МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ ІМЕНІ СЕМЕНА КУЗНЕЦЯ

В. П. Молчанов
О. К. Пандорін

ТЕХНОЛОГІЇ РОЗРОБКИ WEB-РЕСУРСІВ

Навчальний посібник

Харків
ХНЕУ ім. С. Кузнеця
2019

УДК 004.4'27(075.034)
М76

Авторський колектив: канд. техн. наук, доцент В. П. Молчанов – вступ, розділ 1; канд. техн. наук, доцент О. К. Пандорін – розділ 2.

Рецензенти: ректор ПЗВО "Харківський технологічний університет "ШАГ", канд. техн. наук, д-р екон. наук, професор *С. В. Кавун;* доцент кафедри медіасистем та технологій Харківського національного університету радіоелектроніки, канд. техн. наук *А. В. Бізюк.*

Рекомендовано до видання рішенням ученої ради Харківського національного економічного університету імені Семена Кузнеця.
Протокол № 9 від 27.05.2019 р.

Самостійне електронне текстове мережеве видання

Молчанов В. П.

М76 Технології розробки WEB-ресурсів [Електронний ресурс] : навчальний посібник / В. П. Молчанов, О. К. Пандорін. – Харків : ХНЕУ ім. С. Кузнеця, 2019. – 130 с.

ISBN 978-966-676-754-0

Наведено матеріал, що допомагає освоїти всі лекційні теми навчальної дисципліни. Виклад теорії супроводжується великою кількістю ілюстративного матеріалу у вигляді відповідних вікон, рисунків і прикладів. Методика викладення передбачає активне залучення студента до навчального процесу. Наведені приклади дозволяють отримати цілісне уявлення про можливості технології та можуть використовуватися як довідковий матеріал під час виконання лабораторних робіт і завдань для самостійної роботи.

Рекомендовано для студентів спеціальності 186 "Видавництво та поліграфія".

УДК 004.4'27(075.034)

ISBN 978-966-676-754-0

Вступ

WEB-технології зараз постійно розвиваються, надаючи користувачам і розробникам все більше можливостей. Для створення дійсно сучасного ресурсу недостатньо знань тільки базових компонент (HTML, CSS, JS на боці клієнта), необхідно вміти організувати взаємодію із сервером, із базами даних, використовувати можливості XML, сучасну WEB-графіку (SVG і т. п.). Саме ці питання і розглядаються в дисципліні "Технологія розробки WEB-ресурсів".

Навчальна дисципліна "Технологія розробки WEB-ресурсів" належить до професіонального циклу базових навчальних дисциплін. Вона продовжує підготовку фахівця і розвіває вміння створювати документи для мережі Інтернет.

У межах дисципліни розглянуто основні сучасні серверні WEB-технологи та технологічні засоби їх підтримки. Отримані в ході вивчення дисципліни знання та вміння необхідні не лише тим, хто створюватиме WEB-сайти самостійно, але і тим, хто для виконання цих робіт вважає за краще звернутися до професіоналів. Знання предмета стане запорукою ефективної й якісної взаємодії з розробником.

Метою викладання цієї навчальної дисципліни є формування системи теоретичних знань про серверні компоненти сервісу WWW, їх місце серед інших комп'ютерних технологій і комплекс умінь зі створення WEB-ресурсів, їх розміщення в мережі Інтернет та аналізу-функціонування.

Для досягнення мети поставлені такі основні **завдання:**

сформувати понятійний апарат і розуміння взаємозв'язку між основними технологічними компонентами WEB;

набути вмінь створення WEB-ресурси з використанням сучасних підходів;

набути вмінь працювати із сучасними технологічними засобами створення WEB-ресурсів;

навчитися оцінювати якість і ефективність створених ресурсів сервісу WWW.

Об'єктом навчальної дисципліни є процеси створення документів для сервісу WWW.

Предметом навчальної дисципліни є документи та технологічні засоби сервісу WWW (інтерфейси, сервери, мови, протоколи).

У результаті вивчення навчальної дисципліни студенти повинні:

знати:

основні принципи функціонування сервісу WWW, особливості розміщення і пересилки документів по мережі Інтернет за допомогою найбільш поширених серверів;

основні інтерфейси сервера й основні засоби програмування на боці сервера;

програми і методи створення динамічних ресурсів для сервісу WWW;

особливості різних технологій активних сторінок, і засоби їх створення, особливості форматів використовуваних файлів;

додатки мови XML, та їх використання у WEB-технологіях;

можливості нових графічних форматів для WEB;

вміти:

вибирати засоби, методи і технології для ефективного створення WEB-ресурсів;

використовувати додатки мови XML на WEB-ресурсах;

розміщувати на WEB-сторінках динамічну графіку;

створювати динамічні сторінки і обробляти дані з форм, використовуючи засоби програмування на стороні сервера WEB;

виконувати перевірку і відлагодження створюваних програмних елементів;

розміщувати створені ресурси в мережі Інтернет.

Викладення матеріалу супроводжує багато практичних прикладів. Закінчені приклади, які можуть бути використані для запуску на комп'ютері, виділені рамкою та шрифтом. Інший тип прикладів – це працездатні фрагменти, які також можна використати, але їх слід доповнити потрібним кодом.

У тексті посібника в рамці із позначкою вміщено допоміжні матеріали, які під час першого прочитання можна пропустити. Наприкінці кожного розділу наведено список контрольних запитань для самостійної перевірки засвоєння теоретичного матеріалу та набір завдань для самостійної практичної роботи.

Перший розділ присвячено розгляду існуючих серверних технологій та питань взаємодії клієнта із сервером із використанням інтерфейсу CGI, використанню мови XML та основам розробки програм мовою PHP. У другому розділі розглянуті технологічні засоби створення WEB-ресурсів, використання баз даних, асинхронна взаємодія із сервером і динамічна графіка.

Автори висловлюють глибоку вдячність рецензентам: доктору економічних наук, доценту Кавуну С. В. і кандидату технічних наук, доценту Бізюку А. В. за велику увагу до навчального посібника та надані рекомендації і поради стосовно його змістовного наповнення.

Розділ 1. Серверні технології створення динамічних WEB-сторінок

У розділі розглянуто основи створення серверних WEB-додатків, різні підходи і технології, застосування XML та основи використання мови PHP.

Головним має стати розуміння студентами взаємодії браузера на машині користувача із сервером. Яке програмне забезпечення повинно бути встановлено на кожній із машин, а яке розробляється для конкретного ресурсу. Як розподілити функції додатка між клієнтом і сервером, і в чому власне полягає їхня взаємодія.

Ключові поняття розділу: WEB-сервіс, WEB-сервер, клієнт, взаємодія клієнта та сервера, інтерфейс сервера, WEB-сторінка, мова розмітки XML, мова програмування PHP, синтаксис мови PHP, використання мови PHP.

Питання розділу:

Тема 1. Характеристика серверних технологій
- 1.1. Серверні технології
- 1.2. Використання CGI-інтерфейсу

Тема 2. Мова розмітки XML
- 2.1. Загальна характеристика мови XML
- 2.2. Застосування XML

Тема 3. Мова програмування PHP
- 3.1. Розробка програм мовою PHP
- 3.2. Використання функцій
- 3.3. Обробка даних

Цілі вивчення розділу

Інформація, подана в розділі, надає студентові можливість сформувати такі **компетентності:**

здатність обґрунтувати вибір технології для створення WEB-ресурсів;

здатність створювати та обробляти XML-документи, використовувати мову XML для зберігання даних додатків;

здатність аналізувати виконання WEB-додатків;

здатність виконувати розробку та відлагодження додатків мовою PHP;

знання:

основних принципів функціонування сервісу WWW, особливості розміщення документів та функціонування найбільш поширених серверів;

основні інтерфейси сервера і умови виконання програм на сервері WWW;

особливості різних технологій активних сторінок, і засоби їх створення;

додатки мови XML, їх використання у WEB-технологіях;

основи програмування мовою PHP;

уміння:

обирати засоби, методи і технології для створення WEB-ресурсів;

використовувати додатки мови XML на WEB-ресурсах;

створювати динамічні сторінки і обробляти дані з форм, використовуючи мову PHP;

комунікації:

доведення своїх висновків і результатів роботи до учасників команди;

уміння користуватися консультацією фахівців для вирішення проблемних питань;

автономність і відповідальність:

пошук шляхів вирішення проблем, що виникають під час функціонування компонент сервісу WWW;

пошук альтернативних засобів для роботи з WEB-ресурсами;

самостійний пошук та вибір засобів для роботи з XML-документами.

Вступ

Можливості ресурсів мережі Інтернет постійно розширюються. Якщо спочатку це були просто сторінки з інформаційними матеріалами у вигляді тексту та ілюстрацій, то в даний час вони можуть реалізовувати як завгодно складні функції. Реалізується це за рахунок включення до складу ресурсів різних програм. Програми, що виконуються в браузері на боці клієнта (скрипти), були розглянуті в дисципліні "Основи проєктування

WEB-видань". У даному розділі будуть розглянуті питання виконання програм на сервері за запитами з браузера.

Тема 1. Характеристика серверних технологій

1.1. Серверні технології

Функціонування мережі Інтернет засноване на архітектурі клієнт-сервер. Водночас програма-клієнт (зазвичай браузер) на машині користувача відправляє запити на ресурси (зазвичай це запит файла з WEB-сторінкою) програмі-серверу, що встановлена на одній із машин, підключених до мережі та ідентифікованої IP-адресою. Логіка роботи сервера забезпечує видачу файлів, які зберігаються на машині-сервера, браузеру і кілька допоміжних функцій (рис. 1).

