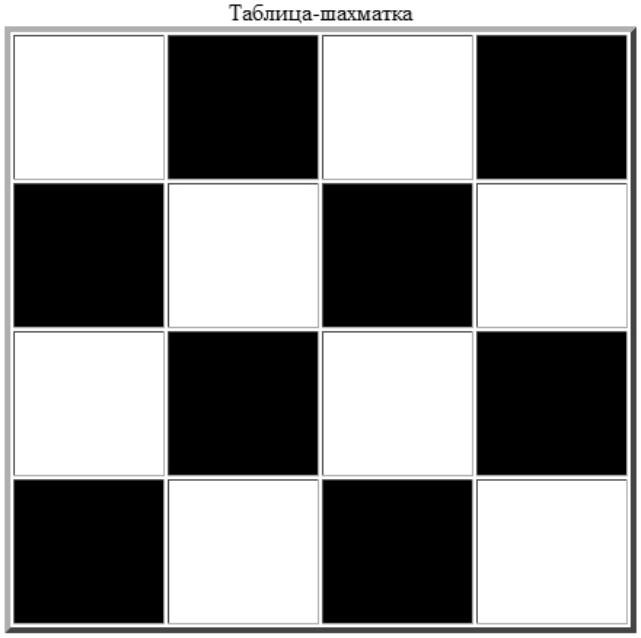

Рис. 5.1. Приклад WEB-сторінки з „шахівницею”, оформленою засобами HTML та CSS

При виконанні завдання треба використати, крім HTML, також елементи CSS (застосувати відповідні класи для білих і чорних комірок шахівниці).

2 завдання.

Завдання полягає в створенні WEB-сторінки з використанням технології “iframe” (як відомо, фреймова технологія наразі не заохочується, але “iframe” – все ж залишається у вжитку).

Вигляд сторінки наводитися нижче на рис. 5.2:

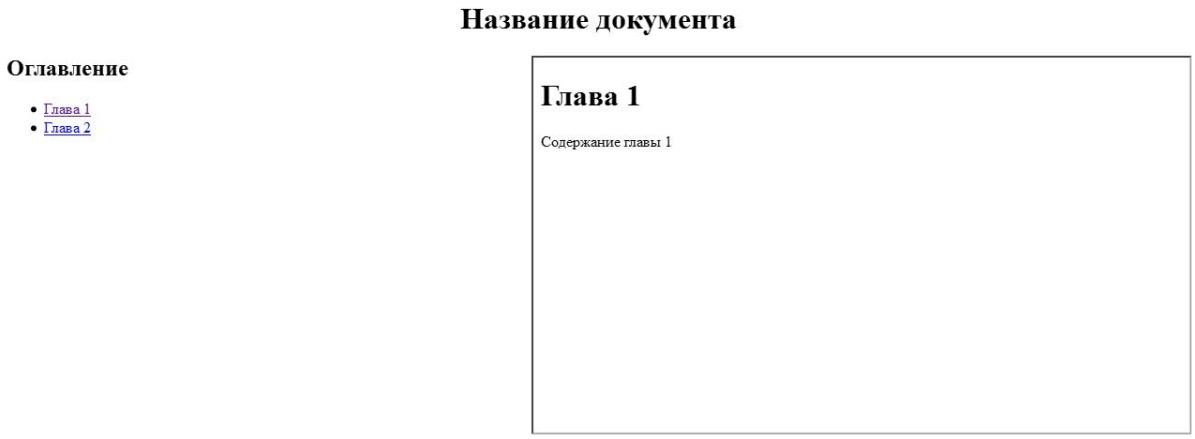

Рис. 5.2. Приклад WEB-сторінки з використанням технології “iframe”

Тут в правій частині знаходиться вікно iframe, в котрому відображується зміст того документу („Глава 1” чи „Глава 2”), який обирається в лівій частині, тобто натискання відповідного гіперпосилання в лівій частині викликає завантаження відповідного документу у вікно iframe (при натисканні гіперпосилання „Глава 1” у вікні iframe відображується вищезображена інформація, а при натисканні „Глава 2” зображення змінюється на таке: „Глава 2”, а нижче текст „Вміст глави 2”).

Для нагадування технології роботи з iframe наводжу довідкову інформацію:

Тег <iframe>. Додавання фрейма в звичайний документ

За допомогою парного тега <iframe> можна вставляти фрейми в звичайний HTML-документ. Якщо тег <iframe> не підтримується, то буде виведений текст, що міститься між тегами <iframe> і </iframe>. Інколи такі фрейми називають "плаваючими". Тег <iframe> має наступні параметри:

- src визначає URL-адресу документа, який має бути завантажений у фрейм. Може бути вказана абсолютна або відносна URL-адреса:

<iframe src="http://www.mysite.ru/doc2.html">

<iframe src="doc2.html">

- name задає унікальне ім'я фрейма:

<iframe src="chapter1.html" name="chapter">

- scrolling забороняє або дозволяє відображення смуг прокрутки у фреймі. Може набувати наступних значень:

• auto — смуги відображуються, лише якщо вміст не поміщається у фреймі (значення за умовчанням):

<iframe src="chapter1.html" name="chapter" scrolling="auto">

• yes — смуги відображуються у будь-якому випадку:

<iframe src="chapter1.html" name="chapter" scrolling="yes">

• no — смуги не відображуються у будь-якому випадку:

<iframe src="chapter1.html" name="chapter" scrolling="no">

- marginwidth і marginheight визначають відстань по горизонталі і по вертикалі між границями фрейма і його вмістом (у пікселах):

<iframe src="chapter1.html" name="chapter" marginwidth="5" marginheight="5">

- frameborder включає або відключає показ границь фрейма. Параметр може приймати одне із значень:

• 1 — кордон відображується:

<iframe src="chapter1.html" name="chapter" frameborder="1">

• 0 — кордон не відображується:

<iframe src="chapter1.html" name="chapter" frameborder="0">

- width і height задають ширину і висоту фрейма:

<iframe src="chapter1.html" name="chapter" width="200" height="200">

- align визначає вирівнювання фрейма. Може набувати наступних значень:

• left — фрейм вирівнюється по лівому краю, текст обтікає фрейм справа:

<iframe src="chapter1.html" name="chapter" align="left">

• right — фрейм вирівнюється по правому краю, текст обтікає фрейм зліва:

<iframe src="chapter1.html" name="chapter" align="right">

• top — вертикальне вирівнювання по верхньому краю:

<iframe src="chapter1.html" name="chapter" align="top">

• middle — вертикальне вирівнювання по центру:

<iframe src="chapter1.html" name="chapter" align="middle">

• bottom — вертикальне вирівнювання по нижньому краю:

<iframe src="chapter1.html" name="chapter" align="bottom">

Для завантаження документу в певний фрейм існує параметр target тега <a>. У параметрі target вказується ім'я фрейма (яке задається за допомогою параметра name тега <iframe>) або одне із зарезервованих значень: _blank, _self, _top, _parent.

Якщо потрібно завантажити документ у фрейм з ім'ям chapter, то гіперпосилання буде таким:

Текст посилання

За результатами виконання роботи в звіті повинні бути представлені:

1. Назва роботи
2. Мета роботи
3. Скрін-шоти сторінок створених сайтів.
4. Висновки

5.3. Питання до самоконтролю

5.3.1. Який синтаксис CSS? Опишіть структуру.

5.3.2. Що таке правила CSS? Опишіть структуру.

5.3.3. Які селектори CSS Ви знаєте? Їх призначення.

5.3.4. Описати призначення та принципи застосування технології “iframe”.

6. ВИКОРИСТАННЯ РОЗШИРЕНИХ МОЖЛИВОСТЕЙ КАСКАДНИХ ТАБЛИЦЬ СТИЛІВ В ДИЗАЙНІ САЙТУ

Мета роботи: Вдосконалити вміння використовувати технології каскадних таблиць стилів (CSS) в дизайні сайту.

6.1. Інформація для самостійної підготовки

Враховуючи засвоєний матеріал попереднього практичного заняття, на даному практичному занятті зосередимось на кількох ще більш складних темах, пов'язаних з використанням CSS. У випадку виникнення потреби нагадувань щодо якихось елементів технологій рекомендується звернутись до відповідних літературних джерел, наприклад [1]. При формулюванні конкретних завдань далі в тексті будуть надані деякі пояснення, що стосуються найскладніших аспектів застосування CSS.

6.2. Порядок виконання роботи:

Устаткування, матеріали та інструменти

- Персональний комп'ютер зі встановленим браузером та текстовим редактором типу Блокнот або WordPad.
- Кілька файлів малюнків в форматі .jpg.

Порядок виконання роботи

Завдання практикуму складається з 3-х частин, що пов'язані з різними темами застосування CSS.

1 завдання.

Завдання полягає у створенні web-сторінки зі списком трьох гіперпосилань, які при їх обиранні повинні здійснювати перехід на інші відповідні web-сторінки (їх оформлення на Ваш смак). Обов'язковою особливістю списку гіперпосилань є його оформлення не звичайними марками типу кружечка або квадратика, а маленьким малюнком

(зображення на Ваш смак), який змінюється на інший малюнок при обиранні мишею відповідного пункту списку.

Приклад зображено на рис. 6.1:

Рис. 6.1. Приклад web-сторінки зі списком гіперпосилань

(В цьому прикладі обрано третій пункт, при цьому малюнок марки списку змінився з глобусу на конверт).

Підказка: приклад використання малюнку link_icon_email.gif (який знаходиться в папці images) у ролі марки списку для того елементу списку, на котрий наведено мишу:

```
a:hover {
  padding-left: 20px;
  background-image: url(images/link_icon_email.gif);
  background-repeat: no-repeat;
}
```

2 завдання.

Завдання полягає у створенні WEB-сторінки з „кнопковим” меню, тобто 4 гіперпосилання повинні мати вигляд кнопок, створених засобами CSS, як зображено на рис. 6.2:

Рис. 6.2. Приклад web-сторінки з „кнопковим” меню

Кожна кнопка є гіперпосиланням на якусь веб-сторінку. Але головне – в оформленні цих кнопок: в початковому стані вони виглядають як випуклі

(ненатиснуті) кнопки (на рис. 6.2 це перша, друга та четверта кнопки), а при наведенні миші на кнопку – вона стає немовби втиснута (на рис. 6.2 – це третя кнопка).

Підказка: ефект створюється завдяки CSS-параметрам

border-top:

border-left:

border-bottom:

border-right:

В початковому положенні лівий і верхній border білиий, а правий і нижній – сірі; а при наведенні миші – навпаки.

Відключення марок по замовчуванню:

list-style: none;

3 завдання.

Завдання полягає у створенні WEB-сторінки з „каркасом шаблону" сайту, створеним за технологією контейнерної верстки.

Приклад зображено на рис. 6.3:

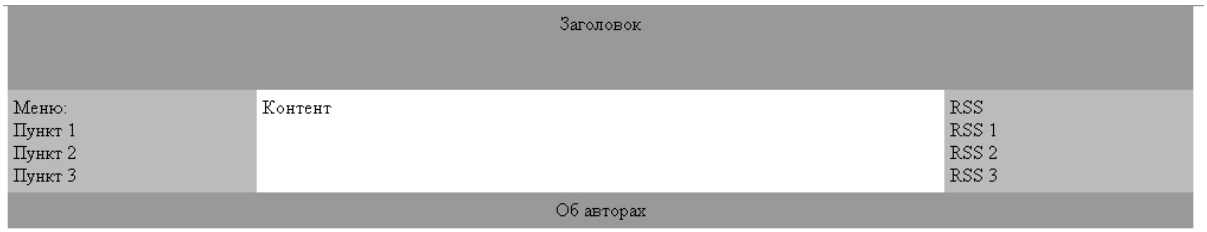

Рис. 6.3. Приклад web-сторінки з „каркасом шаблону" сайту, створеним за технологією контейнерної верстки

Цей каркас шаблону створює місця для розміщення „шапки" сайту (зверху), меню (зліва), контенту (посередині), додаткових модулів (зправа), „footer" (знизу).

Шаблон оформлюється засобами CSS.

Необхідно в цьому шаблоні розмістити якісь зразки малюнку зверху, контенту посередині, меню зліва із кількох пунктів з реалізацією переходів на інші сторінки (оформлені на Ваш смак), в „footer” (знизу) розмістити інформацію про розробника сайту.

За результатами виконання роботи в звіті повинні бути представлені:

1. Назва роботи
2. Мета роботи
3. Скрін-шоти сторінок створених сайтів.
4. Висновки

6.3. Питання до самоконтролю

6.3.1. Як Ви розумієте термін „каскадність” CSS?

6.3.2. Яким чином організовано пріоритетність стилів CSS?

6.3.3. Які є варіанти підключення каскадних таблиць стилів до web-сторінок?

6.3.4. Описати призначення та принципи застосування технології контейнерної верстки.

7. РОЗРОБКА ТА ТРАНСФОРМАЦІЯ XML-СТОРІНОК

Мета роботи: Засвоїти основи використання технологій мови XML та перетворень XML-сторінок.

7.1. Інформація для самостійної підготовки

Технології мови XML в даний час використовуються дуже широко і різнобічно, від функції складової частини технології Ajax до адаптивних технологій WEB-дизайну, формату передачі даних між складовими частинами програмних систем та, навіть, лог-файлів. Тому доцільно приділити цим технологіям достатню увагу.

XML — це спроба удосконалити HTML шляхом створення простої мови розмітки, що описує довільні структуровані дані. Точніше кажучи, це метамова, на якій пишуться спеціалізовані мови, що описують дані певної структури. Такі мови називаються XML-словниками. На відміну від HTML, XML не містить жодних вказівок на те, як повинні відображуватися описані в XML-документі дані. Спосіб відображення даних для різних пристроїв задається мовою опису стилів XSL, який грає для XML приблизно ту ж роль, що CSS для HTML. Інша принципова його відмінність від HTML полягає в тому, що XML може містити будь-які теги, які вважають за потрібне використовувати творці XML-словника.

Простий документ XML може виглядати таким чином:

```
<?xml version="1.1"?>
<list_of_items>
<item id="1"><first/>Перший</item>
<item id="2">Другий <sub_item>підпункт 1</sub_item></item>
<item id="3">Третій</item>
<item id="4"><last/>Останній</item>
```

</list_of_items>

Як видно, документ дуже схожий на звичайну HTML-сторінку. Так само, як і в HTML, інструкції, розміщені у кутових дужках, називаються тегами і служать для розмітки основного тексту документа. У XML існують відкриваючі, закриваючі і порожні теги.

Тіло документа XML складається з елементів розмітки (markup) і безпосередньо вмісту документа - даних (content). XML-тэги призначені для означення елементів документа, їх атрибутів і інших конструкцій мови.

Будь-який XML- документ повинен завжди починатися з інструкції <?xml?>, усередині якої також можна задавати номер версії мови, номер кодової сторінки і інші параметри, необхідні програмі-аналізатору в процесі розбору документа.

Як бачимо, структуру і правила побудови самого XML-документа зрозуміти досить легко [1].

Проте логіку використання мови стилів XSL – значно складніше. Тому зосередимось на деяких поясненнях у цьому питанні, що прийдуть у нагоді при виконанні практичного завдання.

XSL ("розширювана мова таблиць стилів" від англ. *eXtensible Stylesheet Language*) — мова перетворення і візуалізації для XML.

XSLT означає перетворення або трансформація XSL (від англ. *XSL Transformations*) — мова перетворення XML документів.

Покажемо, як можна використовувати XSLT для трансформації XML документів в інші формати (наприклад, для трансформації XML в HTML).

У HTML використовуються відомі визначені теги, значення і спосіб відображення яких добре зрозумілі.

CSS використовується для додавання стилів елементам HTML.

У XML використовуються теги, не визначені заздалегідь, що робить значення кожного тегу не ясним спочатку.

Наприклад, елемент <table> в HTML означає таблицю і нічого іншого. У XML цей елемент може означати все що завгодно — таблицю, стіл і т. д. — і браузери не знають напевно, як його відображувати.

Тут на допомогу приходить XSL, який дозволяє описати вигляд, як елементи XML повинні відображуватися в браузері.

XSL — більше, ніж просто мова таблиці стилів

XSL складається з трьох частин:

- XSLT — мова перетворення XML документів,
- XPath — мова для навігації по елементах XML документа,
- XQuery — мова, що дозволяє робити вибірки з XML даних.