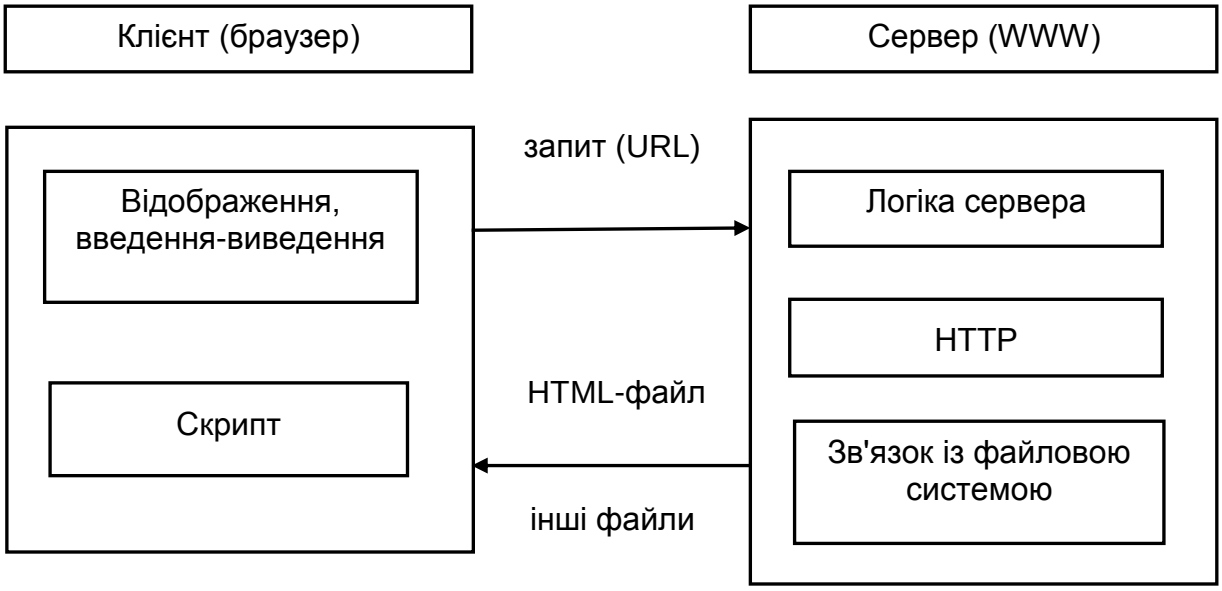


Рис. 1. **Взаємодія клієнта із сервером**

За такої взаємодії сервер грає роль сховища. Для розширення можливостей WEB-сторінок із реалізації більш складних функцій необхідно розширювати логіку сервера за рахунок виконання спеціально розроблених програм. Ці програми можуть виконувати найрізноманітніші функції: оброблення сторінок перед відправкою, зв'язок із базами даних тощо. У зв'язку із цим будемо розуміти під серверними технологіями сукупність засобів і методів запуску виконуваних програм на сервері.

Для забезпечення функціонування сервісу WWW було розроблено досить багато серверів. Вони володіють різними характеристиками, але найбільш масовими є два: IIS (Internet Information Services), розроблений у компанії "Майкрософт", і вільно поширюваний Apache.

IIS (Internet Information Services) – пропріетарний набір серверів для декількох служб інтернету від компанії Майкрософт. IIS функціонує тільки під управлінням ОС сімейства Windows. IIS підтримує протоколи HTTP, HTTPS, FTP, POP3, SMTP, NNTP. Один сервер IIS може обслуговувати кілька сайтів (сервер з однією IP-адресою може обслуговувати на одному TCP-порті кілька сайтів). Для кожного сайта вказується домашній каталог – каталог файлової системи сервера, відведений для сайта. Використовувані версії – 5.0, 6.0, 7.0. Налаштування здійснюється за допомогою консолі управління.

Apache – багатоплатформовий WWW-сервер, функціонує в середовищах операційних систем GNU / Linux, BSD, Mac OS, Microsoft Windows, Novell NetWare, BeOS. Існує кілька різних версій, збірок тощо. Основними достоїнствами Apache вважають надійність і гнучкість конфігурації. Як недолік найбільш часто називається відсутність зручного стандартного інтерфейсу для адміністрування. Система конфігурації Apache заснована на текстових конфігураційних файлах. Має три умовних рівня конфігурації: конфігурацію сервера, конфігурацію віртуального хоста, конфігурацію рівня директорії.

Програми, які розробляються для виконання на сервері крім реалізації основних функцій, повинні мати кілька особливостей (порівняно з тими, що виконуються в середовищі операційної системи):

запуск здійснюється сервером;

має бути забезпечена можливість взаємодії із сервером;

має бути забезпечена можливість відправляти результати (дані) клієнту, який видав запит (браузеру).

Існує декілька технологій створення і використання таких програм, що відрізняються мовами, засобами взаємодії із сервером, ступенем стандартизації та поширеності. До теперішнього часу склалося два підходи щодо їх створення.

Перший – це розробка звичайної програми, що виконується, яка взаємодіє із сервером і вихідні дані якої (HTML-документ) направляється через сервер клієнту. Він отримав назву серверних розширень або розширень WEB-сервера, і використовується для розширення можливостей сервера, створення складних великих додатків.

Другий – вбудовування програмного коду (скрипта) безпосередньо в HTML-сторінку, яка зберігатися на сервері. Такі сторінки отримали назву активних серверних сторінок.

Для створення серверних розширень було розроблено кілька інтерфейсів (специфікацій на взаємодію між сервером та програмою), найбільшого поширення набули: CGI, FCGI і API-інтерфейси. Кожен із них має свої переваги, так само, як і певні недоліки.

Специфікація CGI (Common Gateway Interface) передбачає такий вид взаємодії. Браузер відправляє запит серверу. Як URL в запиті використовується адреса CGI-програми. Сервер визначає тип URL, і породжує процес-нащадок, передає в нього значення змінних оточення, запускає CGI-програму і передає їй управління. CGI-програма робить оброблення інформації за певним алгоритмом і повертає результат обчислень через свій стандартний вихід у формі, визначеній у специфікації CGI.

Після цього процес, в якому виконувалася програма, завершується. Взаємодія браузера, сервера і програми в цьому випадку носить разовий характер і за своєю природою не орієнтована на тривалий діалог.

Основні переваги такого рішення: простота, незалежність від мови, виконання в окремому адресному просторі, відкритий стандарт, архітектурна незалежність.

Головним недоліком є низька швидкість реакції на запит користувача і велике навантаження на сервер (багато запитів – багато процесів, а значить – багато ресурсів).

API-інтерфейси – це розширення, що забезпечують можливість доступу до функцій сервера з програми. Зараз вони розроблені для різних серверів. Зокрема, ISAPI (API для IIS) і API для Apache. Перевага цього механізму полягає у високій швидкості, можливості використання для оброблення даних серверних функцій. Серйозний недолік API – це його складність. Для написання додатка необхідні глибокі знання архітектури сервера та її реалізації. Виконується програма в адресному просторі сервера. Це означає, що збій у додатку може привести до збою в роботі сервера. Написаний для певного сервера додаток не може бути перенесено на іншу платформу.

FAST CGI – ця специфікація була розроблена для IIS з метою поєднання переваги звичайного CGI і API. Для цього серверна програма завантажується для виконання або в момент старту сервера, або на вимогу клієнта і залишається в пам'яті так довго, як це необхідно для обслуговування запитів. Fast CGI не тільки дозволяє збільшити продуктивність додатків, але дозволяє включити додаткове оброблення інформації, що відправляється клієнтові, і розширює можливості сервера.

Знання інтерфейсів необхідні не тільки для розроблення серверних програм, але і під час вибору технології та налаштування сервера.

Другий підхід до створення серверних додатків (включення коду в саму HTML-сторінку) полягає в тому, що запитувана сторінка з розміченим текстом містить програмний код, який виконується на сервері перед відправкою клієнту, змінюючи запитувану сторінку. Цей код може витягувати дані з бази даних, будувати призначений для користувача інтерфейс залежно від отриманих даних, а потім відправляти HTML-сторінку назад клієнту. Кінцевим результатом такого процесу буде відправка браузеру стандартної WEB-сторінки на HTML. Цей процес являє собою динамічне створення WEB-сторінок. Сторінка, яка показується в браузері, насправді є продуктом виконання коду, який був запущений після клієнтського запиту, на сервері такої сторінки фізично не існує.

Таким чином, обидва підходи дозволяють розширити можливості сервера аж до створення WEB-додатка, тобто розподіленого додатка, що використовує для обміну даними протокол HTTP. Перевагою такого додатка є стандартний, всім відомий клієнт-браузер, і зниження витрат на супровід додатків розробниками. Замість того, щоб встановлювати додаток Windows на кожне робоче місце в компанії, можна створити один WEB-додаток, до якого є загальний доступ. Якщо виникає необхідність внесення в цей додаток якихось змін або виправлень, то досить оновити додаток тільки на сервері, а не на всіх клієнтських комп'ютерах. Це також дозволяє кожному клієнту постійно користуватися самою останньою версією програми, оскільки саме вона завжди буде знаходитися на сервері.

Незважаючи на те, що мова йде, по суті, про розподілений додаток, користувач, який працює в браузері (набираючи URL в рядку адреси або клацаючи по посиланнях), може цього не помічати. У зв'язку із цим додатки, що складаються з активних сторінок, продовжують називати сайтами.

1.2. Використання CGI-інтерфейсу

Найпростішим та історично першим засобом створення динамічних ресурсів для мережі Інтернет є використання інтерфейсу CGI. Розроблено він був на самому початку появи мережі Інтернет, і продовжує використовуватися досі.

1.2.1. Особливості інтерфейсу CGI

Common Gateway Interface (CGI) є стандартом інтерфейсу зовнішньої прикладної програми з WWW-сервером. Їхня взаємодія наведена на рис. 2.

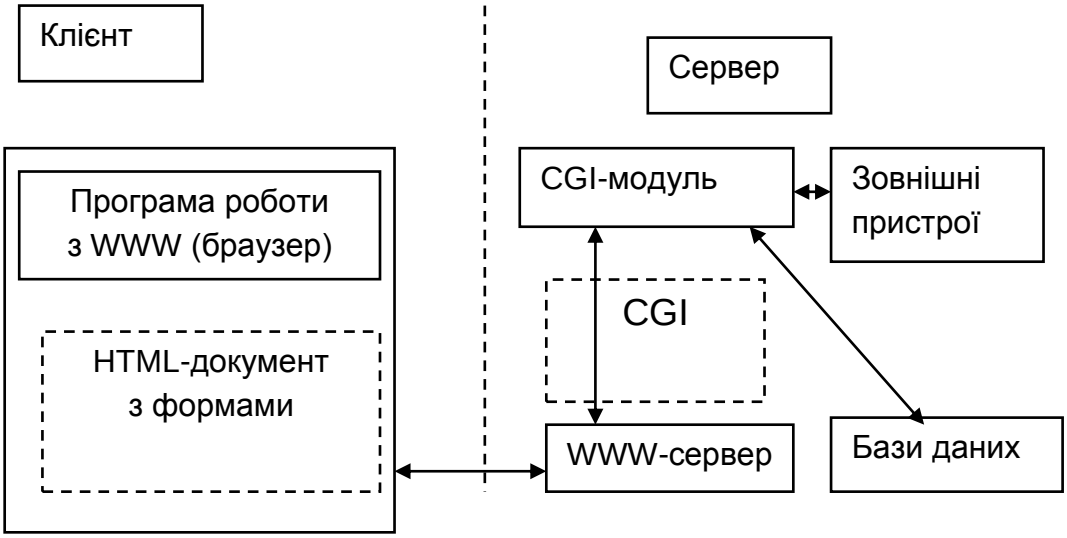


Рис. 2. **Взаємодія між клієнтом та сервером із використанням інтерфейсу CGI**

Серверна частина складається з виконуваної програми, яка вирішує основні завдання оброблення даних, що поступають від клієнтської частини. Після оброблення формується відповідь у форматі HTML і т. д. Таку програму називають ще CGI-модулем.

Клієнтською частиною додатку є HTML-документ, який формує серверна програма. У ньому реалізований інтерфейс із користувачем. Реалізується він в основному за допомогою форм, хоча можуть використовуватися й інші елементи, а також скрипти.

Інтерфейс CGI є набором правил взаємодії сервера із зовнішній відносно до нього програмою (CGI-модулем). Він визначає чотири інформаційні потоки (рис. 3):

1) змінні оточення;
2) стандартний вхідний потік;
3) стандартний вихідний потік;
4) командний рядок.

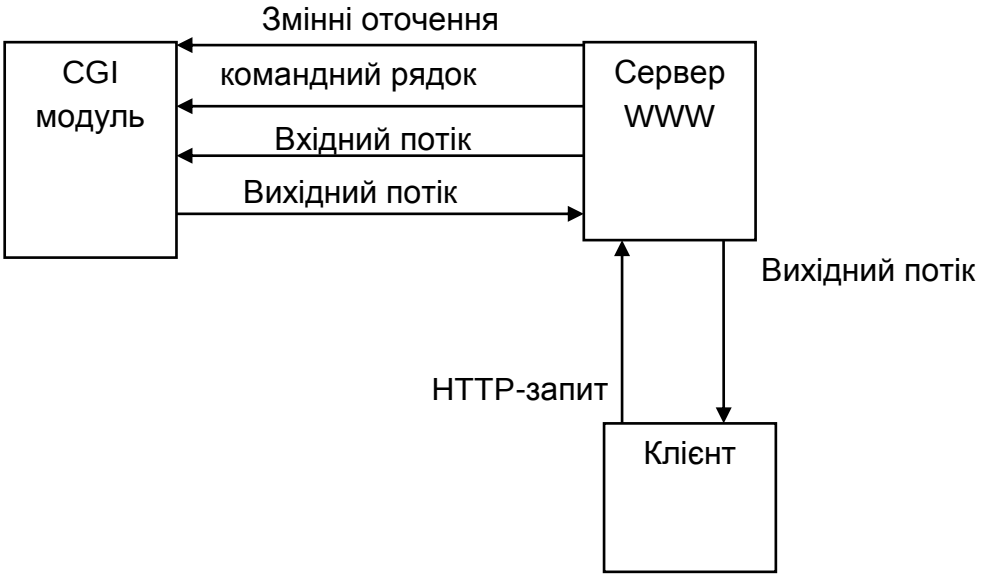


Рис. 3. **CGI-інтерфейс**

Змінні оточення в середовищі будь-якої ОС слугують для зв'язку між процесами, вони є набором спеціальних змінних, значення яких встановлюються перед запуском програми і доступні програмі в ході виконання. Склад цих змінних та їхні імена залежать від процесу, що запускає дану програму. У цьому випадку таким процесом є WWW-сервер.

Цих змінних доволі багато, вони містять різні дані, зокрема:

SERVER_SOFTWARE – дані про WWW сервер (назва / версія);

CONTENT_LENGTH – довжина стандартного вхідного потоку в символах;

CONTENT_TYPE – тип даних, які передані;

REQUEST_METHOD – метод запиту, який був використаний "POST", "GET", "HEAD" і т. д.;

PATH_INFO – значення змінної містить отриманий від клієнта віртуальний шлях до CGI-модуля;

PATH_TRANSLATED – фізичний шлях до CGI-модуля, перетворений із значення PATH_INFO;

QUERY_STRING – рядок символів наступних за знаком "?" у URL відповідному даному запиту;

HTTP_USER_AGENT – назва програми перегляду, яку використовує клієнт під час відправленні запиту.

Аргументи командного рядка. Окрім змінних оточення програмі, що запускається, передається список параметрів, який поміщається в командний рядок під час запуску. CGI-модуль у командному рядку від сервера отримує:

залишок URL після імені CGI-модуля як перший параметр (перший параметр буде порожній, якщо було присутнє лише ім'я CGI-модуля);

список ключових слів як залишок командного рядка для скрипта пошуку або імена полів форми з доданим знаком рівності та відповідних значень змінних.

Стандартний вхідний потік. Склад даних, що направляються сервером у вхідний потік CGI-модуля, залежить від використаного клієнтом методу запиту.

У разі методу запиту POST дані передаються як вміст HTTP запиту в такій формі:

name=value&name1=value1&...&nameN=valueN,

де name – ім'я змінної;

value – значення змінної;

N – кількість змінних.

Водночас встановлюються значення змінних оточення CONTENT_LENGTH і CONTENT_TYPE.

Таким чином, якщо в результаті використання форми затрибутом METHOD="POST" сформовано рядок даних firm=MMM&price=100023, то сервер встановить значення CONTENT_LENGTH, яке дорівнюватиме 21 та CONTENT_TYPE в application/x-www-form-urlencoded. А в стандартний потік введення буде посилатися блок даних.

У разі запиту за методом GET, рядок даних передається як частина URL. Наприклад: http://host/cgi-bin/script?name1=value1&name2=value2.

У цьому випадку змінна оточення QUERY_STRING набуває значення name1=value1&name2=value2, а у вхідний потік нічого не посилається.

Вихідний потік. CGI-модуль виводить інформацію в стандартний вихідний потік. Це може бути або документ, що згенерував CGI-модуль, або інструкція сервера, де отримати необхідний документ.

Виведення CGI-модуля повинне починатися із заголовка, що містить певні рядки і закінчуватися двома символами CR (\n, код 0x10). Ці ряд-

ки сприймаються як директиви сервера. Рядки, що не є директивами сервера, посилаються безпосередньо клієнтові. CGI специфікація визначає три директиви сервера:

Content-type – MIME або тип документа, що повертається;

Location – указує серверу, що повертається не сам документ, а посилання на нього;

***Status** – задає серверу HTTP/1.0 рядок-статус, в якому кодується різна інформація (наприклад, про успішне завершення операції).

У разі відправлення клієнтові HTML-документа директива має вигляд **Content-type: text/html \n\n**.

Якщо буде переданий URL, то директива виглядає, наприклад, так **Location: http://host/file.htm**, і сервер поверне клієнтові заданий цим шляхом документ, неначебто клієнт запрошував цей документ безпосередньо.

1.2.2. Розроблення CGI-програм

У разі взаємодії WWW-клієнта із сервером на сервер відправляються запити різного вигляду. Найчастіше це запит деякого ресурсу і відправка даних із форм.

Запит ресурсу – це URL, що ідентифікує ресурс. Отримавши запит, сервер шукає запрошуваний файл і відправляє його клієнтові.

Якщо запит ідентифікує CGI-програму, то вона не відправляється на машину клієнта, а запускається на виконання на машині сервера.

CGI-програми на сервері можуть розміщуватися в папці з іншими ресурсами сайта. Хоча деякі сервери вимагають їх розміщення в каталозі зі стандартизованим ім'ям (наприклад, cgi-bin).

Шлях до CGI-програмі включається в URL. Але водночас необхідно обов'язково вказувати і розширення файла (тобто exe). Наприклад: http://www.orp.com/mucat/myscript.exe. Оскільки операція запуску виконуваної програми на машині сервера є потенційно небезпечною, то для більшості серверів (наприклад, IIS 7) може знадобитися завдання в налаштуваннях спеціальних дозволів на виконання із зазначенням імені програми.

Використовуватися посилання на CGI-програми можуть у будь-якому атрибуті, який допускає URL (href, а також action з тегу form), можна також просто набрати URL програми в адресному рядку браузера.

Водночас після самої адреси можуть розміщуватися дані, які будуть передані серверу разом із URL. Дані починаються із символів, які знаходяться за знаком "?", наприклад,

визів CGI.

У цьому випадку серверу із браузера буде відправлений такий рядок /mycgi/cd2.exe?date. Сервер передає цей рядок CGI-програмі через змінні оточення (у PATH_INFO буде URL, а в QUERY_STRING – рядок date).

Після отримання і розшифровки запиту виконується динамічне формування HTML-документа (оформлення результату роботи модуля) або, наприклад, таблиці вибірки з бази даних. Для цього CGI-модуль повинен видати у стандартний вихідний потік заголовок, що складається з рядка: **Content-type: text/html** і порожнього рядка (двох символів **CR**).