XSLT — найбільш важлива частина мови XSL, використовується для перетворення XML-документа в іншій XML-документ або в документ іншого типу, який може розпізнаватись браузером, наприклад, HTML і XHTML. Зазвичай, XSLT робить це, перетворюючи кожен XML-елемент в (X)HTML-елемент.

За допомогою XSLT можна додавати/знищувати елементи і атрибути в кінцевий файл. Також можна реорганізовувати і сортувати елементи, тестувати, визначати, які елементи приховати або відобразити, і багато чого іншого.

У загальних фразах процес перетворення можна описати таким чином — XSLT перетворює вхідне дерево XML в XML дерево-результат.

Для пошуку елементів в XML-документі XSLT використовує мову XPath. Мова XPath дозволяє переміщатися по елементах і атрибутах XML-документа.

Як це працює?

В процесі перетворення XSLT за допомогою XPath визначає ті частини вхідного документа, які повинні відповідати одному або декільком визначеним шаблонам. Якщо відповідність буде знайдена, то XSLT перетворить цю частину вхідного документа і створить кінцевий документ.

Всі основні браузери підтримують XSLT і XPath.

Кореневим елементом, що декларує документ таблиці стилів XSL, є <xsl:stylesheet> або <xsl:transform>. Елементи <xsl:stylesheet> і <xsl:transform> є повними синонімами, і для декларації таблиці стилів можна використовувати будь-який з них.

Згідно рекомендації консорціуму W3C таблиця стилів XSL декларується таким чином:

```
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
```

або

```
<xsl:transform version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
```

Аби дістати доступ до елементів, атрибутів і інших функцій XSLT, необхідно на початку документа декларувати простір імен XSLT.

Рядок

```
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
```

вказує на офіційний простір імен XSLT консорціуму W3C. Якщо використовується цей простір імен, то треба також вказувати і атрибут *version="1.0"*.

Приклад.

Припустимо, що нам потрібно перетворити наступний XML-документ (каталог CD-дисків) в XHTML:

```
<?xml version="1.0" encoding="UTF-8"?>
<catalog>
```

```
  <cd>
    <title>Picture book</title>
    <artist>Simply Red</artist>
    <country>EU</country>
    <company>Elektra</company>
    <price>7.20</price>
    <year>1985</year>
  </cd>
  <cd>
    <title>Red</title>
    <artist>The Communards</artist>
    <country>UK</country>
    <company>London</company>
    <price>7.80</price>
    <year>1987</year>
  </cd>
....
  </catalog>
```

XML-документ можна переглянути в браузері і без стильових таблиць. При відкритті в браузері він відображуватиметься у вигляді забарвлених в різні кольори кореневого і дочірніх елементів і їх вмісту. Часто зліва від елементів XML-дерева виводиться знак плюса (+) або мінуса (-), при натисканні на який можна розгорнути/згорнути структуру елементу.

Тепер створюємо таблицю стилів XSL ("Style.xsl") з шаблоном перетворення:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
```

```
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
  <html>
  <body>
  <h2>My CD Collection</h2>
  <table border="1">
    <tr bgcolor="#9acd32">
      <th>Title</th>
      <th>Artist</th>
    </tr>
    <xsl:for-each select="catalog/cd">
    <tr>
      <td><xsl:value-of select="title"/></td>
      <td><xsl:value-of select="artist"/></td>
    </tr>
    </xsl:for-each>
  </table>
  </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

Підключаємо таблицю стилів XSL до XML-документу (додаємо посилання на таблицю стилів XSL в XML-документ):

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="Style.xsl"?>
<catalog>
  <cd>
    <title>Picture book</title>
```

```
  <artist>Simply Red</artist>
  <country>EU</country>
  <company>Elektra</company>
  <price>7.20</price>
  <year>1985</year>
 </cd>
 <cd>
  <title>Red</title>
  <artist>The Communards</artist>
  <country>UK</country>
  <company>London</company>
  <price>7.80</price>
  <year>1987</year>
 </cd>
....
 </catalog>
```

Якщо використовується XSLT-сумісний браузер, то побачимо коректно перетворений з XML в HTML документ (рис. 7.1).

My CD Collection

Title	Artist
Picture book	Simply Red
Red	The Communards
. . .	. . .

Рис. 7.1. Результат XSLT-перетворення XML-документа в HTML-сторінку

Таблиця стилів XSL містить один або більше наборів правил перетворення, які називаються шаблонами перетворення. Шаблон перетворення містить правила, які застосовуються, коли знайдений вузол, відповідає умові пошуку.

Для створення шаблонів перетворення використовується елемент **<xsl:template>**.

При цьому атрибут ***match*** використовується для асоціації шаблону з XML-елементом. Також, атрибут ***match*** може використовуватися, аби визначити шаблон для всього XML-документу цілком. Значення атрибуту ***match*** – це вираз XPath (наприклад, *match="/"* визначає весь документ).

Пояснення

Оскільки таблиця стилів XSL це XML документ, він завжди повинен починатися з XML декларації: **<?xml version="1.0" encoding="UTF-8"?>**.

Наступний елемент, **<xsl:stylesheet>**, визначає, що даний документ це таблиця стилів XSLT (з атрибутами номера версії і простору імен XSLT).

Елемент **<xsl:template>** визначає шаблон. Атрибут ***match="/"*** асоціює шаблон з кореневим елементом вхідного XML-документа, тобто зазначає, що шаблон буде застосовуватись до всього вхідного документу.

Вміст елементу **<xsl:template>** визначає деякий HTML код, який записується у вихідний документ.

Останні два рядки позначають кінець шаблону і кінець таблиці стилів.

Тепер про отримання значень відібраних XML-елементів і додавання їх у вихідний потік перетворюваного документа.

Для цього використовується елемент **<xsl:value-of>**.

Приклад:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
 xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
 <xsl:template match="/">
  <html>
  <body>
  <h2>My CD Collection</h2>
  <table border="1">
   <tr bgcolor="#9acd32">
    <th>Title</th>
    <th>Artist</th>
   </tr>
   <tr>
    <td><xsl:value-of select="catalog/cd/title"/></td>
    <td><xsl:value-of select="catalog/cd/artist"/></td>
   </tr>
  </table>
  </body>
  </html>
 </xsl:template>
</xsl:stylesheet>
```

Атрибут ***select*** у наведеному прикладі містить вираз XPath, який працює, як навігація по файловій системі; пряма коса риска (/) обирає піддиректорії.

В результаті у вихідний XML документ будуть скопійовані необхідні дані, але лише один рядок даних.

Для проходу в циклі по XML-елементам і відображення всіх записів слід використовувати елемент **<xsl:for-each>** .

Елемент **<xsl:for-each>** дозволяє організовувати цикли в процесі XSLT перетворення.

XSL-елемент **<xsl:for-each>** може використовуватися для вибору кожного XML-елементу заданого вузлового набору.

Приклад (відповідає таблиці каталогу CD-дисків, що раніше розглядався):

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
 xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
 <xsl:template match="/">
  <html>
  <body>
  <h2>My CD Collection</h2>
  <table border="1">
   <tr bgcolor="#9acd32">
    <th>Title</th>
    <th>Artist</th>
   </tr>
  <xsl:for-each select="catalog/cd">
   <tr>
    <td><xsl:value-of select="title"/></td>
    <td><xsl:value-of select="artist"/></td>
   </tr>
  </xsl:for-each>
 </table>
 </body>
```

```
  </html>
 </xsl:template>
</xsl:stylesheet>
```

Значенням атрибуту ***select*** в наведеному прикладі є вираз XPath, що працює, як навігація по файловій системі, де пряма коса риска (/) обирає піддиректорії.

Також, можна здійснювати фільтрацію виводу з XML-файлу. Для цього слід додати потрібний критерій до атрибуту ***select*** элемента <xsl:for-each>.

Приклад:

```
<xsl:for-each select="catalog/cd[artist='Bob Dylan']">
```

Допустимі оператори фільтрації:

= (рівно)

!= (не рівно)

< (менше ніж)

> (більш ніж)

Елемент **<xsl:sort>** використовується для сортування вихідних даних.

Аби здійснити сортування вихідних даних, необхідно додати елемент **<xsl:sort>** усередині елементу <xsl:for-each> в XSL файлі.

Приклад:

```
</tr>
    <xsl:for-each select="catalog/cd">
     <xsl:sort select="artist"/>
      <tr>
```

```
      <td><xsl:value-of select="title"/></td>
      <td><xsl:value-of select="artist"/></td>
    </tr>
  </xsl:for-each>
```

Атрибут ***select*** вказує, по якому XML елементу слід сортувати.

Елемент **<xsl:if>** дозволяє задати перевірку на відповідність певній умові вмісту XML файлу.

Аби задати перевірку вмісту XML файлу, потрібно додати елемент **<xsl:if>** в XSL документ.

Синтаксис

```
<xsl:if test="вираз">
...деякі дані, що виводяться, якщо умова виконується...
</xsl:if>
```

Приклад:

```
</tr>
  <xsl:for-each select="catalog/cd">
   <xsl:if test="price &gt; 10">
     <tr>
       <td><xsl:value-of select="title"/></td>
       <td><xsl:value-of select="artist"/></td>
       <td><xsl:value-of select="price"/></td>
     </tr>
   </xsl:if>
  </xsl:for-each>
```

Значення обов'язкового атрибуту ***test*** містить тестовий вираз.

Відповідно до наведеного вище коду виводитися будуть лише ті елементи title і artist, в яких елемент price має значення більше 10.

Елемент **<xsl:apply-templates>** застосовує деякий шаблон до поточного елементу або до дочірнього вузла поточного елементу.

Якщо в елемент **<xsl:apply-templates>** додати атрибут ***select***, то він відноситиметься лише до дочірнього елементу, який відповідає значенню цього атрибуту. Таким чином, атрибут ***select*** може використовуватися для визначення порядку, в якому оброблятимуться дочірні вузли.

7.2. Порядок виконання роботи:

Устаткування, матеріали та інструменти

- Персональний комп’ютер зі встановленим браузером та текстовим редактором типу Блокнот або WordPad.

Порядок виконання роботи

Завдання практикуму полягає у створенні системи на базі XML-файлу з інформацією щодо рейсів різних авіакомпаній у різні міста та XSLT-перетворення, що виводить HTML-сторінку з таблицею, що є результатом пошуку рейсів, які задовольняють користувача, наприклад, всіх рейсів до Києва.

Структуру та контент XML-файлу бази даних рейсів студент створює за своїм смаком, файл XSLT – також за своїм смаком, але найбільш оптимальним чином для зручного пошуку та виведення необхідної інформації користувачем.

Приклад таблиці з отриманими результатами наведено на рис. 7.2.

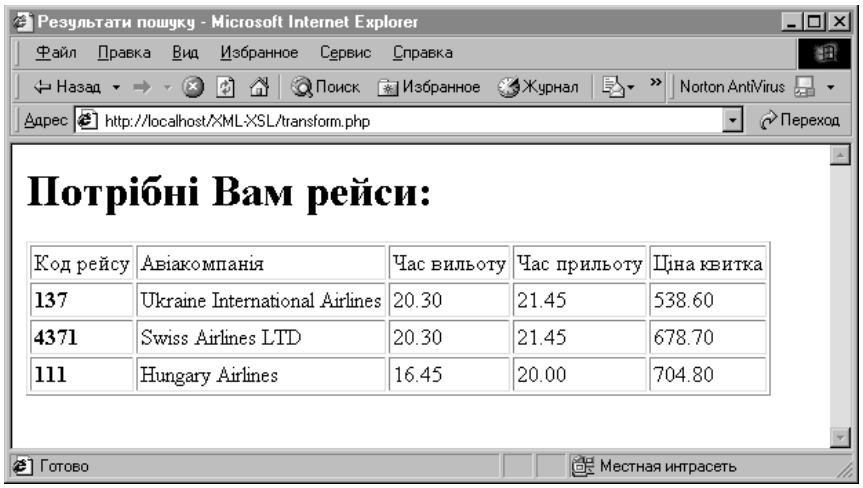

Рис. 7.2. Приклад таблиці з результатами пошуку авіарейсів

За результатами виконання роботи в звіті повинні бути представлені:

1. Назва роботи
2. Мета роботи
3. Скрін-шоти та лістинги створених файлів та сторінок.
4. Висновки

7.3. Питання до самоконтролю

7.3.1. Чим XML відрізняється від HTML?

7.3.2. Як використовується XML?

7.3.3. Що таке парсер?

7.3.4. Що таке XSL і для чого використовується?

7.3.5. Якими способами можна відобразити XML-файл в HTML-сторінку?

8. ТЕХНОЛОГІЇ КАСКАДНИХ ТАБЛИЦЬ СТИЛІВ ТА ЗАСОБІВ МОВИ JAVASCRIPT У WEB-ДИЗАЙНІ

Мета роботи: Вдосконалити вміння використовувати технології каскадних таблиць стилів (CSS) та оволодіти основами застосування засобів сценаріїв JavaScript в дизайні сайту.

8.1. Інформація для самостійної підготовки

Серед кількох існуючих в даний час засобів створення клієнтських сценаріїв найбільшого поширення набула технологія мови JavaScript. Ця мова наразі відіграє роль неформального стандарту в даному напрямі web-технологій.

За допомогою мови JavaScript можна динамічно керувати зовнішнім виглядом та поведінкою усіх HTML-елементів на сторінці. Об'єкти JavaScript, які представляють HTML-елементи, мають властивості, що відповідають атрибутам HTML-тега (часто назви цих властивостей співпадають з назвами відповідних атрибутів).

В середовищі JavaScript всі елементи web-сторінок і оточення, у тому числі параметри браузера, екрану монітора, операційної системи, процесора і т.п. створюють об’єктну модель, що надає можливість динамічної зміни будь-яких їх властивостей.

При підготовці до виконання практикуму слід засвоїти навчальний матеріал з відповідних розділів підручника [1]. А при формулюванні конкретних завдань далі в тексті будуть надані деякі пояснення, що стосуються найскладніших аспектів застосування CSS і JavaScript.

8.2. Порядок виконання роботи:

Устаткування, матеріали та інструменти

- Персональний комп’ютер зі встановленим браузером та текстовим редактором типу Блокнот або WordPad.

- Кілька файлів малюнків в форматі .jpg.

Порядок виконання роботи

Завдання практикуму складається з 3-х частин, що пов'язані з різними темами застосування CSS і JavaScript.

1 завдання.

Завдання полягає у створенні web-сторінки з „каркасом шаблону" сайту, створеним за технологією табличної верстки.

Приклад зображено на рис. 8.1:

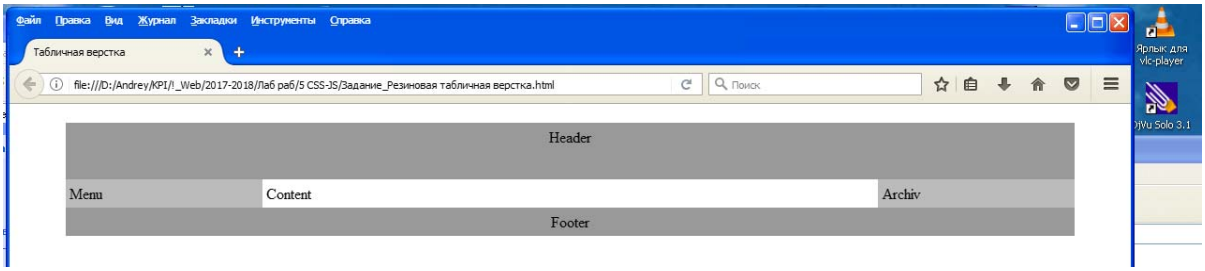

Рис. 8.1. Приклад web-сторінки з „каркасом шаблону" сайту, створеним за технологією табличної верстки

Цей каркас шаблону створює місця (блоки) для розміщення „шапки" сайту (зверху), меню (зліва), контенту (посередині), додаткових модулів (зправа), „footer" (знизу).

Шаблон оформлюється засобами CSS.

Необхідно в цьому шаблоні розмістити якісь зразки малюнку зверху, контенту посередині, меню зліва із кількох пунктів з реалізацією переходів на інші сторінки (оформлені на Ваш смак), в „footer" (знизу) розмістити інформацію про розробника сайту.

Верстка повинна бути „резиновою", тобто при зменшенні горизонтального розміру документа шаблон намагається якнайдовше зберігати розміри лівого і правого блоку, а зменшувати лише розмір блоку-контенту, і тільки за неможливості цього – вже зменшувати і розмір лівого і правого блоків.

Підказка:

Доцільно використати у відповідних CSS-правилах приблизно таке:

margin: 20px auto;

а для інших елементів приблизно таке:

width: 200;

2 завдання.

Завдання полягає у створенні web-сторінки з калькулятором, який здатен (завдяки використанню технологій JavaScript) виконувати такі арифметичні дії над двома числами, які користувач вводить у два поля форми:

додавання, віднімання, перемноження, ділення.

Вигляд калькулятора на сторінці будь-який, наприклад, такий, як на рис. 8.2:

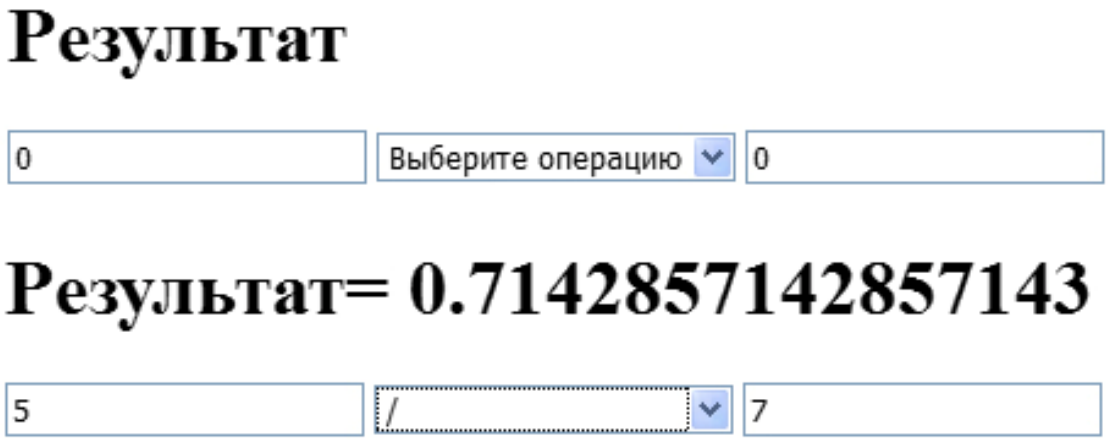

Рис. 8.2. Приклад web-сторінки з інтерфейсом калькулятора

Підказки:

- Набути введеного значення числа з форми можна таким чином: var n=Number(document.getElementById("num1").value),

де "num1"- це id поля введення форми, слово Number - для перетворення введеного рядка в число.

- Викликати сценарій обчислень з форми можна по параметру поля select форми onchange="функція сценарію".

- document.getElementById("res").innerHTML="Результат="+result; – запис в HTML-елемент із збереженням розмітки.

3 завдання.

Завдання полягає у створенні web-сторінки з двома прямокутниками різного кольору, що частково накладаються один на другий, як зображено на рис. 8.3

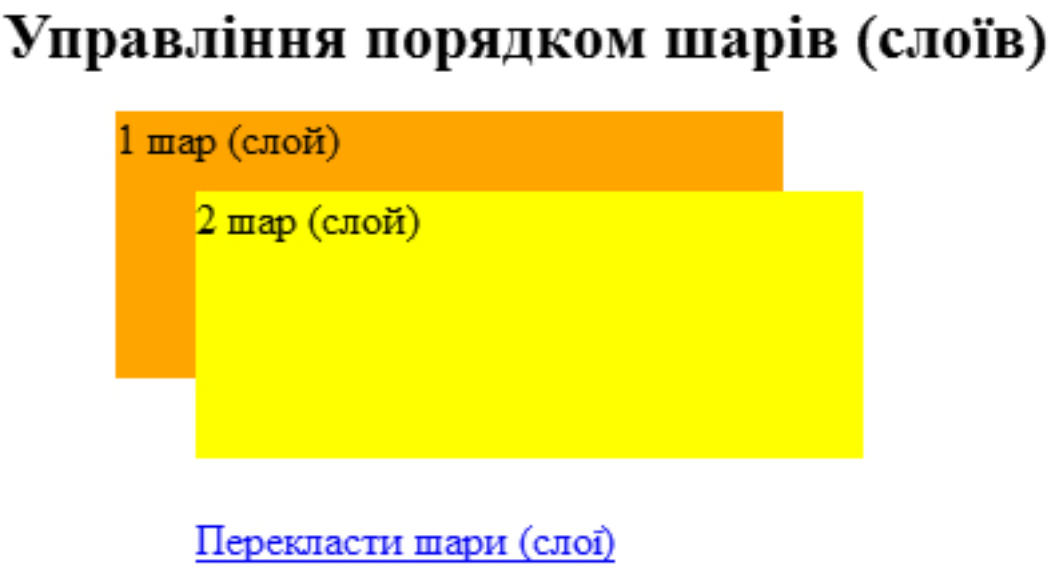

Рис. 8.3. web-сторінка з двома прямокутниками (початковий стан)

У нижній частині є активний напис „Перекласти шари (слої)”, при натисканні на котрий відбувається перекладання шарів (слоїв), тобто померанчевий шар, котрий спочатку був нижче жовтого, стає вище його, як зображено на наступному рис. 8.4.

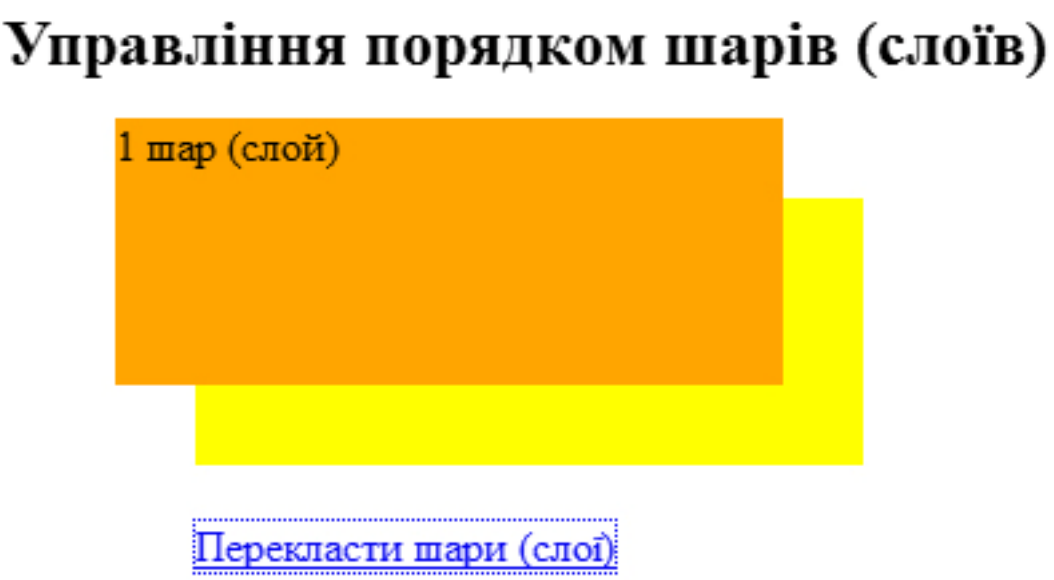

Рис. 8.4. web-сторінка з двома прямокутниками (після перекладання шарів)

Підказки:

- Порядок шарів (слоїв) задається z-індексом, який в стилях пишеться z-index:1, а в JS-сценариях ...style.zIndex.
- Чим більше z-індекс, тим вище шар (слой).
- Об'єкти-прямокутники можна створити, як div-блоки.
- Дістати доступ до такого об'єкту в JS-сценарии можна таким чином: var Layer=document.getElementById("id об'єкту").
- А потім можна використовувати параметр style.zIndex отриманого объекта.
- Перш ніж міняти цю властивість, слід перевіряти, який об'є кт вище (по його z-індексу).
- Перекласти шари (слої) - приклад визову сценарію move().

За результатами виконання роботи в звіті повинні бути представлені:

1. Назва роботи

2. Мета роботи

3. Скрін-шоти сторінок створених сайтів.

4. Висновки

8.3. Питання до самоконтролю

8.3.1. Навіщо використовуються клієнтські сценарії?

8.3.2. Що таке DOM і BOM в технології мови JavaScript?

8.3.3. Яким чином можна отримати доступ до об’єкта на web-сторінці? Як можна змінити цей об’єкт?

8.3.4. Опишіть всі методи пов’язування (вбудовування) JavaScript-коду з HTML-документом

9. ТЕХНОЛОГІЯ КЛІЄНТСЬКИХ СЦЕНАРІЇВ JAVASCRIPT У WEB-ДИЗАЙНІ

Мета роботи: Удосконалити вміння використовувати технології клієнтських сценаріїв JavaScript в дизайні сайту.

9.1. Інформація для самостійної підготовки

Засвоєння технології сценаріїв JavaScript надзвичайно важливе. Достатньо сказати, що, крім ролі неформально стандартного механізму створення клієнтських сценаріїв у web-дизайні, мова JavaScript використовується і в інших технологіях програмування, наприклад, при автоматизації застосувань програмних продуктів Adobe: Photoshop, Illustrator, InDrsign, Acrobat. Технології JavaScript дуже потужні і універсальні, і саме тому підтримуються зараз усіма браузерами, що підкреслює важливість їх поглибленого вивчення.

Теоретичні засади рекомендується вивчати з відповідних розділів підручника [1]. А при формулюванні конкретних завдань далі в тексті будуть надані деякі пояснення, що стосуються найскладніших аспектів застосування JavaScript.

9.2. Порядок виконання роботи:

Устаткування, матеріали та інструменти

- Персональний комп’ютер зі встановленим браузером та текстовим редактором типу Блокнот або WordPad.
- Кілька файлів малюнків в форматі .jpg.

Порядок виконання роботи

Завдання практикуму складається з 3-х частин, що пов’язані з різними темами застосування JavaScript.

1 завдання.

Завдання полягає у створенні web-сторінки з календарем та реальним часом завантаження сторінки, тобто на сторінці повинна відображуватись поточна дата і час таким чином:

Сегодня
четверг 26 октября 2017 11:06:48
26.10.17

Підказки:

- для отримання поточної дати і часу можна використати вираз

```
var d = new Date();
```

тоді в змінній (об'єкті) d буде міститись вся інформація про поточний час і дату;

- для отримання інформації про день тижня, дату, місяць, рік, годину, хвилини та секунди можна використати такі методи раніше створеного об'єкту d:

```
getDay(), getDate(), getMonth(), getFullYear(),
getHours(), getMinutes(), getSeconds()
```

- але ці значення дня тижня, місяця і т.п. будуть у вигляді числа - порядкового номера (тобто понеділок – це 1, вівторок – це 2 і т.д.). Щоб перетворити їх у назви днів тижня, місяців і т.д. пропоную створити масиви цих назв, а вибирати відповідні елементи таких масивів, використовуючи вищезазначені порядкові числа, як індекси цих масивів.

2 завдання.

Завдання полягає у створенні web-сторінки з годинником, що іде і показує реальний час, наприклад, таким чином (рис.9.1):

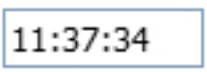

Рис. 9.1. Приклад web-сторінки з годинником

Підказки:

- для отримання поточної дати і часу знову можна використати вираз

```
var d = new Date();
```

тоді в змінній (об'єкті) d буде міститися вся інформація про поточний час і дату;

- для отримання інформації про годину, хвилини та секунди можна також знову використати такі ж методи раніше створеного об'єкту d:

```
getHours(), getMinutes(), getSeconds()
```

а для динаміки, тобто, щоб годинник ішов в реальному часі пропоную використати таку конструкцію:

```
id=setTimeout("clock_form()",100)
```

де функція `setTimeout("clock_form()",100)` запускає функцію `clock_form()` кожні 100 мілісекунд. Тобто, якщо всі вищезазначені операції з отримання реального часу розмістити у функції `clock_form()` і там же розмістити `id=setTimeout("clock_form()",100)`, то все це буде повторюватись в циклі зі затримкою 100 мс, тобто годинник буде іти.

3 завдання.

Завдання полягає у створенні WEB-сторінки з постійно біжучим написом (рис.9.2):

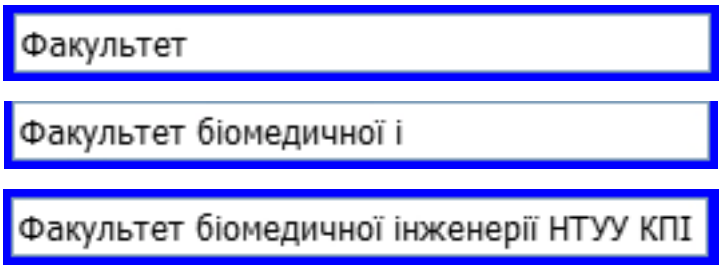

Рис. 9.2. Приклад web-сторінки з біжучим написом

Підказки:

- для динаміки використати таку ж технологію, як у попередньому завданні, а для добавляння букв у кожній ітерації циклу використати, наприклад, такий метод об’єкту-рядка символів `line`