Після цього заголовка може йти будь-який текст у форматі HTML (або іншої мови, яка може інтерпретуватися браузером, наприклад, XML).

Під час оброблення CGI-програмою запиту з даними необхідно враховувати методи передавання (GET або POST). У форми метод передавання задається атрибутом metod.

У разі методу GET дані додаються до URL (як і в посиланнях) і передаються CGI-програмі через змінні оточення. У разі POST – через вхідний потік.

Послідовність дій під час використання методу GET:

набути значення змінної QUERY_STRING;

декодувати імена та їх значення (враховуючи, що всі пропуски сервером були замінені символом "+" і всі символи з десятковим кодом більше 128 перетворені в символ "%" і наступним за ним шістнадцятирічним кодом символу);

сформувати структуру відповідності "ім'я – значення" для подальшоговикористання в CGI-модулі.

Послідовність дій під час використання методу POST:

отримати із стандартного вхідного потоку CONTENT_LENGTH символів;

декодувати імена та їх значення (враховуючи, що всі пропуски сервером були замінені символом "+" і всі символи з десятковим кодом більше 128 перетворені в символ "%" і наступним за ним шістнадцятирічним кодом символу);

сформувати структуру відповідності "ім'я – значення" для подальшого використання в CGI-модулі.

Очевидно, що відмінність є лише в джерелі даних. Тому можливе створення єдиного модуля для методів POST і GET. Необхідно додати в початок програми перевірку значення змінної REQUEST_METHOD для визначення методу запиту, отримати дані. Після декодування и формування структури "ім'я – значення" можна приступити до вирішення завдань, заради яких, власне, створювався CGI-модуль. Зрозуміло, що завдання, що вирішуються CGI-модулем, можуть бути дуже різноманітними (отримання і оброблення пошти, доступ до баз даних, гостьова книга тощо).

Мовою написання CGI-програм може бути будь-яка, проте під час використання мов із транслятором інтерпретуючого типу (Perl, PHP, VBAScript та ін.) сервер повинен містити відповідний інтерпретатор (зазвичай у вигляді модуля, що підключається). Під час використання мови з компілятором (Pascal, C, Java), виконуваний модуль повинен створюватися в консольному режимі.

Приклад 1. Сторінка зі зверненням до CGI-програми, яка у відповідь видає сторінку з вітанням.

Сторінка зі зверненням містить різні варіанти виклику: посилання, форму з атрибутом action. Код HTML-сторінки:

```
<!DOCTYPE HTML>
<html>
    <head>
        <title>звернення до CGI-програми</title>
    </head>
    <body>
    <a href="con1.exe">Звернення до CGI-програми</a>
    <form action="con2.exe">
    <input type=submit value="Виконати CGI-програму">
    </form>
    </body>
</html>
```

Зовнішній вигляд сторінки у вікні браузера наведено на рис. 4.

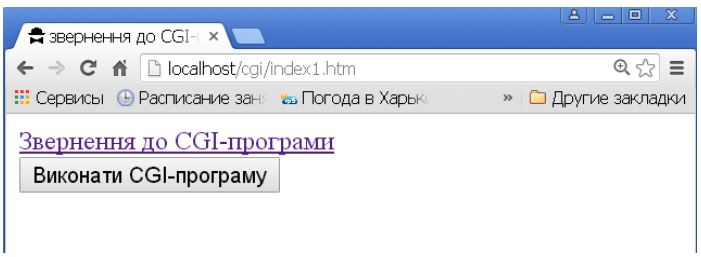


Рис. 4. **Вигляд сторінки у вікні браузера**

Розроблення програми доцільно виконувати в середовищі Visual Studio в режимі створення консольного додатку. Для розроблення програми використана мова C#. Код програми:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace ConsoleApplication1
{
   class Program
   {
      static void Main(string[] args)
      {
         Console.Write("Content-Type:text/html\n\n");
         Console.WriteLine("<html>");
         Console.WriteLine("<head>");
         Console.WriteLine("<title>CGI-програма</title>");
         Console.WriteLine("</head>");
         Console.WriteLine("<body>");
         Console.WriteLine("<h1>Вас вітає найпростіша CGI-програма!</h1>");
         Console.WriteLine("</body>");
         Console.WriteLine("</html>");
}  }  }
```

Зовнішній вигляд відповіді у вікні браузера наведено на рис. 5.

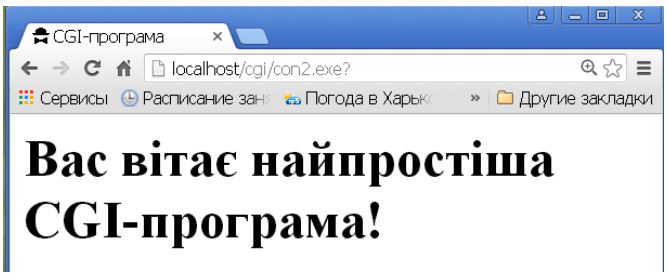


Рис. 5. **Вигляд відповіді у вікні браузера**

Наведений приклад досить простий, але дає уявлення про розроблення WEB-додатків із використанням інтерфейсу CGI та взаємодію браузера, сервера и CGI-програми. Програма, яка повинна отримати від клієнта дані для оброблення, виглядає дещо складніше за рахунок того, що потрібно виконувати розбір отримуваних рядків.

Тема 2. Мова розмітки XML

2.1. Загальна характеристика мови XML

Призначення мов розмітки полягає в описі структурованих документів. Кількість і різноманітність необхідних мов розмітки визначається кількістю методів обробки даних. Саме тому і була запропонована XML – мова, використовуючи яку можна створювати свою власну мову розмітки для конкретних застосувань.

2.1.1. Структура XML-документів

XML-документ складається з трьох частин: прологу (може бути порожнім), кореневого елемента і необов'язкової загальної частини.

Пролог може містити XML-оголошення (наприклад <?xml version = "1.0"?>), оголошення типу документа або DTD (<!DOCTYPE .>), інструкції з оброблення, коментарі і пропуски. Рекомендується включати в пролог, як мінімум, XML-оголошення, в якому указується вживана версія мови XML.

Кореневий елемент містить теги елементів, склад яких відповідає предметної області. XML-документ повинен містити тільки один кореневий елемент.

Необов'язкова загальна частина може складатися з XML-коментарів, інструкцій з оброблення і пропусків (пропуски, табуляції).

Оголошення знаходиться в першому рядку в елементі <?xml...?>, наприклад <?xml version = "1.0" standalone="yes" encoding="UTF-8"?>

У XML-оголошенні допускається вказівка таких атрибутів:

version – версія XML-оголошення;

encoding – кодування символів документа (за замовчуванням передбачається UTF-8);

standalone – "yes", якщо цей документ не посилається на зовнішні об'єкти,"no" – інакше, (за замовчуванням – "yes").

Коментарі схожі на HTML-коментарі. Їх можна використовувати для включення в документ пояснень, які ігноруються під час синтаксичного розбору. Коментар поміщається між символами <!-- ... -->.

Коментарі не повинні розміщуватися перед XML-оголошенням. Коментар не можна поміщати усередині символів розмітки (тегів). Неприпустимо використовувати символ "--" усередині коментаря.

Інструкції з оброблення управляють функціонуванням XML-процесора. Вони починаються з послідовності символів <?, і завершуються символами ?>. Не можна використовувати тег <?xml … ?> (це оголошення). Інструкції з оброблення залежать від використовуваного процесора, а не є специфікаціями мови XML. У загальному випадку, інструкції можуть бути поміщені в будь-яке місце документів поза елементами.

Наприклад, інструкція для підключення таблиць стилів:

<?xml-stylesheet type="text/css" href="my .css" ?>.

Ця інструкція буде правильно виконана всіма браузерами, які підтримують XML.

Структура XML-документа наведена на рис. 6.

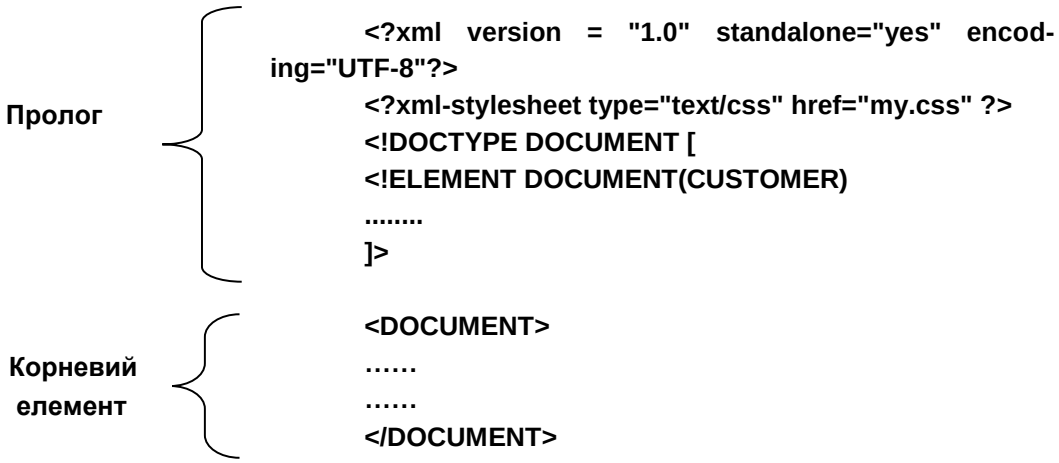


Рис. 6. **Структура XML-документа**

2.1.2. Правила створення XML-документів

Правила розмітки XML багато в чому схожі на HTML. Головна відмінність полягає в тому, що імена тегів, їх значення і зміст задаються розробником відповідно до предметної області конкретного завдання.

Теги виділяються символами < і >. Кінцеві теги починаються символами </. Одиночні (непарні) теги, які не мають змісту (порожні), повинні закінчуються символами "/>". Наприклад, <Greeting />.

Теги можуть вкладатися один в одного без "перетину" (якщо елемент починається усередині іншого елемента, він повинен і закінчуватися усередині цього елемента).

Імена (назви) тегів повинні починатися з букви, символу підкреслення або двокрапки, і містити букви, цифри, символи підкреслення, дефіси, крапки і двокрапки (але не пропуски). Хоча в рекомендаціях XML1.0 явно це не указується, слід уникати використання двокрапки в іменах тегів, оскільки двокрапка використовується у XML в разі вказівки просторів імен. Імена задаються з урахуванням регістра. <DOCUMENT> і <document> це різні теги.

Слід особливо підкреслити, що не забороняється використання кирилиці та інших національних алфавітів.

Початкові і порожні теги можуть містити атрибути, що дозволяють визначити додаткові дані, вони задаються як і в HTML-розмітці (ім'я_атр="значення_атр"). Наприклад, <birthday year="1983">Степаненко </birthday>.

Як початковий символ в імені атрибута може застосовуватися буква, знак підкреслення або двокрапка, далі можуть йти букви, цифри, символи підкреслення, дефіси, крапки і двокрапки (не допускається використання пропусків, оскільки з їх допомогою розділяються атрибути).

У XML не допускаються атрибути, яким не привласнені які-небудь значення.

Атрибути, як і інші елементи розмітки, представлені символами. Навіть якщо атрибуту привласнюється числове значення, воно трактується у вигляді текстового рядка (поміщається в лапки):

<circle origin_x="10.0" origin_y " 20.0" radius="10.0" />.

Лапки застосовуються подвійні або одинарні. за необхідності задати текст зі складнішим поєднанням спеціальних символів можна використовувати підстановки (як і в HTML), тут вони називаються об'єктними посиланнями (&apos – символ **'**, &quot – символ **"**, &amp – символ **&**, &lt – символ **<**, &gt – символ **>**, $ &#x1a; і т. п.).

Для виключення з процесу синтаксичного аналізу великих фрагментів коду використовуються спеціальні секції <![CDATA [....]]> Усе, що знаходиться усередині секції не аналізується. Наприклад, це може бути код скрипта (а у нього свої правила коректності) тощо.

Оскільки користувач сам визначає імена тегів і зміст елементів, то одні і ті самі дані можна розмістити або в тегах, або в атрибутах.

Твердо сталого правила немає, проте занадто велика кількість атрибутів призводить до того, що документ важко читати.

Приклад 2. Створено XML-документ із прологом і кореневим елементом:

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/css" href="my .css" ?>
<!DOCTYPE DOCUMENT [
<!ELEMENT DOCUMENT (ZAG , STUDENT* )>
<!ELEMENT ZAG (#PCDATA )>
<!ELEMENT STUDENT (NAME , photo,BOOK* )>
<!ELEMENT NAME (#PCDATA )>
<!ELEMENT photo (#PCDATA )>
<!ELEMENT BOOK (AUTHOR,TITLE )>
<!ELEMENT AUTHOR (#PCDATA )>
<!ELEMENT TITLE (#PCDATA )>
]>
<DOCUMENT >
<ZAG>Список студентів, що не здали книги</ZAG>
<STUDENT >
  <NAME>Иванов</NAME>
  <photo>im1.gif</photo>
  <BOOK >
   <AUTHOR>Пушкар </AUTHOR>
   <TITLE > Основи наукових досліджень</TITLE>
  </BOOK >
  <BOOK >
    <AUTHOR>Молчанов </AUTHOR>
    <TITLE > Технології Web-дизайну</TITLE>
   </BOOK >
 </STUDENT >
 </DOCUMENT >
```

Пролог цього документа містить оголошення (перший рядок), інструкцію з оброблення (другий рядок) і DTD (опис типу документа, починається з <!DOCTYPE …>, та описує допустиму вкладеність тегів і їх вміст).

Інструкція забезпечує підключення таблиці стилів із файла my.css.

Кореневий елемент (<DOCUMENT>...</DOCUMENT>) містить теги з даними про студентів, що не здали книги (для стислості поміщений тільки один тег <STUDENT>).

Таким чином мова XML є засобом створення мов розмітки для різних предметних областей. Такі мови отримали назву додатків XML. Вони описують структуровані документи різного призначення. Документи можуть перевірятися, накопичуватися, оброблятися, пересилатися між додатками. Для цього технологія передбачає додаткові засоби.

2.2. Застосування XML

Мова XML є мовою розмітки, імена тегів у якій вибираються розробником з урахуванням конкретної предметної області. Завдяки цьому можна використовувати свою розмітку для кожного застосування. Це забезпечує надзвичайну гнучкість. Крім того, XML і технології, які з нею пов'язані, забезпечують перевірку документів, зручний механізм їх відображення і перетворення. Розглянемо як реалізується це на практиці.

2.2.1. Перевірка XML-документів

Важливою можливістю технології XML є можливість перевірки документа на відповідність формальним критеріям відсутності помилок. Існують два критерії для перевірки: коректність (правильність) і валідність (дієвість).

Документ є формально коректним, якщо він задовольняє загальні вимоги до XML-документів (п.п. 1.2.1.2).

Під валідністю розуміється відповідність документа синтаксису, який визначає розробник.

Перевірка коректності повинна забезпечити відповідність всього документа основним вимогам синтаксису XML. Водночас перевіряється, чи немає незакритих тегів, чи всі атрибути поміщені в лапки, чи немає тегів із порушенням вкладеності, чи має документ єдиний кореневий елемент і т. п. Дана перевірка є обов'язковою для всіх XML-документів. Якщо документ не є коректним, подальше оброблення повинно припинятися. Само так і працює більшість браузерів.

Валідність документа – це відповідність синтаксису, який задав розробник. До цього належать конкретний набір тегів, їх вкладеність, імена атрибутів, їх можливі значення тощо. Якщо документ не є валідним, то його оброблення може бути продовжено, а реакція залежить від засобу (програми) оброблення. Це може бути попередження, повідомлення про помилку.

Для перевірки валідності документ повинен містити формальний опис синтаксису, визначеного розробником. Перевірка валідності можлива лише для тих документів, для яких заданий такий формальний опис.

Для опису синтаксису XML-документів нині створено дві мови: DTD (Document Type Definition) і XSD (XML Schema Definition). Обидві мови стандартизовано на рівні World Wide Web Consortium (W3C). Фактично XSD і DTD – це "граматика для певного класу документів".

Якщо порівнювати XSD і DTD, то мова XSD дозволяє задавати більш суворі обмеження на типи даних XML-документа, але є складнішою. Не дивлячись на те, що схеми даних XSD самі по собі є документами XML, редагувати їх за допомогою звичайного XML-редактора виявляється незручним через їх достатню складність. Тому, як правило, для їх створення використовуються спеціальні редактори, що відображають їх у наглядному графічному вигляді. Саме тому ми розглянемо DTD.

Оголошення типу документа (DTD) визначає структуру документа та зміст його елементів. Кожен елемент і атрибут, який використовується, повинні відповідати специфікації DTD, записаній у відповідному оголошенні.

Оголошенням типу документа (DTD) є блок XML-розмітки, який може розташовуватися в будь-якому місці прологу – поза іншою розміткою, після XML-оголошення.

DTD може міститися в пролозі XML-документа (внутрішнє) або в зовнішньому файлі, ім'я якого вказується в документі (зовнішнє). Для внутрішнього оголошення це виглядає наступним чином:

```
...
<! DOCTYPE rootname [
<!ELEMENT rootname (contacts, issues, authors )>
...
]>
...
```

Або таким чином для зовнішнього:

```
<?xml version=1.0 standalone="no " ?>
<! DOCTYPE rootname SYSTEM "fileDTD.dtd">
...
```

Можна одночасно використовувати внутрішні і зовнішні визначення DTD за допомогою елемента <!DOCTYPE> такого вигляду:

```
!DOCTYPE rootname SYSTEM " fileDTD.dtd "
 […
]>
```

Зазвичай внутрішні визначення доповнюють зовнішні.

Частіше за все DTD містить такі типи оголошень розмітки:

оголошення типів елементів, що визначають типи елементів, які може містити документ, а також вміст і порядок проходження елементів;

оголошення списків атрибутів, кожне оголошення яких задає імена атрибутів, які можуть бути використані з певним типом елемента, а також типи даних і встановлювані за замовчуванням значення цих атрибутів.

Оголошення типів елементів. У валідному XML-документі необхідно повністю оголосити тип кожного елемента, який використовується. Оголошення типу елемента указує на ім'я типу елемента, допустимий вміст елемента і порядок розміщення дочірніх елементів. Як єдине ціле, оголошення типів елементів у DTD задає повну логічну структуру документа.

Оголошення типу елемента має таку узагальнену форму:

<!ELEMENT Ім'я опис_вмісту>

Тут "Ім'я" це ім'я оголошуваного типу елемента. А "опис_вмісту" визначає, що може містити елемент. Оголошувати певний тип елемента в даному документі можна тільки один раз.

Наприклад, оголошення типу елемента з ім'ям TITLE, для вмісту якого можуть використовуватися тільки символьні дані (дочірні елементи не допускаються):

```
<!ELEMENT TITLE (#PCDATA )>
```

#PCDATA (parseable character data) – будь-яка інформація, з якою може працювати програма-аналізатор. Це текст, що не є розміткою. Його називають розібраними символьними даними.

Крім #PCDATA "опис_вмісту" може містить один із чотирьох типів вмісту: EMPTY (елемент має бути порожнім – тобто не може мати вмісту), ANY (будь-який, елемент цього типу може містити або не містити дочірні елементи, мати або не мати символьних даних), дочірній і змішаний вміст.

Якщо елемент має дочірній вміст, він може безпосередньо містити лише певні дочірні елементи, але не символьні дані. У тексті документа можна розділяти дочірні елементи символами пропуску, табуляції, повернення каретки або переведення рядка, щоб поліпшити сприйняття документа людиною, але процесор ігноруватиме ці символи і не даватиме їх додатку.