```
line.substring(0,i)
```

котрий виділяє підстроку з літерами від 0-го до і-го з об’єкту-строки `line`.

Цей об’єкт-строка символів має такий зміст:

var line="Факультет біомедичної інженерії НТУУ КПІ”

За результатами виконання роботи в звіті повинні бути представлені:

1. Назва роботи
2. Мета роботи
3. Скрін-шоти сторінок створених сайтів.
4. Висновки

9.3. Питання до самоконтролю

9.3.1. Розкрити зміст поняття об'єктної моделі документа (DOM) в JavaScript.

9.3.2. Проаналізувати використання властивостей, методів і подій для об'єктів в JavaScript?

9.3.3. Для чого служить об'єкт Math в JavaScript?

9.3.4. Для чого служить об'єкт Date в JavaScript?

10. ВИКОРИСТАННЯ РОЗШИРЕНИХ МОЖЛИВОСТЕЙ ТЕХНОЛОГІЇ КЛІЄНТСЬКИХ СЦЕНАРІЇВ JAVASCRIPT У WEB-ДИЗАЙНІ

Мета роботи: Подальше удосконалення вміння використовувати технології клієнтських сценаріїв JavaScript в дизайні сайту.

10.1. Інформація для самостійної підготовки

Зважаючи важливу роль технології сценаріїв JavaScript у web-програмуванні та типовість застосування об'єктно-орієнтованого напряму стосовно інших мов і засобів, доцільно закріпити і удосконалити отримані на попередніх практикумах знання, виконуючи ще складніші завдання.

Теоретичні засади знову рекомендується вивчати з відповідних розділів підручника [1]. А при формулюванні конкретних завдань далі в тексті будуть надані деякі пояснення, що стосуються найскладніших аспектів застосування JavaScript.

10.2. Порядок виконання роботи:

Устаткування, матеріали та інструменти

- Персональний комп'ютер зі встановленим браузером та текстовим редактором типу Блокнот або WordPad.
- Кілька файлів малюнків в форматі .jpg.

Порядок виконання роботи

Завдання практикуму складається з 3-х частин, що пов'язані з різними темами застосування JavaScript.

1 завдання.

Завдання полягає у наступному:

- створити таблицю, наприклад, студентів з графою „дата народження", як показано на рис.10.1:

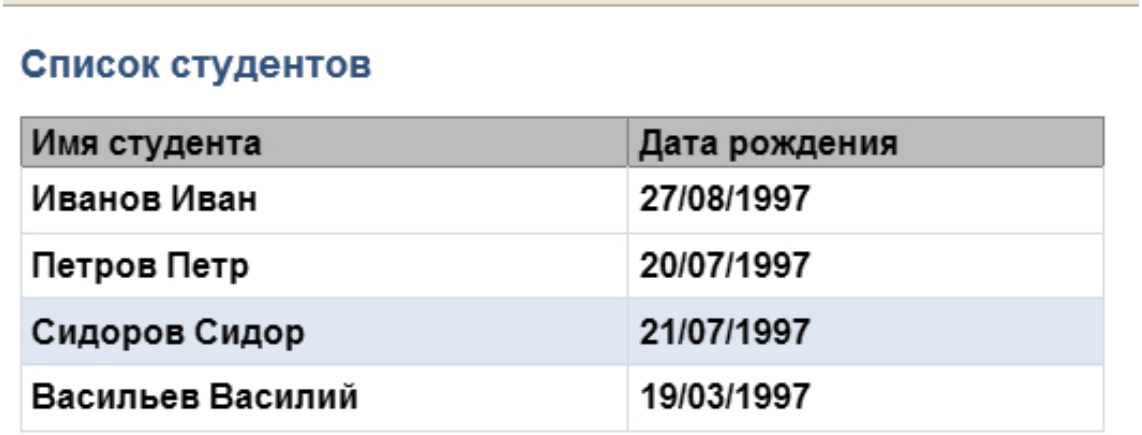

Рис. 10.1. Приклад web-сторінки з активною таблицею

- при наведенні миші на якийсь рядок, фон цього рядка повинен змінюватися; при виведенні миші з рядка його фон повертається до попереднього вигляду;
- зробити це треба засобами JavaScript через зміну стилю відповідного рядка при винекненні подій наведення миші та виведення миші.

Підказки:

- Існує така властивість, наприклад, об’єкта `this`, через яку можна змінити CSS клас, що засосовується до цього об’єкта
  `this.className`
- Або можна зчитувати значення якогось атрибута якогось об’єкта (елемента) методом `getAttribute()`, а встановлювати значення атрибута методом `setAttribute()`
- Або можна стильову властивість змінювати таким чином (тут змінюється колір параграфів)
  `par.style.color="red";`

2 завдання.

Завдання полягає у наступному:

- Виводиться запит рядка тексту.
- Після введення рядка програма виконує сортування його літер в алфавітному порядку.
- Виводиться вхідний рядок та рядок, що є результатом сортування.

Підказки:

- Метод split("") розбиває рядок на масив символів.
- Метод sort() сортує масив в алфавітному порядку (сортування відбувається в тому ж масиві, тобто повертається той же масив, але відсортований).

3 завдання.

Завдання полягає у створенні WEB-сторінки з малюнком (у кожного студента повинен бути свій малюнок), що безперервно рухається по сторінці в межах умовного прямокутника (поля), відбиваючись немов більярдна куля від уявних меж цього прямокутника. Приклад зображено на рис.10.2:

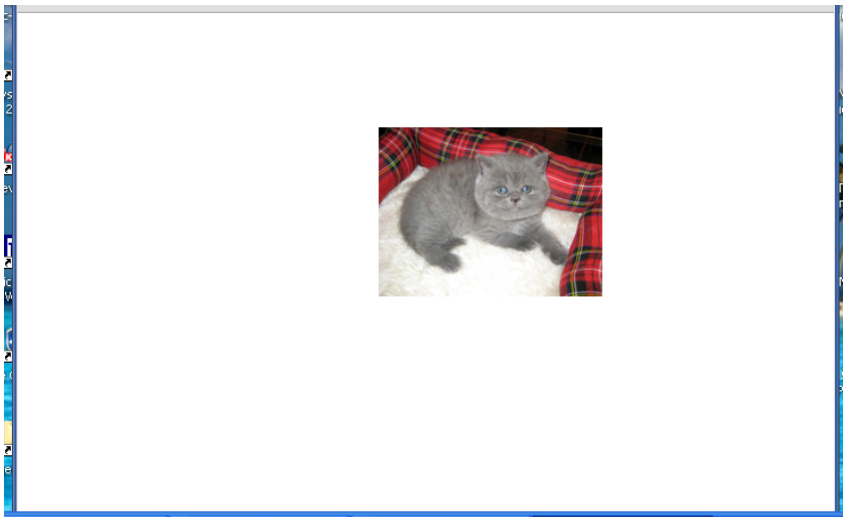

Рис. 10.2. Приклад web-сторінки з малюнком, що рухається

Підказки:

- Нагадаємо з минулого заняття функцію, що створює динаміку на веб-сторінці

  `id=setTimeout("clock_form()",100)`

  де функція `setTimeout("clock_form()",100)` запускає функцію `clock_form()` через 100 мілісекунд (із затримкою 100 мілісекунд). Тобто, якщо всі необхідні нам операції розмістити у функції `clock_form()` і там же розмістити `id=setTimeout("clock_form()",100)`, то все це буде повторюватись в циклі зі затримкою 100 мс.
- Інший варіант створення динаміки – це функція (метод) setInterval, що має синтаксис, аналогічний setTimeout.

  `var timerId = setInterval(func, delay)`

  Призначення аргументів – ті самі. Але, на відміну від `setTimeout`, цей метод запускає на виконання функцію `func` не один раз, а регулярно повторює її через вказаний інтервал часу. Зупинити виконання можна викликом `clearInterval(timerId)`.
- Задати режим абсолютного позиціювання блоку з картинкою можна за допомогою стильового правила `style="position:absolute"`.
- Коли блок з картинкою досягає, наприклад, лівого борту нашого поля і відбивається, то в коді треба змінити напрям горизонтального переміщення на протилежний; аналогічно для верхнього та нижнього бортів.

За результатами виконання роботи в звіті повинні бути представлені:

1. Назва роботи
2. Мета роботи
3. Скрін-шоти сторінок створених сайтів.
4. Висновки

10.3. Питання до самоконтролю

10.3.1. Що таке колекція в JavaScript та чим вона відрізняється від масива? Які Ви знаєте колекції об’єкту Document?

10.3.2. Яким чином можна змінювати стильові властивості об’єктів за допомогою JavaScript?

10.3.3. Яким чином в JavaScript здійснюється передача аргументів у функцію, що працює із змінним числом аргументів? Що таке об'єкт arguments?

10.3.4. Яким чином використовуються події для запуску сценаріїв JavaScript?

11. ВИКОРИСТАННЯ ЗАСОБІВ JQUERY У WEB-ТЕХНОЛОГІЯХ

Мета роботи: Оволодіти основами застосування засобів jQuery з метою реалізації клієнтських сценаріїв для створення динамічних web-сторінок.

11.1. Інформація для самостійної підготовки

Хоча jQuery насправді є лише бібліотекою для сценаріїв JavaScript, вона виявилась настільки вдалою та універсальною, що стала майже стандартною, дуже популярною і немовби окремою технологією у web-програмуванні. Вона дуже спрощує типові задачі при створенні динамічних web-сторінок і заслуговує окремого детального вивчення.

Теоретичні засади рекомендується вивчати з відповідних розділів підручника [2]. А при формулюванні конкретних завдань далі в тексті будуть надані деякі пояснення, що стосуються найскладніших аспектів застосування jQuery.

11.2. Порядок виконання роботи:

Устаткування, матеріали та інструменти

- Персональний комп’ютер зі встановленим браузером та текстовим редактором типу Блокнот або WordPad.
- Кілька файлів малюнків в форматі .jpg.

Порядок виконання роботи

Завдання практикуму складається з 3-х частин, що пов’язані з різними темами застосування jQuery.

1 завдання.

Завдання полягає у наступному.

У даному прикладі є три елементи div, елемент textarea і кнопка button (рис.11.1). Треба зробити так, щоб після того, як користувач введе який-

небудь HTML-код в текстове поле і натисне кнопку, цей HTML-код був вставлений у всі елементи div, які є на сторінці. Треба використати технологію jQuery.

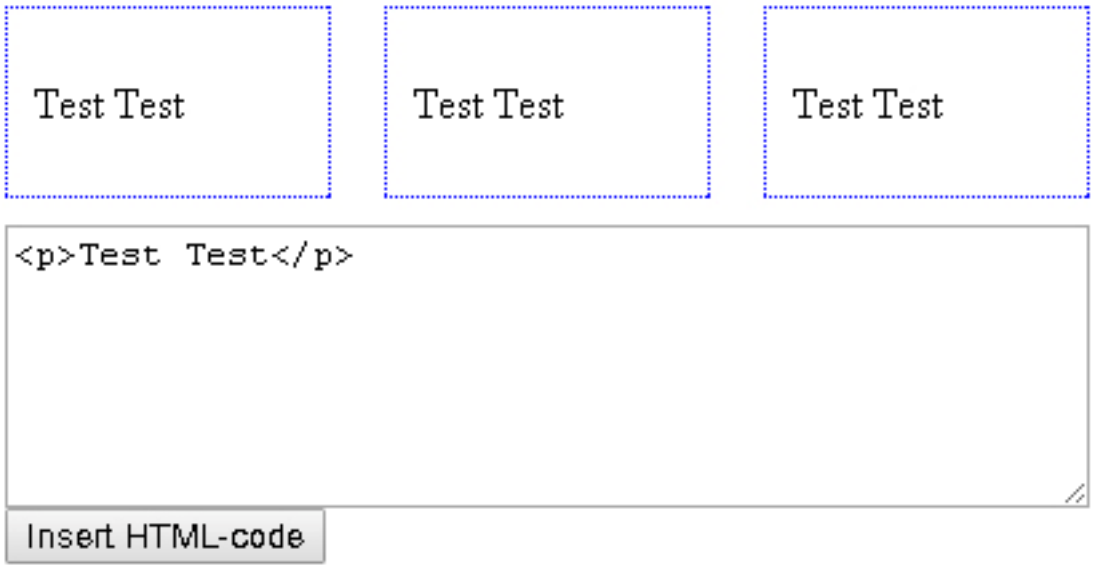

Рис. 11.1. Приклад web-сторінки з трьома елементами div, елементом textarea і кнопкою button

Підказки:

В jQuery метод .html(val) розміщує у всіх елементах набору (який створить функція jQuery або функція $) вміст аргумента val, який повинен бути у вигляді HTML елементу.

Слід пов'язати елемент button з подією click, а в якості обробника події використати функцію, яка спочатку запише у змінну val значення атрибуту value елементу textarea, а потім за допомогою методу .html() вставить цей код у всі елементи div.

Отримати значення атрибуту value елементу textarea можна за допомогою jQuery-метода .attr("value").

2 завдання.

Завдання полягає у наступному.

При відкритті HTML-сторінки на ній існує кнопка з написом „To clone!”.

При натисканні на цю кнопку вона клонується зправа поруч з початковою кнопкою, причому разом зі своїми властивостями, як об’єкт, тобто на цей клон також можна натискати, і він також буде клонуватись (рис.11.2):

Рис. 11.2. Приклад web-сторінки з клонуванням кнопки

Підказки:

В jQuery функція (метод) .clone() клонує jQuery-набір, до якого вона застосовується, а .clone(true) клонує jQuery-набір разом з його властивостями, методами, обробниками подій.

Селектор this вказує на сам об’єкт, в якому знаходиться миша.

Вставити об’єкт після this можна методом .insertAfter(this)

3 завдання.

Завдання полягає у наступному.

Засобами jQuery реалізувати анімацію об’єкта.

Прямокутник повинен рухатись праворуч, потім вниз, ліворуч і знову вверх на своє початкове місце. При цьому він повинен змінювати свій розмір (зменшуватись, а потім збільшуватись) і прозорість (рис.11.3):

Рис. 11.3. Приклад web-сторінки з анімацією

Підказки:

В jQuery функція animate() призначена для створення анімації на основі зміни стилів CSS об'єкту. Синтаксис функції можна представити:

```
animate (параметри [тривалість][вигляд
зміни][callback])
```

Параметри — це CSS властивості. Так, якщо вказати marginTop:’40px’ то це означатиме, що в кінці анімації до об'єкту, що анімується, буде застосовано дану властивість. Початкова точка — 0, або задане значення. Таким чином, всі вписані параметри визначають кінцевий результат анімації. Задавати значення властивостей можна не лише за допомогою px, але і використовуючи знак % або em. Приклад функції з параметрами:

```
$("#go").click(function(){
$("#block").animate({
width: "70%",
opacity: 0.4
}, 1500 );
});
```

У даному прикладі в кінцевому стані об'єкт #block зменшить ширину до 70%, прозорість до 40%. Тривалість анімації — 1500мс, або 1.5 секунди.

Зверніть увагу, що властивості розділяються комами, а не крапкою з комою, як в CSS! Після останньої властивості знаки не ставляться.

„Вигляд зміни” (в синтаксисі) визначає динаміку виконання анімації. Доступний два параметри: ‘linear’ — рівномірна анімація параметрів і ‘swing’ — прискорена. За умовчанням застосовується анімація ‘swing’.

Слід зауважити наступне. Всі властивості, які були раніше прописані, анімувалися одночасно. Якщо необхідно анімувати властивості послідовно, потрібно розбити їх на декілька блоків .animate().

Зверніть увагу, що після .animate() не ставиться знаків крапка з комою! Подібний запис називається «Чергою (Queue)».

За результатами виконання роботи в звіті повинні бути представлені:

1. Назва роботи
2. Мета роботи
3. Скрін-шоти сторінок створених сайтів.
4. Висновки

11.3. Питання до самоконтролю

11.3.1. Описати основні ідеї і засоби jQuery.

11.3.2. Що таке „набір елементів” в jQuery? Яким чином в jQuery використовуються селектори, схожі з селекторами CSS?

11.3.3. Які переваги використання jQuery?

11.3.4. Яким чином можна підключити бібліотеку jQuery до web-сторінки?

.

12. ДИНАМІЧНЕ УПРАВЛІННЯ У WEB-ДОКУМЕНТАХ ЗАСОБАМИ JAVASCRIPT ТА JQUERY

Мета роботи: Навчитися застосовувати можливості мови JavaScript та бібліотеки jQuery для динамічної зміни вигляду та поведінки HTML-елементів у відповідь на дії користувача.

12.1. Інформація для самостійної підготовки

За допомогою мови JavaScript можна динамічно керувати зовнішнім виглядом та поведінкою усіх HTML-елементів на сторінці. Об'єкти JavaScript, які представляють HTML-елементи, мають властивості, що відповідають атрибутам HTML-тега (часто назви цих властивостей співпадають з назвами відповідних атрибутів). Наприклад, форма має атрибути action, method та target.

Існують і властивості action, method та target у JavaScript-об'єкта, який представляє форму. Оскільки JavaScript-код можна викликати у відповідь на якусь дію користувача, то вже після завантаження сторінки можливо поміняти значення атрибутів форми (а також всіх інших елементів сторінки). Форма має метод submit(). Виклик цього методу призводить до відправлення форми на сервер.

Розглянемо використовувані при виконанні даної роботи властивості, за допомогою яких JavaScript-код керує виглядом та поведінкою HTML-елементів.

Елементи форми мають атрибут disabled, який визначає, чи є елемент форми активним. У JavaScript для кожного елемента форми визначена властивість disabled, доступна як для читання, так і для запису. При disabled = false елемент форми є активний, а при disabled = true – неактивний.

Приклад. Нехай існує текстове поле з атрибутом id, рівним "text1". Тоді наступний рядок JavaScript-коду робить це поле неактивним:

document.getElementById("text1").disabled = true;

Елементи форми, що оголошуються за допомогою тега <INPUT>, володіють атрибутом value. Для текстового поля цей атрибут задає текст, введений у поле, для кнопки – текст напису на ній. Для елементів форми, що мають атрибут value, у JavaScript визначена властивість value (доступна для читання та для запису). Властивість value є типу "текстовий рядок".

За допомогою мови JavaScript можна динамічно управляти властивостями стилів усіх HTML-елементів. Кожен об'єкт, що представляє HTML-елемент сторінки, володіє властивістю style. Об'єкт style, у свою чергу, має величезний набір властивостей, які визначають зовнішній вигляд HTML-елемента.

Приклад. Якщо на сторінці існує текстове поле з атрибутом id, рівним "text1", то наступний фрагмент JavaScript-коду задає цьому текстовому полю наступні властивості: колір літер білий, розмір шрифту 8 pt, колір фону чорний, рамка текстового поля відсутня, висота текстового поля становить 24 пікселі:

```
var textStyle = document.getElementById("text1").style;
textStyle.border = "none";
textStyle.fontSize = "8pt";
textStyle.color = "#FFFFFF";
textStyle.backgroundColor = "#000000";
```

Інколи потрібно приховати частину HTML-елементів сторінки. Для цього використовують одну з двох властивостей об'єкта style: visibility або ж display. Щоб приховати об'єкт, слід властивості visibility задати значення "hidden":

document.getElementById("UN").style.visibility = "hidden";

Щоб зробити HTML-елемент видимим, властивості visibility слід присвоїти значення "visible":

document.getElementById("UN").style.visibility = "visible";

Інший спосіб приховати об'єкт – присвоїти властивості display значення "none" (відповідно, щоб показати об'єкт, прихований таким чином, слід властивості display задати значення "block"):

document.getElementById("UN").style.display = "none";

document.getElementById("UN").style.display = "block";

Використання властивостей visibility та display дає різний ефект. Об'єкти, приховані за допомогою властивості visibility, стають невидимими, проте продовжують займати місце на сторінці. Натомість об'єкти, приховані за допомогою властивості display, не лише стають невидимими, але й звільняють місце, яке вони займали, будучи видимими.

Об'єкт JavaScript, що представляє елемент форми "список", має властивість options. Дана властивість є масивом. Масив options представляє набір елементів списку. У options є властивість length, яка містить кількість елементів масиву (тобто, кількість елементів у випадаючому списку). Елементи масиву нумеруються починаючи з нуля.

До елементів масиву слід звертатися за індексом. Наприклад, options[0] представляє перший елемент випадаючого списку, options[1] – другий і т.д. Кожен елемент масиву options має властивості text, value та selected. Властивість text представляє текст елемента списку, властивість value – значення атрибута value (індекс масиву) елемента списку. Властивість selected містить стан елемента списку – вибраний або невибраний.

Властивість selectedIndex містить індекс елемента, що є вибраним у даний момент (для списку з можливістю вибору лише одного елемента).

Для динамічного додавання та вилучення елементів випадаючого списку застосовуються методи add та remove відповідно. Метод add приймає два параметри. Перший параметр задає елемент, який слід додати до списку. Другий параметр вказує індекс, який матиме доданий елемент. Метод remove приймає лише один параметр – індекс елемента списку, який потрібно видалити.

Для прапорців та перемикачів визначена властивість checked, яка дорівнює true, якщо прапорець (або перемикач) ввімкнений і false – в іншому випадку.

Розглянемо завдання із застосування описаних вище властивостей об’єктів.

Завдання 1. Нехай HTML-сторінка містить текстове поле для вводу імені користувача, поле для вводу паролю та кнопку з написом “Send”. Нехай для текстового поля id=“UN”, для поля вводу паролю id=“pwd”, а для кнопки id=“send”. Слід зробити кнопку активною лише у тому випадку, якщо текстове поле і поле для вводу паролю обидва є непустими, і неактивною – в іншому випадку.

Оголосимо наступну функцію:

```
function checkFields()
{
if(document.getElementById("UN").value==""                    ||
document.getElementById("pwd").value == "")
   document.getElementById("send").disabled = true;
else
      document.getElementById("send").disabled = false;
}
```

Цю функцію слід розмістити у контейнері <script></script>, який, у свою чергу, розміщений у контейнері <head></head>.

Функцію checkFields доцільно викликати як обробник події onkeypress як текстового поля, так і поля вводу паролю.

Завдання 2. Нехай у HTML-сторінці оголошено такий випадаючий список:

```
<select id="city">
<option>Виберіть місто</option>
<option>Львів</option>
<option>Київ</option>
<option>Одеса</option>
<option>Харків</option>
</select>
```

Напишемо функцію, яка блокує кнопку “Send”, якщо зі списку нічого не вибрано або якщо вибрано елемент списку «Виберіть місто»:

```
function handleSelect()
{ var sel = document.getElementById("city");
  if(sel.selectedIndex == 0 || sel.selectedIndex == -1)
        document.getElementById("send").disabled = true;
  else
        document.getElementById("send").disabled = false;
}
```

Оскільки елемент списку “Виберіть місто” оголошений першим елементом списку, то у масиві options цей елемент матиме індекс 0, індекс "-1" відповідає випадку, коли нічого не вибрано.

Функцію handleSelect доцільно викликати у відповідь на зміну вибраного елемента списку, тобто, як реакцію на подію onchange випадаючого списку.

Завдання 3. Організуємо наступну функціональність: додамо до сторінки оголошення текстового поля з id= “newElement” та кнопки з текстом “Add element”. При натисненні кнопки “Add element” будемо додавати в кінець випадаючого списку новий елемент з текстом, введеним у текстове поле (за умови, що у текстове поле взагалі введено якийсь текст).

Реалізуємо таку функцію:

```
function addToList()
{
  if(document.getElementById("newElement").value!="") /*Якщо вміст текстового поля не пустий*/
    {
        var sel = document.getElementById("city");
        var newItem = new Option();/*Оголошення нового об’єкта типу «Елемент списку»*/
        newItem.text=document.getElementById("newElement").value; /*У властивість text нового елемента списку записуємо вміст текстового поля*/
        newItem.value=sel.options.length-1; /*У властивість value нового елемента списку записуємо його майбутній порядковий номер у списку */
        sel.add(newItem, sel.length); /*До списку додаємо визначений елемент на останню позицію */
}
}
```

Функцію addToList() слід зробити обробником події onclick кнопки “Add element”.

Завдання 4. Додамо у сторінку кнопку “Видалити вибраний елемент”. При натисненні цієї кнопки повинен видалятися вибраний елемент списку, а вибраним має стати перший елемент списку.

Напишемо таку функцію:

```
function deleteItem()
{
  var sel = document.getElementById("city"); /* у змінну sel запишемо посилання на випадаючий список */
  sel.remove(sel.selectedIndex); /* Видаляємо елемент, що є вибраним у списку на даний момент */
  sel.selectedIndex = 0; /* Робимо вибраним елемент з індексом 0*/
}
```

Дану функцію зробимо обробником події onclick кнопки “Видалити вибраний елемент”.

Об'єкт event

Об'єкт події (event) – це об'єкт, що містить дані про подію, яка сталася. Передається у всі обробники подій, які були встановлені засобами бібліотеки jQuery.

Наступні поля гарантовано присутні в кожному об'єкті події (хоча деякі з них можуть мати значення undefined): altKey, attrChange, attrName, bubbles, button, cancelable, charCode, clientX, clientY, ctrlKey, currentTarget, data, detail, eventPhase, fromElement, handler, keyCode, layerX, layerY, metaKey, namespace (лише з jQuery-1.4.3 і старіше), newValue, offsetX, offsetY, originalTarget, pageX, pageY, prevValue, relatedNode, relatedTarget, screenX, screenY, shiftKey, srcElement, target, toElement, view, wheelDelta, which.

Об'єкти event, які надає безпосередньо javascript, можуть відрізнятися від браузеру до браузеру. Тому jQuery може змінити деякі поля для

забезпечення кросбраузерности перед передачею об'єкту event обробникові.

jQuery-конструкція

```
$(window).keydown(function(event){
…….. тіло функції ……
});
```

описує реакцію системи на подію натискання клавіші. Як видно, тут keydown – це мотод jQuery-об’єкта, що створений по селектору window, а функції-обробнику події передається об’єкт event.

Властивість об’єкта події event.keyCode містить код клавіші, яка була натиснута.

12.2. Порядок виконання роботи:

Устаткування, матеріали та інструменти

- Персональний комп’ютер зі встановленим браузером та текстовим редактором типу Блокнот або WordPad.
- Кілька файлів малюнків в форматі .jpg.

Порядок виконання роботи

Завдання практикуму складається з 4-х частин.

1. Реалізуйте описані вище приклади 1-4.

2. Створіть поле для вводу паролю та кнопку. При натисненні на кнопку, якщо поле для вводу паролю пусте, то воно повинно набувати таких властивостей: колір фону – червоний.

3. Створіть на сторінці таблицю з трьох рядків та двох стовпців і кнопку (під таблицею). При натисненні на кнопку перший і третій рядки таблиці повинні приховуватися, причому перший рядок повинен перестати займати місце на екрані, а третій – зберегти своє місце на екрані.

4. За допомогою засобів jQuery створити програмку, що при натисканні якоїсь клавіші (тобто за подією keydown) повідомляє користувачеві код натиснутої клавіші.

За результатами виконання роботи в звіті повинні бути представлені:

1. Назва роботи
2. Мета роботи
3. Скрін-шоти сторінок створених сайтів.
4. Висновки

12.3. Питання до самоконтролю

12.3.1. Що таке „подія” в DOM?

12.3.2. Яким чином програмується засобами jQuery реакція елементів web-сторінки на події?

12.3.3. Порівняти програмування реакції на події засобами JavaScript та jQuery

12.3.4. Чому event можна вважати об’єктом і як використовуються його властивості?

13. ВСТАНОВЛЕННЯ, НАЛАШТУВАННЯ ТА ТЕСТУВАННЯ СЕРВЕРНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ APACHE, PHP, MYSQL

Мета роботи: Набути здатностей і досвіду роботи з сервером. Підготовути робоче місце для наступних завдань.

13.1. Інформація для самостійної підготовки

Хоча на перших практичних заняттях була засвоєна технологія „Денвер" спрощеного створення локального сервера – середовища для розробки сайтів, доцільно все ж навчитись працювати з повноцінним сервером Apache і необхідним оточенням, знати їх параметри та процедури налаштування. При інсталяції „Денвер" все робилось автоматично, і жодних знань щодо налаштувань середовища не вимагалось, проте це може знадобитись у реальній практиці.

Тому розглянемо технологію встановлення і налаштування повноцінного серверного середовища Apache, PHP, MySQL.

Встановлення Apache

Дистрибутив сервера можна викачати з офіційного сайту Apache: http://www.apache.org, вибравши найостаннішу версію сервера для платформи Windows.

Далі наводяться інструкції по установці і конфігурації.

Етап перший: установка

Цей процес типовий для установки будь-якого програмного забезпечення і пояснень не вимагає. Тобто треба запустити файл дистрибутиву Apache та виконувати дії, щодо яких буде з'являтися запит.

Етап другий: налаштування файлу конфігурації Apache

На цьому етапі потрібно визначитися з каталогом, в якому зберігатимуться сайти. За умовчанням Apache використовує для цього

C:\Program Files\Apache Group\Apache\htdocs, де відразу після установки можна знайти документацію. Треба створити в цьому каталозі підкаталоги, в яких зберігатимуться наші сайти.

Тепер треба відкрити в Блокноті файл конфігурації httpd.conf, який розташований в підкаталозі conf каталога Apache (C:\Program Files\Apache Group\Apache).

Httpd.conf — єдиний файл, який потрібно налаштувати. Треба знайти і змінити в ньому деякі рядки. Слід зауважити, що у файлі кожен параметр супроводжується декількома рядками коментарів.

Спершу налаштовуємо параметри для головного хоста Apache — localhost, а також параметри за умовчанням, які будуть успадковані всіма рештою віртуальними хостами, якщо ми коли-небудь захочемо їх створити.

Задаємо значення параметра ServerName таким чином:

ServerName localhost

При цьому слід не забути розкрити коментар для поля ServerName, тобто прибрати символ # перед цим параметром (встановлений за умовчанням), оскільки все, що йде після цього символу і до кінця рядка, Apache ігнорує.

У полі DocumentRoot слід вказати той каталог, в якому будуть розміщені наші HTML-файли, залишаємо шлях, вказаний за умовчанням (C:/Program Files/Apache Group/Apache/htdocs).

Далі слід ініціалізувати параметр DirectoryIndex так:

DirectoryIndex index.htm index.html

Це так звані файли індексу, які автоматично повертаються сервером при зверненні до якого-небудь каталога, якщо не вказано ім'я HTML-файлу в цьому каталозі. В принципі, можна додати сюди і інші імена, наприклад,

index.php, і так далі. Проте, додаткові налаштування все ж краще робити у файлах .htaccess для кожного сайту окремо.

Далі слід знайти наступний параметр:

ScriptAlias /cgi-bin/ C:/Program Files/Apache Group/Apache/cgi-bin/

Це буде той каталог, в якому повинні розташовуватися CGI-сценарії. Подібний параметр говорить Apache, що, якщо буде вказаний шлях виду http://localhost/cgi-bin, то насправді слід звернутися до каталога C:/Program Files/Apache Group/Apache/cgi-bin.

Далі слід знайти і налаштувати наступний параметр:

AddHandler cgi-script .bat .exe .cgi

Він говорить Apache про те, що файли з розширеннями exe, bat і cgi треба розглядати як CGI-модулі.

І останнє — слід розкоментувати і налаштувати наступні параметри:

AddType text/html .shtml

AddHandler server-parsed .shtml .html .htm

Цим ми примушуємо Apache обробляти файли з вказаними розширеннями процесором SSI.

Тепер слід зберегти зміни і закрити Блокнот.

Етап третій: тестування Apache

Apache налаштований і повинен вже працювати. Для запуску сервера слід натиснути кнопку Пуск, потім вибрати Програми, Apache HTTP Server і Start Apache in Consol, при цьому спливе вікно, дуже схоже на Сеанс MSDOS, і нічого більше не станеться. Не треба його закривати і чіпати до кінця роботи з Apache.

Якщо вікно відкривається і тут же закривається, це означає, що ми допустили якусь помилку у файлі httpd.conf. В цьому випадку доведеться шукати неточність.

Перевірка html.

Наберемо в браузері: http://localhost/. Повинен завантажитися файл index.html з каталога C:/Program Files/Apache Group/Apache/htdocs.

У підкаталозі каталога C:/Program Files/Apache Group/Apache/htdocs, який Ви створили для зберігання свого сайту, створемо файл index.html з будь-яким текстовим наповненням. Якщо тепер запустити браузер і набрати:

http://localhost/Назва_нашого_підкаталогу/index.html

або просто

http://localhost/Назва_нашего_підкаталогу/

повинен завантажитися наш файл.

Встановлення PHP

Завантажити PHP можна з офіційного сайту PHP http://www.php.net з секції Downloads (два файли).

Процес інсталяції також типовий. Тобто треба запустити exe-файл та виконувати відповідні дії згідно запитів інсталяційного пакету.

В текстових полях рекомендується залишати значення за умовчанням.

Щодо сервера, на який буде настроєний PHP, слід обрати Apache.

На цьому етапі PHP можна вважати вже майже встановленим. Залишилося тільки налаштувати Apache, щоб він міг розпізнати PHP-сценарії, а також підключити додаткові модулі, які містяться в завантаженому нами zip-архіві.

Налаштування Apache для роботи з PHP

1. Слід відкрити в Блокноті файл конфігурації Apache httpd.conf, що знаходиться в каталозі C:\Program Files\Apache Group\Apache\conf.

2. Знайти в тексті файлу такий закоментований рядок:

#AddType application/x-httpd-php php

3. Розкрити коментар:

AddType application/x-httpd-php php

Таким чином, ми привласнили всім файлам з розширенням php тип application/x-httpd-php.

4. Відразу ж після цього рядка слід додати такі налаштування:

ScriptAlias /_php/ "C:/PHP/"

Action application/x-httpd-php "/_php/php.exe"

Цим ми, по-перше, створюємо синонім _php для каталога з процесором PHP, щоб Apache міг дістати до нього доступ, а по-друге, пов'язуємо всі файли типа application/x-httpd-php з обробником php.exe.

Префікс до рядка "_php" вибраний з такого розрахунку, щоб він в майбутньому не конфліктував з іменами створюваних на хості каталогів.

5. Зберегти зміни у файлі конфігурації, зупинити Apache, якщо він був до цього запущений (пункт Пуск > Програми > Apache Web Server > Management > Stop Apache), і стартувати сервер знову. Якщо Apache не запускається (його вікно відкривається і тут же закривається), значить, ми десь допустили синтаксичну помилку.

Тестування PHP

Аби переконаємося, що PHP-сценарії працюють, створимо в каталозі C:/Program Files/Apache Group/Apache/htdocs файл test.php з наступним змістом:

```
<? php
 echo "It works! \n";
 phpinfo();
 ?>
```

Якщо тепер набрати в браузері: http://localhost/test.php, повинна відображуватися сторінка з різноманітною інформацією про PHP, яка генерується функцією phpinfo().

Встановлення MySQL

1. Дистрибутив MySQL можна завантажити з офіційного сайту MYSQL (http://www.mysql.com, розділ Downloads). Дистрибутивом є zip-архив, який потрібно розвернути в будь-який зручний каталог.

2. Запустити setup.exe з розархівованого дистрибутива.

В текстових полях рекомендується залишати значення за умовчанням.

3. Для того, щоб активізувати MySQL-сервер, слід запустити виконуваний файл C:\MySQL\bin\mysqld.exe. Можете створити для нього ярлик, проте, оскільки зазвичай MySQL працює "у зв'язці" з Apache, буде логічно створити командний файл, який стартуватиме і Apache, і MySQL. Назвемо його server.bat. Ось зміст цього файлу:

```
@echo off
 "C:\MySQL\bin\mysqld"
 start /m "C:\Program Files\Apache Group\Apache\Apache"
```

Для нових операційних систем, проте, зручніше використовувати дещо інші команди (інакше в цих системах вікно процесу MySQL буде постійне видно на екрані, що небажано):

```
@echo off
 start C:\MySQL\bin\mysqld-nt --standalone
 C:\Progra~1\Apache~1\Apache\Apache -k start
```

Перед вимкненням або перезавантаженням комп'ютера потрібно завершувати роботу Apache і MySQL. Для цього найзручніше створити наступний bat-файл з ім'ям, наприклад, shutdown.bat.

```
@echo off
 "C:\Program Files\Apache Group\Apache\Apache" -k shutdown
 "C:\MySQL\bin\mysqladmin" -u root shutdown
```

Тестування MySQL

Для перевірки MySQL слід спершу запустити наш файл server.bat, щоб активізувати сервер. Далі створити наступний PHP-сценарій з ім'ям mysql.php у каталозі C:/Program Files/Apache Group/Apache/htdocs.

```
<? php
 define("DBName","test");
 define("HostName","localhost");
 define("UserName","root");
 define("Password""");
 if(!mysql_connect(HostName,UserName,Password))
 { echo "Не можу з'єднатися з базою ".DBName."!
";
 echo mysql_error();
 exit;
 }
 mysql_select_db(DBName);
 // Створюємо таблицю t. Якщо така таблиця вже є
 // повідомлення про помилку буде пригнічено
 @mysql_query("create table t(id int, a text)");
 // Вставляємо в таблицю 10 записів
```

```
for($i=0; $i<10; $i++)
{ $id=time();
mysql_query("insert into t(id, a) values($id, 'No.$i!')");
}
$r=mysql_query("select * from t");
for($i=0; $i
{ $f=mysql_fetch_array($r);
echo "$f[id]-> $f[a]
\n";
}
?>
```

Тепер слід набрати в браузері:

http://localhost/mysql.php

Якщо все конфігуровано правильно, отримаємо декілька рядків виводу в браузері без повідомлень про помилки. При кожному запуску в таблицю t додаються нові рядки, так що з кожним натисненням кнопки Відновити в браузері об'єм таблиці буде збільшуватися.

У тексті програми присутні константи DBName, HostName, UserName і Password.

DBName повинен містити ім'я бази даних (у нашому випадку це test — база даних, яка створюється MySQL за умовчанням). HostName — завжди localhost, адже ми працюємо на локальному комп'ютері. У макросі UserName найпростіше підставляти root, який є власником всіх таблиць.

При установці MySQL користувачеві root не призначається пароль, так що константа Password дорівнює порожньому рядку.

13.2. Порядок виконання роботи:

Устаткування, матеріали та інструменти

- Персональний комп'ютер зі встановленим браузером та текстовим редактором типу Блокнот або WordPad.

Порядок виконання роботи

Завдання полягає у інсталяції серверного середовища Apache, PHP, MySQL на локальному комп'ютері та перевірці його функціонування шляхом виконання вищезазначених тестів.

За результатами виконання роботи в звіті повинні бути представлені:

1. Назва роботи
2. Мета роботи
3. Скрін-шоти сторінок з результатами виконання тестів..
4. Висновки

13.3. Питання до самоконтролю

13.3.1. Описати зміст окремих налаштувань сервера Apache і середовища.

13.3.2. Порівняти процеси інсталяції пакету „Денвер" і повноцінного сервера Apache.

13.3.3. У чому полягає гнучкість повноцінного сервера Apache у порівнянні з пакетом „Денвер"?

14. СЕРВЕРНІ СЦЕНАРІЇ. МОВА PHP

Мета роботи: Засвоїти основні методичні та технологічні засади використання PHP при розробці серверних сценаріїв.

14.1. Інформація для самостійної підготовки

Технологія мови PHP для розробки серверних сценаріїв в базовому варіанті зараз широко відома і використовується майже в кожному сайті. Теоретичі засади надаються в лекційному циклі. При виникненні потреби нагадувань щодо якихось елементів технологій рекомендується звернутись до лекційного матеріалу або відповідних літературних джерел, наприклад [1]. А на даному практичному занятті зосередимось на кількох основних темах, пов'язаних з використанням PHP, маючи на увазі загалом володіння слухачами базовими технологіями. При формулюванні конкретних завдань далі в тексті будуть надані деякі пояснення, що стосуються найскладніших аспектів застосування PHP.

Слід нагадати, що для виконання наступних завдань, пов'язаних з використанням серверних сценаріїв, необхідно мати налаштований локальний веб-сервер, процесу інсталяції котрого були присвячені попередні практичні заняття № 1 та №13. Слід згадати, наприклад, з першої лабораторної роботи технологію ДЕНВЕР створення локального сервера, причому пом'ятати, що серверні сценарії PHP треба розміщати в папці HOME цього локального сервера, як нові сайти з певними назвами.

14.2. Порядок виконання роботи:

Устаткування, матеріали та інструменти

- Персональний комп'ютер зі встановленим браузером та текстовим редактором типу Блокнот або WordPad.
- Кілька файлів малюнків в форматі .jpg.

Порядок виконання роботи

Завдання практикуму складається з 4-х частин, що пов’язані з різними темами застосування PHP.

1 завдання.

Завдання полягає в наступному.

При звертанні до сайту повинно з’явитись поле форми із запитом „Введіть ваше ім'я”. Після введення імені (наприклад, Сергій) з’являється напис „Hello, Сергій”.

Серверний сценарій реалізувати засобами PHP.

Підказки:

1) При відправленні форми на сервер введене користувачем значення поля, яке має ім’я “name”, зберігається на сервері у суперглобальному масиві $_GET, а саме в елементі $_GET['name'] цього масиву.

2) Серверний сценарій повинен спочатку перевірити чи не пусте значення цього елемента масиву: якщо воно пусте, то вивести форму із запитом „Введіть ваше ім'я”, а якщо не пусте, то вивести це значення у вигляді рядка, наприклад, „Hello, Сергій”.

3) Апостроф у рядку „Введіть ваше ім'я” буде викликати синтаксичну помилку PHP-інтерпретатора. Щоб цього уникнути, треба поставити перед апострофом обернений „слеш”.

2 завдання.

Завдання полягає в наступному.

В сценарії PHP сайту треба створити асоціативний масив, у якому індекс кожного елемента – це назва якогось фрукта, а значення – це його ціна у гривнях.

При звертанні до цього сайту в браузері виводиться оформлена у вигляді таблиці загальна інформація про всі фрукти та їх ціни (рис.14.1).

Фрукт	Ціна
банани	30 грн
ананаси	50 грн
яблука	12 грн
апельсини	35 грн

Рис. 14.1. Приклад web-сторінки з інформацією про ціни

Підказки:

1) Створення асоціативного масиву в PHP реалізується, наприклад, таким чином:

$fruits["банани"] = "30 грн";

2) Доступ до всіх елементів асоціативного масиву, тобто одержання всіх значень ключів елементів та самих елементів, реалізується, наприклад, таким чином:

foreach ($fruits as $key => $value) {…тіло функції…};

3) При оформленні границь таблиці нам знадобиться задавати значення параметрів тегів, і вони повинні бути в лапках, а html-сторінка з PHP-сценарію виводиться за допомогою оператора echo, після якого іде строка тексту також у лапках. Тут, щоб не було плутанини для інтерпретатора PHP, слід згадати, що існує два типи лапок: звичайні і типу апостроф. Слід правильно використати ці різновиди у операторі echo.

3 завдання.

Завдання полягає в наступному.

При завантаженні сайту повинен з’являтися календар, тобто приблизно така інформація (рис.14.2)

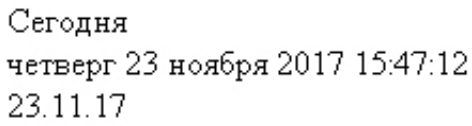

Рис. 14.2. Приклад web-сторінки з календарем

Календар повинен бути створений засобами PHP.

Підказки:

Робота з датами відбувається із застосуванням функції date(). Її синтаксис

date(”Строка формату дати/часу”);

Ця функція повертає дату і час у вигляді, що визначається значенням аргументу ”Строка формату дати/часу”.

Ці значення можуть бути наступними;

• U — кількість секунд, що пройшли з 1 січня 1970 г.;

• Y — рік з 4 цифр;

• y — рік з 2 цифр;

• z — день з початку року (от 0 до 365);

• F — назва місяця по-англійськи;

• m — номер місяця з попереднім нулем (від "01" до "12");

• n — номер місяця без попереднього нуля (от "1" до "12");

• M — аббревіатура місяця з 3-х букв по-англійськи;

• d — номер дня з попереднім нулем (от "01" до "31");

• j — номер дня без попереднього нуля (от "1" до "31");

• l — назва дня тижня по-англійськи;

• w — номер дня тижня (0 — для неділі, 6 — для суботи);

• D — аббревіатура дня тижня з 3-х букв по-англійськи;

• A — "AM" (до обіду) або "PM" (після обіду);

• a — "am" (до обіду) или "pm" (після обіду);

• H — години у 24-годинному форматі (від "00" до "23");

• h — години у 12-годинному форматі (від "01" до "12");

• i — хвилини (від "00" до "59");

• s — секунди (від "00" до "59")

4 завдання.

Завдання полягає в наступному.

При завантаженні сайту повино з’являтися поле для вибору якоїсь дати. Після вибору і підтвердження виводиться інформація про кількість днів, що пройшли від вибраної дати до поточної.

Підказки:

Робота з поточною датою аналогічна попередньому завданню.

2) Обирати дату, що нас цікавить, рекомендується через поле форми <input type="date">.

3) Перетворити дату з текстового формату (який сценарій одержить через форму) у формат Timeshtamp (тобто у кількість секунд, що пройшло від 1 січня 1970р.) можна за допомогою PHP-функції strtotime().

За результатами виконання роботи в звіті повинні бути представлені:

1. Назва роботи

2. Мета роботи

3. Лістинги сценаріїв та скрін-шоти сторінок результатів їх роботи.

4. Висновки

14.3. Питання до самоконтролю

14.3.1. Як відбувається обробка запиту до сторінки, яка містить сценарій PHP?

14.3.2. Назвіть вбудовані типи даних PHP.

14.3.3. У чому особливість масивів у PHP?

14.3.4. Охарактеризуйте основні функції PHP для роботи з масивами.

15. РОЗШИРЕНІ МОЖЛИВОСТІ МОВИ PHP ПРИ СТВОРЕННІ СЕРВЕРНИХ СЦЕНАРІЇВ

Мета роботи: Удосконалити володіння основними методичними та технологічними засадами використання PHP при розробці серверних сценаріїв.

15.1. Інформація для самостійної підготовки

Практичне заняття присвячене удосконаленню вмінь щодо застосування мови PHP для розробки серверних сценаріїв. Теоретичі засади надаються в лекційному циклі. При виникненні потреби нагадувань щодо якихось елементів технологій рекомендується звернутись до лекційного матеріалу або відповідних літературних джерел, наприклад [1]. А на даному практичному занятті зосередимось на кількох ускладнених темах, пов'язаних з використанням PHP, маючи на увазі загалом володіння слухачами базовими технологіями. При формулюванні конкретних завдань далі в тексті будуть надані деякі пояснення, що стосуються найскладніших аспектів застосування PHP.

Слід знову нагадати, що для виконання наступних завдань, пов'язаних з використанням серверних сценаріїв, необхідно мати налаштований локальний веб-сервер, процесу інсталяції котрого були присвячені попередні практичні заняття № 1 та №13. Слід згадати, наприклад, з першої лабораторної роботи технологію ДЕНВЕР створення локального сервера, причому пом'ятати, що серверні сценарії PHP треба розміщати в папці HOME цього локального сервера, як нові сайти з певними назвами.

15.2. Порядок виконання роботи:

Устаткування, матеріали та інструменти

- Персональний комп'ютер зі встановленим браузером та текстовим редактором типу Блокнот або WordPad.

- Кілька файлів малюнків в форматі .jpg.

Порядок виконання роботи

Завдання практикуму складається з 3-х частин, що пов'язані з різними темами застосування PHP.

1 завдання.

Завдання полягає в наступному.

php-сценарій повинен перебрати всі цілі числа від 1 до 1000 та перевірити кожне з них, чи не просте це число (простим називається число, що не має інших дільників, крім одиниці і самого себе). Всі числа від 1 до 1000 виводяться на екран браузера, причому прості числа зафарбовуються червоним кольором (див. рис.15.1).

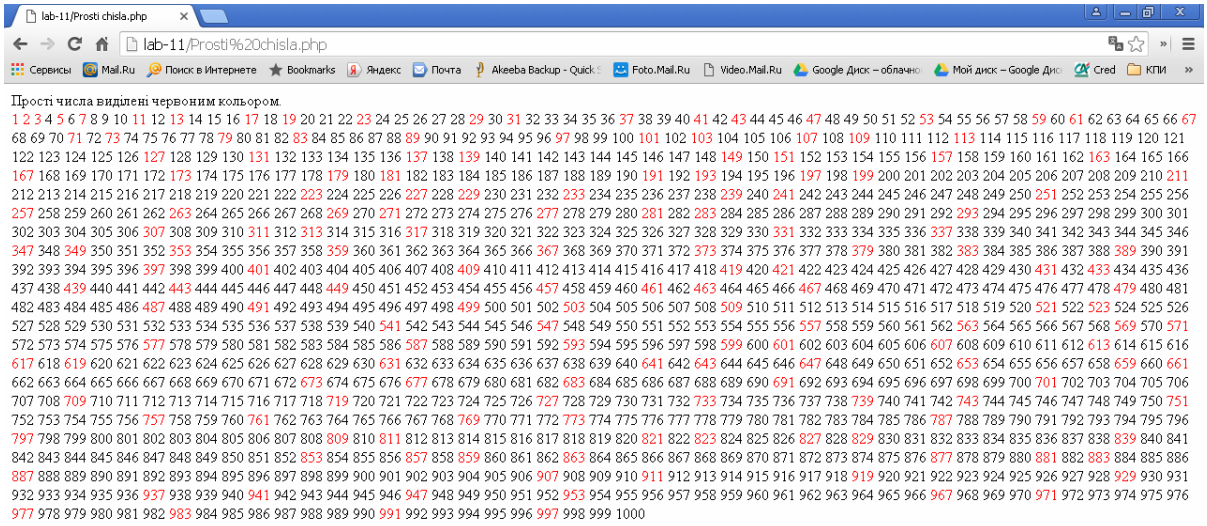

Рис. 15.1. Приклад – таблиця простих чисел

Підказки:

- Тут треба використати два цикли, перший з яких перебирає всі числа від 1 до 1000, а другий шукає всі дільники цього числа.

- Дільники числа N шукаються перебором всіх цілих чисел від 2 до N-1 та перевіркою для кожного з претендентів $i умови ($n % $i)==0, тобто умови ділення без залишку числа $n на число $i.

- При відсутності цілих дільників відповідне число зафарбовується червоним кольором із застосуванням відповідної властивості CSS.

2 завдання.

Завдання полягає в наступному.

php-сценарій виводить поле форми із запитом „Введіть вашу mail” (рис.15.2)

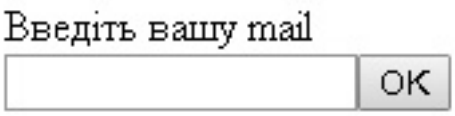

Рис. 15.2. Приклад – запит в полі форми

Користувач вводить свою адресу електронної пошти. Якщо він помиляється з форматом адреси (тобто такої адреси не може бути за синтаксисом), то виводиться наступна інформація (рис.15.3)

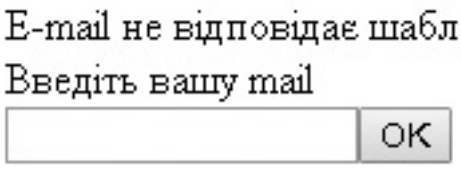

Рис. 15.3. Інформація щодо помилки при введенні адреси

Якщо ж mail-адреса відповідає шаблону, то виводиться наступна інформація (наприклад для адреси a@i.ua) (рис.15.4)

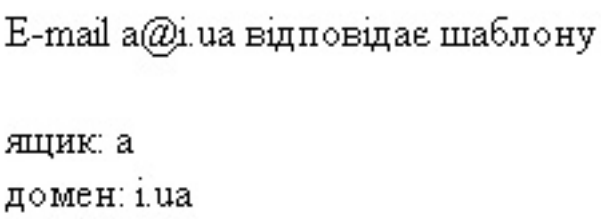

Рис. 15.4. Інформація щодо правильного введення адреси

Тут в другому та третьому рядку – введена mail-адреса розділена на складові частини: ім’я поштового ящика та домену.

Підказки:

- Тут використовується технологія „регулярних виразів” для аналізу строкових змінних. Трохи теорії:

php-функція preg_match() шукає перший збіг з шаблоном в заданому рядку.

Функція має наступний формат:

preg_match(<Шаблон>, <Рядок> [<Масив збігів>] [<Прапор>]

[<Зсув від початку рядка>]);

Функція повертає 0, якщо збіг не знайдений, і 1 в разі відповідності шаблону. Якщо вказано параметр <Масив збігів>, то перший елемент масиву міститиме фрагмент, повністю відповідний шаблону, а інші елементи масиву – це фрагменти, що відповідають частинам шаблону, які розміщені в круглих дужках.

Зрозуміло, що при побудові шаблону для виконання даного завдання треба окремі його частини взяти в круглі дужки, щоб отримати масив необхідних нам частин mail-адреси (тобто назви поштового ящика та назви домену).

Правила побудови шаблону для технології „регулярних виразів”.

Шаблон є рядком, розміщеним у лапках або апострофах, де між двома обмежувачами вказується регулярний вираз. За останнім обмежувачем може бути вказаний модифікатор. У якості обмежувача можуть слугувати однакові символи або парні дужки. Наприклад:

'/<Регулярний вираз>/[<Модифікатор>]'

'#<Регулярний вираз>#[<Модифікатор>]'

'"< Регулярний вираз >"[<Модифікатор>]'

'{< Регулярний вираз >}[<Модифікатор>]'

'(<Регулярний вираз >)[<Модифікатор>]'

У параметрі <Модифікатор> можуть бути вказані наступні прапорці (або їх комбінація):

i - пошук без врахування регістра;

s - однорядковий режим. Символ ^ відповідає прив'язці до початку рядки, а символ $ відповідає кінцю рядка. Метасимвол "точка" відповідає будь-якому символу, у тому числі і символу переведення рядка;

u - використовується для обробки рядків в кодуванні UTF-8; (є й інші модифікатори).

Метасимволи, використовувані в регулярних виразах:

^ - прив'язка до початку рядка (призначення залежить від модифікатора);

$ - прив'язка до кінця рядка (призначення залежить від модифікатора);

\A - прив'язка до початку рядка (не залежить від модифікатора);

\z - прив'язка до кінця рядка (не залежить від модифікатора);

[ ] - дозволяє вказати символи, які можуть зустрічатися на цьому місці в рядку. Можна перераховувати символи підряд або вказати диапазон через тире;

[^] - дозволяє вказати символи, які не можуть зустрічатися на цьому місці в рядку. Можна перераховувати символи підряд або вказати диапазон через тире;

n|m - відповідає одному з символів n або m;

. (крапка) - будь-який символ, окрім символу переведення рядка (\n). Усередині квадратних дужок крапка не має спеціального значення.

Крім того, в регулярних виразах можна використовувати наступні стандартні класи:

\d - відповідає будь-якій цифрі;

\w - відповідає будь-якій букві або цифрі;

\s - будь-який пробільний символ (пропуск, табуляція, переведення сторінки, новий рядок або переведення каретки);

\D - не цифра;

\W - не буква і не цифра;

\S - не пробільний символ.

Квантифікатори, використовувані в регулярних виразах, дозволяють задати число повторів попереднього символу або виразу:

{n} - n входжень символу в рядок;

{n,} - n або більше входжень символу в рядок;

{n,m} - не менше n і не більш за m входжень символу в рядок. Числа зазначаються через кому без пропуску;

* - нуль або більше входжень символу в рядок;

+ - одне або більше входжень символу в рядок;

? - жодного або одне входження символу в рядок.

3 завдання.

Завдання полягає в наступному.

php-сценарій виводить поле форми із запитом „Введіть рядок різних символів” (рис.15.5)

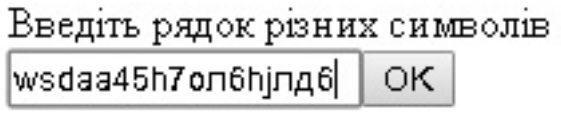

Рис. 15.5. Приклад – запит для введення рядка символів

Коли користувач вводить рядок довільних символів і підтверджує „ОК”, сценарій відбирає з цього рядка символів тільки числа та виводить їх на екран браузера

4 - 5 - 7 - 6 - 6

Підказки:

- Тут також використовується технологія „регулярних виразів” для аналізу строкових змінних.

Функція $arr = preg_grep($pattern, $str1) повертає з масиву $str1 підмасив елементів, що відповідають шаблону $pattern. Шаблон створюється відповідно до наведених вище пояснень щодо регулярних виразів.

- Спочатку введений користувачем рядок символів треба перетворити в масив; для цього можна використати функцію $str1 =str_split($str).

- Вивести на сторінку браузера масив-результат можна за допомогою функції echo implode(" - ", $arr), де " - " позначає знак, що буде розділяти знайдені елементи на екрані.

За результатами виконання роботи в звіті повинні бути представлені:

1. Назва роботи
2. Мета роботи
3. Лістинги сценаріїв та скрін-шоти сторінок результатів їх роботи.
4. Висновки

15.3. Питання до самоконтролю

15.3.1. Що таке регулярні вирази и як вони використовуються?

15.3.2. Як в PHP-сценарії передається інформація, котра введена користувачем в поле форми?

15.3.3. Яким чином PHP-сценарії вбудовуються в код HTML-сторінки?

15.3.4. Пояснити сутність та призначення суперглобальних масивів в PHP, їх різновиди та як позначаються.

16. РЕАЛІЗАЦІЯ ВЗАЄМОДІЇ З БАЗАМИ ДАНИХ В СЕРВЕРНИХ СЦЕНАРІЯХ

Мета роботи: Навчитись застосовувати засоби PHP у взаємодії з MySQL для роботи з базами дагних при реалізації серверних сценаріїв WEB.

16.1. Інформація для самостійної підготовки

Сучасні технології для зберігання інформації використовують реляційні бази даних, і так склалося, що найпоширенішою з них є MySQL. Нагадаємо, що при створенні і налаштуванні середовища локального сервера на практичних заняттях №1 та №13 однією зі складових цього середовища була якраз MySQL, і ми вже навчились її встановлювати і налаштовувати. З базами MySQL можна працювати безпосередньо за допомогою програми-монітора, але типовим варіантом є застосування зв'язки PHP-MySQL, тобто використання серверних PHP-сценаріїв для роботи з MySQL. Тому саме аспекту взаємодії сценаріїв PHP з MySQL буде приділена увага. При виникненні потреби нагадувань щодо якихось елементів технологій рекомендується звернутись до відповідних літературних джерел, наприклад [1], проте тут коротко наведемо інформацію щодо PHP-засобів для роботи з MySQL.

Підключення до сервера MySQL і вибір бази даних

Підключення до сервера баз даних MYSQL виконується наступною командою:

resource mysql_connect ([string server [,string username [,string password [,bool new_link [,int client_flags]]]]])

Функція mysql_connect() повертає значення ресурсного типу даних, що містить покажчик на з'єднання з сервером MySQL. В разі невдачі повертається значення false.

Параметр server містить ім'я вузла, на якому запущений сервер. За умовчанням його значення рівне localhost, але у разі потреби можна вказати шлях і номер порту для підключення до сервера, наприклад: /path_to_MySQL_socket або database:3306. Для локального сервера MYSQL використовується значення за умовчанням.

Параметри username і password задають, відповідно, логін і пароль для підключення до сервера MySQL.

У нашому випадку використання Денвер, як пам’ятаємо з 1-го практичного заняття, username – це "root", а пароль password – порожня строка, тобто "".

Останні параметри менш істотні. Присвоєння значення true параметру new_link призводить до відкриття нового з'єднання з MySQL в разі повторного звернення до функції mysql_connect(). За умовчанням встановлений режим, при якому функція просто поверне покажчик на вже відкрите з'єднання. Параметр client_flags задає особливі режими роботи з сервером.

Якщо помістити перед викликом функції mysql_connect() символ @, то в разі виникнення помилки при відкритті з'єднання користувач не отримає відповідне повідомлення. Це зручний засіб, оскільки такі повідомлення, як правило, недружні до користувача. Замість цього слід при кожному відкритті з'єднання перевіряти набутого значення функції і видавати власні повідомлення, зрозуміліші споживачам нашого сценарію.

З'єднання з сервером MySQL закривається або після закінчення сценарію, або шляхом виклику функції mysql_close().

Розглянемо приклад коректного відкриття і закриття з'єднання з MYSQL:

```
<?php
$connection = mysql_connect("localhost", "user_MySQL",
"password_MySQL");
if($connection==false)
die("Неможливо підключитися до MYSQL: " . mysql_error());
echo "З'єднання з MYSQL виконане успішно";
mysql_close ($connection);
?>
```

Тут у разі неуспішного виконання функції MySQL викликається функція die(), яка виводить повідомлення про помилку і завершує роботу сценарію. Дане повідомлення містить наш текст, до якого операцією конкатенації текстових рядків («.») додається текст повідомлення про помилку, повернутий функцією mysql_error().

Після з'єднання з сервером MySQL потрібно вибрати базу даних, яка нам потрібна для подальших дій. Це робиться за допомогою оператора

Bool mysql_select_db(string ім’я_бази_даних [, resource вказівник_з’єднання_з_MySQL])

В разі успішного вибору бази даних функція повертає значення true, неуспіху — false. Для вибору бази даних функції передаються ім'я бази даних і покажчик на з'єднання з сервером MySQL, який ми отримали від функції myssql_connect() у попередньому прикладі. Якщо опустити цей покажчик, буде використано останнє відкрите з'єднання з MySQL, а в разі його відсутності буде зроблена спроба відкрити з'єднання з MySQL, причому за допомогою параметрів, які використовує за умовчанням функція mysql_connect().

Якщо виклик функції mysql_select_db() завершиться успішно, то всі подальші запити до бази даних, що виконуються зі сценарію, будуть звернені до вибраної бази даних. Для виконання таких запитів PHP надає безліч функцій. Опишемо деякі з них, найчастіше вживані в сценаріях.

Робота з базою даних MySQL

По-перше, створення нової бази даних засобами PHP. Для цього використовується функція mysql_query(), що виконує запити до сервера MySQL. Вона має наступний синтаксис:

resource mysql_query (string mysql_query [, resource mysql_connection])

Помилка при виконанні функції повертає значення false, інакше, в разі запиту на створення бази даних, повертається true (для інших запитів значення, що повертається, може містити покажчик на результат запиту).

Параметр mysql_query містить рядок запиту до бази даних, а mysql_connection — покажчик на з'єднання з сервером.

Рядок запиту формується за наступними правилами. При створенні нової бази даних запит записується таким чином:

CREATE DATABASE IF NOT EXIST db_name

Тут параметр db_name містить ім'я створюваної бази даних. Згідно запиту вона буде створена, якщо такої бази даних ще не існує.

Наведемо приклад коду, що виконує створення бази даних MySQL:

```
$mysql_query="CREATE DATABASE IF NOT EXIST my_db"
$result= mysql_query($mysql_query);
If (!$result)
die ("Помилка створення бази даних");
```

Зміст цього коду досить простий: ми формуємо в змінній $mysql_query рядок запиту до MySQL на створення бази даних з ім'ям

my_db, після чого звертаємося до функції mysql_query(). Якщо запит на створення бази даних виконається з помилкою, виконання сценарію припиняється.

Після створення бази даних ми повинні заповнити її інформацією. Для цього в базах даних використовуються таблиці. Таблиця бази даних створюється наступним запитом:

CREATE TABLE table_name (column_1, column_2, ...column_N)

У запиті задаються ім'я таблиці (змінна table_name) і імена стовпців (змінні column_...).

Так само формується запит на додавання в таблицю даних:

INSERT INTO table_name (column_1, column_2 ..., column_N) VALUE (value_1, value_2 ., value_N)

Тут змінні value_1 ..., value_N містять значення, що додаються в стовпці таблиці, перераховані в операторі після її імені. При додаванні значень до всіх стовпців їх список можна опустити.

Запит вибірки даних має наступний вигляд:

SELECT name_1, name_2 ...name_N FROM name_table

Тут name_... — це імена стовпців таблиці з назвою name_table.

16.2. Порядок виконання роботи:

Устаткування, матеріали та інструменти

- Персональний комп’ютер зі встановленим браузером та текстовим редактором типу Блокнот або WordPad.

Порядок виконання роботи

Робота складається з одного завдання, але у кожного студента свій варіант, номер якого відповідає порядковому номеру у списку групи студентів.

Кожному студенту треба створити веб-сторінку з відповідною формою, через яку можна додавати, видаляти та передивлятись записи в базі даних засобами PHP-MySQL.

Варіанти бази даних наступні:

1) База даних студентів з полями: прізвище, рік народження, середній бал, місто.
2) База даних автомобілів: номер, рік випуску, марка, колір, прізвище власника.
3) База даних книгжок в бібліотеці: назва, автор, рік видання.
4) База даних співробітників фірми, що мають комп'ютер на робочому місці: прізвище, номер кімнати, назва відділу, дані про комп'ютер.
5) База даних замовлень, що отримані співробітниками фірми: прізвище, сума замовлення, найменування товару, прізвище замовника.
6) База даних оцінок, що отримані студентами на іспитах: прізвище, група, предмет, номер квитка, оцінка.
7) База даних списку розсилки і підписчиків: тема і вміст листа, дата відправки, імена і адреси підписчиків.
8) База даних книжкового інтернет-магазину: назва книжки, автор, ціна, кількість на складі.
9) База даних фільмів, які можна передивитись: назва, рік випуску, студія, жанр.
10) База даних товарів на складі: найменування товару, вага, ціна, строк зберігання, кількість на складі.
11) База даних авіарейсів: акропорти вильоту і прильоту, час і дата вильоту, авіакомпания, ціна.

12) База даних театральної каси: назва вистави, дата, театр, вартість квитків.
13) База даних автомобільного магазину: марка, рік випуску, колір, ціна, кількість на складі.
14) База даних – телефонний довідник: ПІБ абонента, адреса, номер телефона.
15) База даних інтернет-магазину побутової техніки: вид товару, компанія-виробник, марка, ціна.
16) База даних – учбовий розклад: назва дисципліни, вид занять, аудиторія, час проведення.
17) База даних спортивних змагань в місті: вид спорту, час і місце проведення, команди-учасники, вартість квитків.
18) База даних автобусних рейсів: кінцевий пункт, час відправлення, ціна квитка, кількість вільних місць.
19) База даних – навчальний план дисципліни: тема занять, кількість лекційних годин, кількість практичних занять, тематика для самостійного опрацювання.
20) База даних – турнірна таблиця з футболу: команда, порядковий номер в турнірному розташуванні, кількість очок, кількість зіграних ігор, кількість виграшів.
21) База даних – меню ресторану: назва блюда, категорія (перші страви, салати, напої…), ціна.
22) База даних колекції CD-дисків: назва, виконавець, рік випуску, кількість хвилин.
23) База даних публікацій науковця: назва публікації, реквізити (журнал, рік, номер).
24) База даних машрутів туристичної агенції: країна, строки поїздки, умови, ціна.

25) База даних пацієнтів лікарні: ПІБ, рік народження, адреса, діагноз.

За результатами виконання роботи в звіті повинні бути представлені:

1. Назва роботи
2. Мета роботи
3. Лістинги сценаріїв та скрін-шоти сторінок результатів їх роботи.
4. Висновки

16.3. Питання до самоконтролю

16.3.1. Що розуміється під терміном „реляційна” база даних?

16.3.2. Що таке Індекси в технології реляційних баз даних, їх призначення і створення.? Що таке індексування баз даних?

16.3.3. Пояснити структуру операторів запиту даних в мові SQL.

16.3.4. Що таке „ресурсний тип даних” та як такі дані використовуються в технологіях реляційних баз даних?

17. ВИКОРИСТАННЯ ТЕХНОЛОГІЇ AJAX ДЛЯ ПОКРАЩЕННЯ ЕКСПЛУАТАЦІЙНИХ ЯКОСТЕЙ ВЕБ-ЗАСТОСУВАНЬ

Мета роботи: Навчитись використовувати технологію AJAX при створенні якісних програмних продуктів.

17.1. Інформація для самостійної підготовки

Сучасні веб-технології мають справу з великими обсягами інформації та значними інформаційними потоками між сервером і клієнтом. В традиційному варіанті будь-який запит додаткової інформації від клієнта до сервера викликає перезавантаження всієї веб-сторінки. Це непродуктивно, викликає значні затримки в часі та надмірне навантаження на інформаційні канали. Тому у якісних веб-продуктах вважається гарним стилем використання технології AJAX, завдяки котрій інформація на сторінці оновлюється частинами, дуже швидко і з меншим навантаженням на канали.

Зміст технології AJAX (Asynchronous JavaScript and XML), як це випливає з її назви, полягає в створенні запитів серверу, які виконуються в асинхронному режимі. Це означає, що серверна частина веб-застосування, що використовує AJAX, працює незалежно від клієнтської частини. Після того, як клієнт послав запит серверу, останній починає його обробку, а клієнт продовжує свою роботу, не вимушуючи користувача чекати відповіді. Наприклад, користувач може вводити дані у форму, що відображується у вікні клієнта, а клієнт, не чекаючи закінчення введення і натискання кнопки підтвердження форми, посилає дані на сервер, який непомітно для користувача обробляє їх і посилає результати клієнтові.

AJAX згідно назви базується на технологіях мови JavaScript і XML.

Краще всього проілюструвати базові принципи AJAX на конкретному прикладі [1].

Створимо і збережемо в каталозі examples файл ajax.html наступного HTML-документу:

Лістинг 17.1. Веб-сторінка із запитами AJAX