Приклад XML-документа, що описує одну книгу:

```
<?xml version="1.0" encoding="windows-1251 " ?>
<!DOCTYPE BOOK
        [
        <!ELEMENT BOOK (TITLE, AUTHOR )>
        <!ELEMENT TITLE (#PCDATA )>
        <!ELEMENT AUTHOR (#PCDATA )>
        ]
>
<BOOK >
        <TITLE>The Scarlet Letter</TITLE>
        <AUTHOR>Nathaniel Hawthorne</AUTHOR>
</BOOK >
```

У цьому документі тип елемента BOOK оголошений таким, що має дочірній вміст: елементи TITLE і AUTHOR. У даному прикладі елемент BOOK повинен мати рівно один дочірній елемент TITLE, за яким іде рівно один дочірній елемент AUTHOR. Пропуск дочірнього елемента або використання одного і того самого типу дочірнього елемента більше одного разу також неприпустимо.

За необхідності можна вказати, що елемент може мати будь-який із серії допустимих дочірніх елементів, елементи розділяються символом "|". Наприклад, наступне DTD указує, що елемент FILM може складатися з одного дочірнього елемента STAR, або одного дочірнього елемента NARRATOR, або одного дочірнього елемента INSTRUCTOR:

```
<!DOCTYPE FILM
        [
        <!ELEMENT FILM (STAR | NARRATOR | INSTRUCTOR)>
        <!ELEMENT STAR (#PCDATA )>
        <!ELEMENT NARRATOR (#PCDATA )>
        <!ELEMENT INSTRUCTOR (#PCDATA )>
        ]
>
```

Будь-яку із цих форм вмісту можна змінювати, використовуючи символи: знак питання (?), знак плюс (+) і зірочка (*), значення яких описані в табл. 1.

Таблиця 1

Символи модифікації

Символ	Значення
?	Жоден або один із попередніх елементів
+	Один або більше з попередніх елементів
*	Жоден або будь-яка кількість з попередніх елементів

Наприклад, наступне оголошення означає, що можна включити один або більше дочірніх елементів NAME, і що дочірній елемент HEIGHT є не обов'язковим:

```
<!ELEMENT MOUNTAIN (NAME+, HEIGHT ?, STATE )>
```

Таким чином, такий елемент буде валідним:

```
<MOUNTAIN >
   <NAME>Pueblo Peak</NAME>
   <NAME>Taos Mountain</NAME>
   <STATE>New Mexico</STATE>
</MOUNTAIN >
```

Символи модифікації (?, +, *) можна помістити після дужок. Наприклад, наступне оголошення допускає включення одного або декількох дочірніх елементів будь-якого із цих трьох типів у будь-якому порядку:

```
<!ELEMENT FILM (STAR | NARRATOR | INSTRUCTOR )+>
```

Таке оголошення робить валідним всі наступні елементи:

```
<FILM >
   <NARRATOR>Bertram Wooster</NARRATOR>
   <STAR>Sean Connery</STAR>
   <NARRATOR>Plug Basham</NARRATOR>
</FILM >
<FILM >
   <STAR>Sean Connery</STAR>
   <STAR>Meg Ryan</STAR>
</FILM >
<FILM >
```

```
   <INSTRUCTOR>Stinker Pike</INSTRUCTOR>
</FILM >
```

Для формування більш складнішого вмісту можна вкладати одну модель в іншу. Наприклад, наступне DTD задає, що кожен елемент FILM повинен мати один дочірній елемент TITLE, за ним повинен слідувати один дочірній елемент CLASS, після нього повинні йти один із дочірніх елементів STAR, NARRATOR або INSTRUCTOR.

```
<!DOCTYPE FILM
        [
        <!ELEMENT FILM (TITLE, CLASS, (STAR | NARRATOR | INSTRUCTOR ) )>
        <!ELEMENT TITLE (#PCDATA )>
        <!ELEMENT CLASS (#PCDATA )>
        <!ELEMENT STAR (#PCDATA )>
        <!ELEMENT NARRATOR (#PCDATA )>
        <!ELEMENT INSTRUCTOR (#PCDATA )>
        ]
>
```

Якщо елемент має змішаний вміст, він може включати і дочірні елементи, і символьні дані. У разі змішаного вмісту можна задавати типи дочірніх елементів, але не порядок або кількість входжень.

Щоб оголосити тип елемента, який може містити символьні дані і на додаток жодного або декілька дочірніх елементів, кожен тип дочірнього елемента перераховується після ключового слова #PCDATA і розділяється символами "|". У кінці поміщається *. Кожне ім'я елемента може з'являтися в оголошенні тільки один раз.

Наприклад, наступне оголошення указує, що елемент TITLE може містити символьні дані, а також жодного або декілька дочірніх елементів SUBTITLE:

```
<!ELEMENT TITLE (#PCDATA | SUBTITLE )*>
```

Оголошення атрибутів. У валідному XML-документі необхідно оголосити всі атрибути, які передбачається використовувати для елементів. Атрибути, що асоціюються з певним елементом, визначаються за допомогою спеціального типу DTD-розмітки, званого оголошенням списку атрибутів. Воно:

визначає імена атрибутів, що асоціюються з елементом;

встановлює тип даних кожного атрибута;

задає обов'язковість для кожного атрибута.

Якщо атрибут необов'язковий, в оголошенні списку атрибутів указується, що повинен чинити процесор, якщо атрибут опущений. У оголошенні повинно, наприклад, міститися значення атрибута за замовчуванням, яке використовуватиме процесор.

Оголошення списку атрибутів має таку загальну форму:

<!ATTLIST Ім'я ВизАт>

Тут "Ім'я" є ім'я елемента, що асоціюється з атрибутом або атрибутами. "ВизАт" – це одне або декілька значень атрибутів, кожне з яких визначає один атрибут.

Визначення атрибута має таку форму запису:

Ім'я ТипАтр ОгУмов

Тут "Ім'я" є ім'ям атрибута.

"ТипАтр" є типом атрибута, тобто види значень, які можуть бути привласнені атрибута.

"ОгУмов" – це оголошення за замовчуванням, яке указує на обов'язковість атрибута і містить іншу інформацію.

Припустимо, оголошений тип елемента з ім'ям FILM таким чином:

<!ELEMENT FILM (TITLE (STAR NARRATOR INSTRUCTOR))>

Ось приклад оголошення списку атрибутів, яке описує два атрибути Class і Year для елемента FILM:

<!ATTLIST FILM Class CDATA "fictional" Year CDATA #REQUIRED >

На рис. 7 подано складові частини цього оголошення.

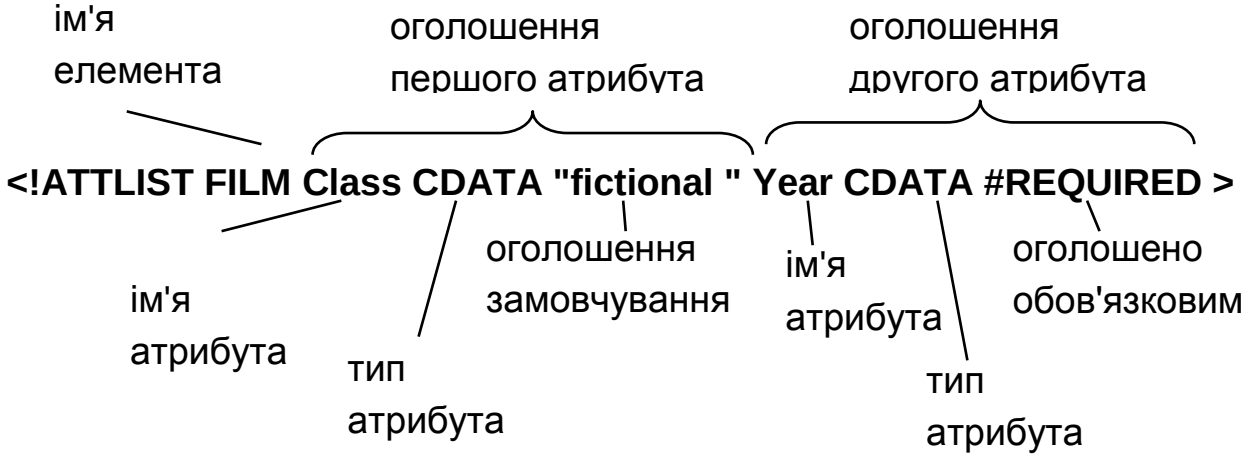


Рис. 7. **Складові частини оголошення атрибутів**

Атрибуту Class можна привласнити будь-який рядок у лапках (ключове слово CDATA). Якщо атрибут для певного елемента буде опущений, йому буде автоматично привласнено значення за замовчуванням "fictional". Атрибуту Year можна привласнити будь-який рядок у лапках; однак, це повинно бути зроблено обов'язково для кожного елемента FILM (ключове слово #REQUIRED), тому він не має значення за замовчуванням.

Наступний повний XML-документ містить оголошення списку атрибутів, а також елемент FILM:

```
<?xml version="1.0" encoding="windows-1251 " ?>
<!DOCTYPE FILM
   [
   <!ELEMENT FILM (TITLE, (STAR | NARRATOR | INSTRUCTOR ) )>
   <!ATTLIST FILM Class CDATA "fictional " Year CDATA #REQUIRED >
   <!ELEMENT TITLE (#PCDATA )>
   <!ELEMENT STAR (#PCDATA )>
   <!ELEMENT NARRATOR (#PCDATA )>
   <!ELEMENT INSTRUCTOR (#PCDATA )>
   ]
>
<FILM Year="1948 ">
   <TITLE>The Morning After</TITLE>
   <STAR>Morgan Attenbury</STAR>
</FILM >
```

Для елемента FILM атрибуту Year привласнено значення "1948". Атрибут Class опущений; проте, оскільки цей атрибут має значення за замовчуванням ("fictional"), його привласнюють атрибуту, неначебто його записали як значення атрибута.

Якщо використовують для одного елемента більше, ніж одне оголошення списку атрибутів, зміст оголошень об'єднується. Якщо атрибут із заданим ім'ям оголошений для одного і того самого елемента кілька разів, перше оголошення використовується, а подальші – ігноруються.

Усього існує декілька можливих типів значень атрибута:

CDATA – вмістом можуть бути будь-які символьні дані;

ID – визначає унікальний ідентифікатор елемента в документі;

IDREF (IDREFS) – указує, що значенням атрибута повинна виступати назва (або декілька таких назв, розділених пропусками) унікального ідентифікатора визначеного в цьому документі елемента;

NMTOKEN (NMTOKENS) – вмістом елемента може бути тільки одне окреме слово (тобто цей параметр є обмеженим варіантом CDATA);

список допустимих значень – визначається список значень, які може мати даний атрибут.

Також у оголошенні атрибута можна використовувати такі параметри:

#REQUIRED – визначає обов'язковий атрибут, який має бути заданий у всіх елементах даного типу;

#IMPLIED – атрибут не є обов'язковим;

#FIXED "значення" – указує, що атрибут повинен мати лише вказане значення, однак само визначення атрибута не є обов'язковим, але в процесі розбору його значення у будь-якому випадку буде передано програмі-аналізатору;

значення – задає значення атрибута за замовчуванням.

За наявності DTD для перевірки валідності можуть використовуватися універсальні програми-аналізатори. Крім того, різні застосування можуть спільно використовувати DTD для забезпечення єдиного погляду на XML-документ. У цьому випадку визначення повинні зберігатися окремо і бути доступними.

Аналізатори валідності вбудовані в більшість браузерів. Проте браузери, зазвичай, вимагають, щоб XML-документи були формально коректними, але не обов'язково дійсними (валідними). Вони не потребують обов'язкової наявності визначень DTD, але якщо ті присутні, то можуть бути використані в процесі перевірки XML-документа. Існують і спеціальні модулі перевірки валідності документів.

 Перелік деяких програм, які можна знайти в інтернеті:

http://validator.w3.org – офіційний модуль перевірки HTML-коду консорціуму W3C;

www.w3.org/People/Raggett/tidy – утиліта для очищення і відновлення Web-сторінок, що включають обмежену підтримку XML;

http://www.xml.eom/pub/a/tools/ruwf/check.html – модуль перевірки дійсності XML-коду групи XML.COM, заснований на процесорі Lark;

www.ltg.ed.ac.uk/~richard/xml-check.html – модуль перевірки XML-коду групи Language Technology Group Едінбурзького університету, заснований на синтаксичному аналізаторі RXP;

http://www.stg.brown.edu/service/xmlvalid – модуль перевірки дійсності XML-коду групи Scholarly Technology Group при університеті Брауна.

Модуль перевірки валідності входить у більшість програм для роботи з XML, зокрема, в редактор XML Copy Editor та ін.

Браузери також проводить синтаксичний розбір і перевірку валідності XML-документів. Проте результати не виводять. Їх можна вивести, звернувшись до відповідного об'єкта із скрипта.

2.2.2. Відображення XML-документів

Можна відкрити XML-документ безпосередньо в браузері. Останні версії цих програм підтримують XML і мають вбудовані засоби контролю правильності документа. Проте якщо XML-документ не містить зв'язку з таблицею стилів, то браузер лише позначає різні складові частини документа різним кольором, а також представляє кореневий елемент у вигляді ієрархічного дерева з можливістю згортання і розгортання структури і перегляду з меншою або більшою мірою деталізації (символи знаку мінус "−" або плюс "+" зліва від початкового тега).

Приклад 3. Створимо XML-документ зі списком літератури з тегами кирилицею, та розкриємо його в браузері.

Документ:

```
<?xml version="1.0" encoding="UTF-8"?>
<список> Список літератури з дисципліни "Засоби систем обробки даних"
 <джерело наявність="так">
  <автор>Холзнер С.</автор>
  <назва>Энциклопедия, 2-е изд.</назва>
  <видавництво>-СПб.:Питер</видавництво>
  <рік>2004</рік>
  <кіл_сторінок>1101</кіл_сторінок>
 </джерело>
 <джерело наявність="ні">
  <автор>Сакс Т., Мак-Клейн Г.</автор>
  <назва>Дизайн и архитектура современного веб-сайта. Опыт профессионалов.</назва>
  <видавництво>-М.: Издательский дом "Вильямс"</видавництво>
  <рік>2002</рік>
  <кіл_сторінок>304</кіл_сторінок>
 </джерело>
 <джерело наявність="так">
  <автор>Пушкарь А.И. та ін.</автор>
```

```
  <назва>Современные компьютерные технологии. Учебное пособие. /Под
ред. А.И. Пушкаря</назва>
  <видавництво>-ИД "ИНЖЕК"</видавництво>
  <рік>2004</рік>
  <кіл_сторінок>463</кіл_сторінок>
 </джерело>
</список>
```

Відображення в браузері надано на рис. 8.

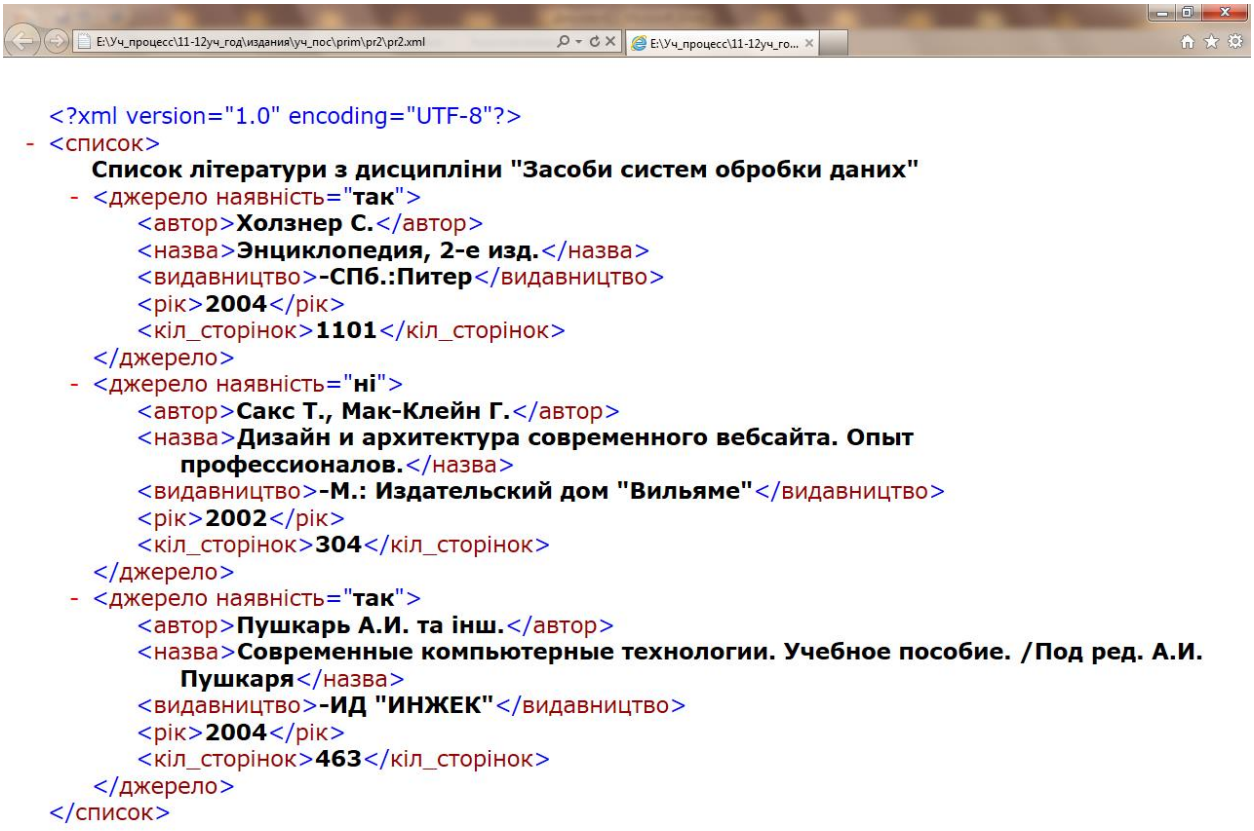


Рис. 8. **Відображення XML-документа в браузері**

Перш ніж браузер відображуватиме XML-документ, його вбудований синтаксичний XML-аналізатор (parser) прогляне вміст файла. Під час виявлення помилки буде відображена сторінка з відповідним повідомленням. Але за відсутності визначень DTD, перевіряється лише відповідність документа загальним формальним правилам побудови.

Отже, під час відкриття цих документів у браузері, ми побачимо структуру розміченого тексту (текст і теги різними кольорами).

Оскільки в XML розробник створює свої власні елементи, браузер не має вбудованих засобів, що дозволяють правильно відобразити їх.

Найбільш простим способом описати, як повинні відображатися елементи є створення таблиці стилів і скріплення її з XML-документом. Сучасні браузери забезпечують високий рівень підтримки каскадних таблиць стилів. Крім того, зберігання інструкцій з відображення окремо від самого XML-документа підвищує гнучкість використання і полегшує роботу з ним.

Нагадаємо, що таблиця стилів є текстовим файлом, зазвичай із розширенням .css, який містить набір правил, що складаються з селектора (у простому випадку це ім'я тегу) і декларацій (набір пар *ім'я_властивості: значення* у фігурних дужках), що повідомляють браузеру, яким чином форматувати і відображати елементи. Створюватися він може найпростішим текстовим редактором (наприклад, Блокнот). Більшість властивостей, які розглядалися під час форматування HTML, можуть бути застосовні і до XML-документів.

Приклад 4. Створимо таблицю стилів для відображення даних документа з прикладу 3.