```
<html>
<head>
<title>ПрикладAJAX</title>
<script type="text/javascript" src="ajax.js"></script>
</head>
<body onload=’request()’>
Enter your login:
<input type="text" id="myLogin" />
<div id="Message" />
</body>
</html>
```

Як бачимо, він містить дескриптор завантаження сценарію JavaScript, збереженого у файлі ajax.js (див. лістинг 17.2). У свою чергу, сценарій містить оператори функцій JavaScript для ініціації запитів AJAX і їх виконань.

Лістинг 17.2. Сценарій ініціації і обробки запитів AJAX

```
if(window.ActiveXObject)
var xmlHttp = new ActiveXObject("Microsoft.XMLHTTP");
else
var xmlHttp = new XMLHttpRequest();
if (!xmlHttp)
alert("Error creating the XMLHttpRequest object.");
function request() {
if (xmlHttp.readyState == 4 || xmlHttp. readyState == 0) {
```

```
name = document.all("myLogin").value;
xmlHttp.open("GET", "ajax.php?login=" + name, true);
xmlHttp.onreadystatechange = ResponseHandler;
xmlHttp.send(null);
}
else
setTimeout('request()', 1000);
}
function ResponseHandler() {
if (xmlHttp.readyState == 4) {
if (xmlHttp.status == 200) {
xmlResponse = xmlHttp.responseXML;
xmlDocumentElement = xmlResponse.documentElement;
Response = xmlDocumentElement.firstChild.data;
document.all("Message").innerHTML =
'<i>' + Response + '</i>';
setTimeout('request()', 1000);
}
else {
alert("Помилка доступу до сервера");
} } }
```

Збережемо файл цього сценарію в каталозі examples під ім'ям ajax.js. Далі створимо сценарій PHP і збережемо його в каталозі examples під ім'ям ajax.php (див. лістинг 17.3).

Лістинг 17.3. Серверна частина застосування

```
<?php
header ('Content-Type: text/xml');
echo '<?xml version="1.0" ?>';
```

```
echo ,<response>‘;
$login = $_GET[,login‘];
$userNames = array(,PETYA‘, ,SERGEY‘, ,IVAN‘, ,MASHA‘, ,LENA‘);
if (in_array(strtoupper($login) $userNames))
echo ,Hello, user , . $login . ,!‘;
else if (trim($login)== ,‘)
echo ,Empty login‘;
else
echo $login, ,, - unregistered user‘;
echo ,</response>‘;
?>
```

Відкриємо документ ajax.html, ввівши localhost/examples/ajax.html у адресний рядок браузеру. У вікні браузеру відображуватиметься поле Enter your login для введення вхідного імені користувача. Почнемо вводити в нього який-небудь текст і помітимо, як паралельно з введенням в рядку під полем відображується повідомлення сервера про результати обробки введених даних.

Звернемо увагу, що від нас не вимагається натискати кнопку підтвердження відправки форми, як ми це робимо зазвичай. Сервер сам, незалежно від наших дій, виконує прочитування введених даних і їх порівняння із списком відомих йому імен користувачів. Якщо введений текст не збігається з яким-небудь відомим серверу ім'ям, відображується повідомлення Unregistered user (Незареєстрований користувач). Але якщо ввести ім'я, відоме серверу, то відображуватиметься вітання.

Тепер розглянемо, як працює наше веб-застосування. При завантаженні сторінки разом з наведеним в лістингу 17.1 HTML-кодом завантажується код програми JavaScript, представлений в лістингу 17.2.

При цьому згідно правил обробки коду HTML-документа буде виконана частина програми JavaScript, що знаходиться на початку лістингу:

```
if(window.ActiveXObject)
var xmlHttp = new ActiveXObject("Microsoft.XMLHTTP");
else
var xmlHttp = new XMLHttpRequest();
if (!xmlHttp)
alert("Error creating the XMLHttpRequest object.");
```

Тут виконується створення об'єкту XMLHttpRequest, що відповідає за здійснення запитів AJAX. Залежно від того, який браузер використовується, об'єкт створюється викликом функції або ActiveXObject(), або XMLHttpRequest(). При успішному створенні об'єкту змінній xmlHttp призначається значення посилання на об'єкт, в разі невдачі генерується повідомлення.

Тепер звернемося до рядка коду HTML з першого лістингу:

```
<body onload=’request ()’>
```

Він означає, що при завантаженні сторінки викликається функція request() з сценарію ajax.js:

```
function request() {
if (xmlHttp.readyState == 4 || xmlHttp.readyState == 0) {
name = document.all("myLogin").value;
xmlHttp.open("GET", "ajax.php?login=" + name, true);
xmlHttp.onreadystatechange = ResponseHandler;
xmlHttp.send(null);
}
else
setTimeout(‘request()’, 1000);
}
```

Робота вказаної функції складається з двох частин: підготовка запитів AJAX до виконання і запуску механізму таких запитів. Спочатку перевіряється стан готовності об'єкту XMLHttpRequest. Це робиться за допомогою параметра объекта readyState, який може мати наступні значення:

0 — запит не ініціалізовано;

1 — йде відправка запиту;

2 — запит відправлений;

3 — відбувається обмін даними;

4 — запит завершений.

Отже, ми перевіряємо, чи був запит завершений або не ініціалізований, і в будь-якому з цих випадків починаємо його виконання. Інакше за допомогою функції setTimeout() встановлюється затримка повторного виклику функції request():

setTimeout (‘request()’, 1000);

На першому етапі підготовки запитів AJAX за допомогою колекції all ми визначаємо значення елементу MyLogin в нашому документі HTML, тобто поля, що містить введений користувачем текст:

name = document.all("myLogin").value;

Далі за допомогою методу open об'єкту XMLHttpRequest ми викликаємо виконання серверного сценарію PHP:

xmlHttp.open("GET", "ajax.php?login=" + name, true);

Як видно з рядка виклику сценарію

"ajax.php?login=" + name

йому передається текст, введений користувачем в полі нашого документа HTML. Крім того, ми задаємо асинхронний режим обміну інформації з сервером, оскільки другий параметр методу рівний true.

Після цього запускається механізм AJAX асинхронних запитів: клієнт буде автоматично, без жодного втручання користувача, посилати запити серверу і в разі одержання відповіді передавати отримані дані на обробку спеціальною функцією. Ця функція призначається наступним оператором:

xmlHttp.onreadystatechange= ResponseHandler;

тут ми просто призначили параметру onreadystatechange об'єкта XMLHttpRequest посилання на функцію ResponseHandler(), тобто при зміні стану об’єкта буде викликатися функція ResponseHandler().

Тепер все готово для запитів AJAX. Викликаємо метод send() об'єкту XMLHttpRequest:

xmlHttp.send (null);

і відправляємо запит на сервер.

На стороні сервера виконується дуже простий сценарій PHP (див. лістинг 17.3). Спочатку ми відправляємо у відповідь на запит заголовок XML-повідомлення:

header(‘Content-Type: text/xml’);

echo ‘<?xml version="1.0" ?>’;

Перший рядок містить важливий елемент заголовка, що вказує на те, що ми передаємо дані у форматі xml. Далі ми відправляємо інструкцію браузеру, що містить інформацію про використану в повідомленні версію мови XML, і перший тег цього повідомлення:

echo‘<response>’;

Формування повідомлення починається з процедури отримання з рядка виклику сценарію переданого йому параметра

$login= $_GET[‘login’];

Далі ми просто порівнюємо його із значеннями в масиві $userNames, використовуючи для цього вбудовану функцію PHP in_array(), і, якщо збіг є, передаємо повідомлення

Echo ‘Hello, user ‘ . $login. ‘!’;

Інакше ми перевіряємо, чи не порожній рядок був переданий серверу. Це не зайва перевірка, оскільки саме з такого випадку починаються запити до серверу, коли користувач ще не встиг ввести жодних даних. Аби перевірка була коректною, спочатку за допомогою функції trim() з переданої змінної видаляються всі пробіли trim($login), які можуть знаходитися на початку і кінці рядка. Якщо переданий параметр порожній, відображується повідомлення про порожнє поле.

Якщо введемо ім'я, відоме серверу, то відображуватиметься вітання.

Але сервер передає свої повідомлення в незрозумілому браузеру форматі xml. Як же такі повідомлення відображуються на екрані браузера? Відповідь проста: це робить програма-обробник переривань:

Лістинг 17.4. Програма-обробник переривань

```
Function ResponseHandler() {
if (xmlHttp.readyState== 4) {
if (xmlHttp.status == 200) {
xmlResponse = xmlHttp.responseXML;
xmlDocumentElement = xmlResponse.documentElement;
Response = xmlDocumentElement.firstChild.data;
document.all("Message").innerHTML=’<i>’+Response+’</i>’;
setTimeout(‘request()’, 1000);
}
else {
alert("Помилка доступу до сервера");
} } }
```

Даний сценарій автоматично викликається клієнтом при кожній зміні в статусі відправленого запиту AJAX. На першому кроці перевіряється стан запиту згідно значенню вже згаданого параметру readyState об'єкту XMLHttpRequest:

If(xmlHttp.readyState== 4)

Якщо запит успішно завершений, перевіряється інший параметр status об'єкту XMLHttpRequest:

if(xmlHttp.status== 200)

Значення 200 повідомляє про успішну відповідь. Якщо запит успішно завершений, сценарій приступає до розбирання отриманого повідомлення XML для одержання корисної інформації, переданої сервером. Для цього використовуються параметри об'єкту XMLHttpRequest:

xmlHttp.responseXML — вилучає отримане повідомлення XML;

xmlDocumentElement = xmlResponse.documentElement — вилучає кореневий елемент в отриманому повідомленні XML;

xmlDocumentElement.firstChild.data — вилучає вміст кореневого елементу XML.

Отримана в результаті корисна інформація призначається змінній Response, за допомогою якої складається відповідь сервера:

‘’+Response+’’

Нарешті, результат всіх цих дій відображується на екрані браузера в елементі

.

AJAX-запити в jQuery

Крім описаної вище технології AJAX з безпосереднім використанням засобів JavaScript та PHP існує можливість її суттєвого спрощення через використання розглянутої на практичному занятті №12 бібліотеки jQuery.

Краще всього це проілюструвати, як і раніше, на конкретному прикладі [2].

Зміст прикладу полягає в наступному: при завантаженні сторінки (лістинг 27.5) ми бачимо лише кнопку Start. Проте при натисканні на цю кнопку буде виконано AJAX-запит до файла, код якого наведено в лістингу 17.6, і ми побачимо, як вміст цього файла завантажиться в елемент div з ідентифікатором #example, і в браузері з'явиться повноцінна веб-сторінка з відображенням деякого меню.

Лістинг 17.5. Програма реалізації AJAX-запиту

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script type="text/javascript" src="js/jquery-1.2.6.js"></script>
<script type="text/javascript">
<!--
$(document).ready(function(){
$("button").click(function() {
$("#example").load("27.2.html");
});
});
-->
</script>
<style type="text/css">
span {
color:#00f;
}
</style>
</head>
<body>
<div id="example"></div>
```

```
<button>Start</button>
</body>
</html>
```

Лістинг 17.6. Веб-сторінка, яка завантажується після AJAX-запиту

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title> Веб-сторінка, яка завантажується після AJAX-запиту</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
</head>
<body>
<h1>Сервер вітає Вас!</h1>
<ul id="menu">
<li>Пункт меню № 1</li>
<li>Пункт меню № 2</li>
<li>Пункт меню № 3</li>
<li>Пункт меню № 4</li>
<li>Пункт меню № 5</li>
</ul>
<p>Якийсь текст, що знаходится в елементі p</p>
</body>
</html>
```

Звернемо увагу, що в цьому прикладі всі функції, необхідні для реалізації технології AJAX, виконує метод load() з бібліотеки jQuery. Слід

зауважити, що завдяки бібліотеці jQuery маємо ще багато варіантів реалізації технології AJAX, з котрими рекомендується ознайомитись [2].

17.2. Порядок виконання роботи:

Устаткування, матеріали та інструменти

- Персональний комп'ютер зі встановленим браузером та текстовим редактором типу Блокнот або WordPad.

Порядок виконання роботи

Робота складається з одного завдання з використанням технології AJAX. Слід створити маленький аналог пошукової системи Google. Тобто кожен студент створює невелику базу даних з кількох записів за якоюсь тематикою (тематика на свій смак, але у всіх студентів різна). Далі створюється інтерфейс для задавання пошукового рядка символів, і система асинхронно за технологією AJAX намагається знайти в базі даних відповідний запис (асинхронно виводить при кожній деталізації запиту всі записи бази даних, що відповідають шаблону запиту).

За результатами виконання роботи в звіті повинні бути представлені:

1. Назва роботи
2. Мета роботи
3. Лістинги сценаріїв та скрін-шоти сторінок результатів їх роботи.
4. Висновки

17.3. Питання до самоконтролю

17.3.1. Що розуміється під терміном „асинхронно" в технології AJAX?

17.3.2. Чому оптимальним форматом передачі інформації від сервера в технології AJAX вважається xml?

17.3.3. У чому полягають переваги використання технології AJAX в сучасних веб-проектах?

ПРЕДМЕТНИЙ ПОКАЖЧИК

В

Д

О

С

ПЕРЕЛІК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Ташков П. А. Веб-мастеринг на 100 %: HTML, CSS, JavaScript, PHP, CMS, AJAX, раскрутка. – СПб.: Питер, 2010. – 512 с.
2. Бенкен, Е. С. AJAX: программирование для Интернета / Е. С. Бенкен, Г. А. Самков. – СПб.: БХВ-Петербург, 2009. – 464 с.

ПЕРЕЛІК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ

3. Прохоренок Н.А. HTML, JavaScript, PHP и MySQL. Джентльменский набор Web-мастера. – 3-е изд., перераб. и доп. – СПб.: БХВ-Петербург, 2010. – 912 с.
4. Никсон Р. Создаем динамические веб-сайты с помощью PHP, MySQL и JavaScript. – СПб.: Питер, 2011. – 496 с.
5. Дари К. AJAX и PHP: разработка динамических веб-приложений / К. Дари, Б. Бринзаре, Ф. Черчез-Тоза, М. Бусика. – СПб.: Символ-Плюс, 2007. – 336 с.
6. Девис Е.М. Изучаем PHP и MySQL / Е.М. Девис, Дж.А. Филипс. – СПб.: Символ-Плюс, 2008. – 448 с.