Для відображення можна використовувати такі правила:

```
список  {display:block;font-size:36pt;text-align:center;margin-left:20px}
джерело {display:list-item; list-style-type:decimal;list-style-position:inside;font-size:20pt;text-indent:20px;text-align:left}
автор,назва,видавництво,рік,кіл_сторінок {display:inline;font-size:20pt; color:black}
```

Для встановлення зв'язку з таблицею стилів у XML-файл додано рядок <?xml-stylesheet type="text/css" href="pr3.css" ?>:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/css" href="pr3.css" ?>
<список> Список літератури з дисципліни " Засоби систем обробки даних"
 <джерело наявність="так">
  <автор>Холзнер С.</автор>
  <назва>Энциклопедия, 2-е изд.</назва>
  <видавництво>-СПб.:Питер</видавництво>
  <рік>2004</рік>
  <кіл_сторінок>1101</кіл_сторінок>
 </джерело>
<джерело наявність="ні">
  <автор>Сакс Т., Мак-Клейн Г.</автор>
   …
```

Такій файл буде відображено браузером, як зображено на рис. 9.

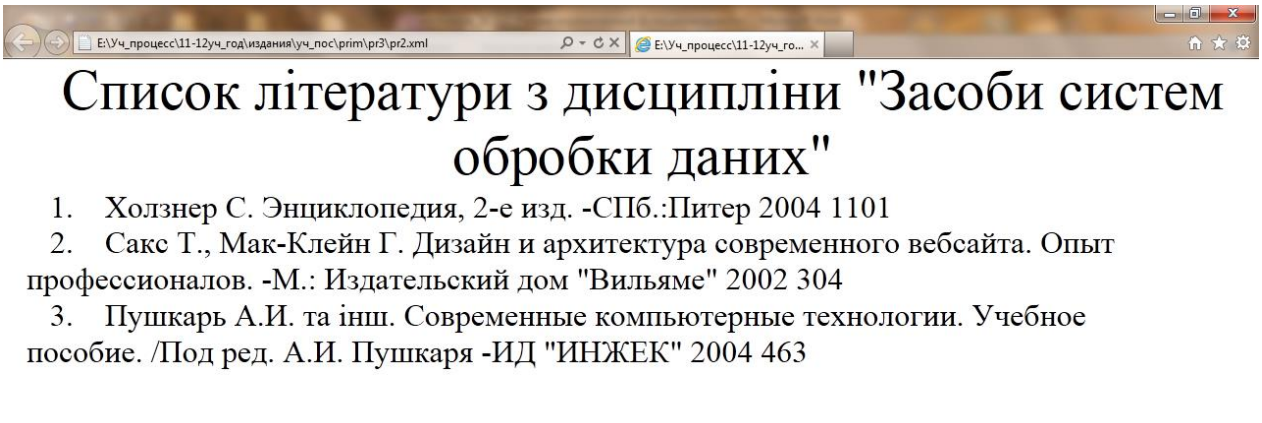


Рис. 9. **Відображення XML-файла з таблицею стилів у браузері**

Специфікація CSS містить велику кількість різних селекторів. Зокрема, імена тегів (p,h1), класові (.cl) і id-селектори (#id). Проте, під час роботи з XML-документами класові та id-селектори у такому вигляді не працюють.

Цей факт пояснюється дуже легко. У HTML імена тегів і атрибутів фіксовані, тому атрибути class і id мають фіксований сенс, з ним і пов'язана форма селектора. У XML цього немає, class і id нічим не виділяються серед інших атрибутів, тому така форма не підтримується.

Для зв'язку правил з елементами більш за все підходить селектор атрибутів. Він конструюється з імені елемента і імені атрибута у квадратних дужках:

p[class] – усі теги **p**, що мають атрибут class;

p[class="c1"] – усі теги **p**, що мають атрибут class із значенням c1;

*[id="id1"] – усі теги, що мають атрибут id із значенням id1.

Підкреслимо, що використовувати можна будь-які атрибути, а не лише class та id.

Відображаючи документи в різних браузерів та аналізуючи результат, можна створити висновок про кілька обмежень, пов'язаних із використанням CSS.

Таблиці стилів не дають можливості модифікувати або реорганізовувати вміст документа.

Вони не дозволяє здійснювати доступ до атрибутів, інструкцій з оброблення та іншим компонентам XML, а також не дають можливості обробляти

інформацію, яку ці компоненти містять. Тому, наприклад, неможливо відобразити рисунок.

Частіше за все таблиці стилів використовуються для відображення текстового змісту документа.

Універсальним підходом для відображення ХМЛ-документів можна вважати розроблення HTML-сторінок, що містять клієнтський скрипт. Скрипт виконує читання XML-файла, в пам'яті браузера будується DOM (Document Object Model), дуже схожий на DOM HTML-сторінки. Потім із деревом DOM працюють стандартними методами (переміщаються по вузлах, витягуються дані і т. п.). Для виконання цих дій у кожному браузері є відповідний об'єкт, у скрипті створюється його екземпляр і виконуються необхідні дії. З використанням отриманих даних динамічна формуються HTML-елементи для відображення у вікні браузера.

З використанням моделі DOM можна відображати дані XML-документа в довільному вигляді. Проте, передача XML-файла для розбору в різних браузерах проводиться по-різному. Тому доводиться розробляти скрипти з урахуванням типу використовуваної програми.

2.2.3. Перетворення XML-документів

У п.п. 2.2.1 та 2.2.2 були розглянуті основні питання XML-розмітки, перевірки коректності і валідності, відображення розмічених документів у вікні браузера. Водночас методи, які використовувались, були прив'язані до вже знайомих технологій (таблиці стилів CSS, програмування на боці клієнта JScript). Проте в межах розвитку технологій на основі XML були запропоновані рішення, які істотно розширюють можливості з оброблення і перетворення документів. Це розширювана мова таблиць стилів (Extensible Stylesheet Language, XSL).

У цей час пропрацьовано два підходи, що розширюють можливості з оброблення і відображення XML-документів: XSL-перетворення (XSL Transformations, XSLT) і форматуючи XSL-об'єкти (XSL Formatting Objects, XSL-FO). Обидва підходи базуються на XML (є його додатками) і забезпечують відображення даних XML-документа. Розглянемо XSLT.

XSLT – це XML-додаток, що визначає правила, відповідно до яких один XML-документ перетвориться в інший. XSLT-документ, тобто трансформуюча таблиця стилів XSLT, містить шаблони (опис форматування). XSLT-процесор порівнює елементи вхідного XML-документа із шаблонами

в таблиці стилів. Коли відповідний шаблон знайдено, процесор записує вміст шаблону у вихідне дерево. Після закінчення цього етапу вихідне дерево записується або в XML-документ, або в інший формат, такий як звичайний текст або HTML.

Таким чином, XSLT застосовується для оброблення документів (фільтрація даних, внесення додаткової розмітки і т. п.). Ця мова забезпечує доступ до вмісту XML-документів, а також позволяє створювати на їх основі нові документи. Одне дерево перетвориться в інше, одна розмітка – в іншу. Документ може бути перетворений в інший формат (HTML, спеціальна розмітка, звичайний текст).

Якщо таблиця XSLT створена, можна перетворити початковий документ за допомогою XSLT-процесора. XSLT-процесор – це програма, яка читає таблицю стилів XSLT, читає вхідний XML-документ і перетворить вхідний документ у вихідний відповідно до інструкцій, даних в таблиці стилів (рис. 10).

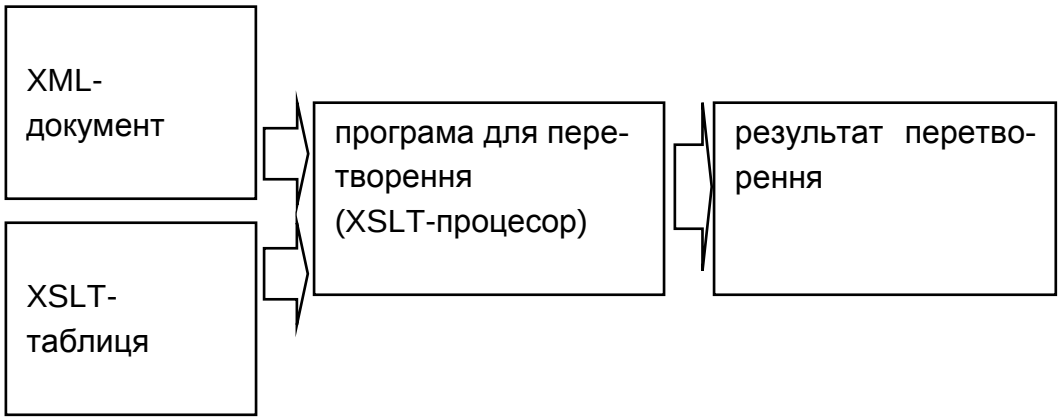


Рис. 10. **Використання XSLT**

XSLT-процесор може бути вбудованим у браузер, у сервер або сервер додатків, а може бути окремою програмою.

Достатньо зручно використовувати браузер і перетворювачі, вбудовані в XML-редактори.

Створення XSLT. Таблиця XSLT є коректне сформованим XML-документом, має XML-оголошення, може містити оголошення типу документа (DTD), хоча в більшості таблиць стилів його немає. Кореневим елементом цього документа є або stylesheet, або transform. Вони обидва мають однаковий набір можливих дочірніх елементів і атрибутів, і обидва означають для XSLT-процесора одне і те саме.

Частіше за все, XSLT-документ виглядає так:

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<!-- один або декілька елементів шаблону...-->
</xsl:stylesheet >
```

Елемент **xsl:stylesheet** ідентифікує документ як XSL-таблицю стилів і слугує контейнером для інших елементів. Усі імена, що мають відношення до таблиці, повинні мати префікс "xsl:".

Файли, що містять XSLT, мають розширення xsl. XML-документи, що підлягають обробленню з використанням таблиць стилів, повинні мати в пролозі інструкцію оброблення xml-stylesheet, яка указує, де знаходиться відповідна документу таблиця стилів. Якщо це таблиця стилів XSLT, то атрибут type повинен мати значення text/xml (на відміну від text/css для CSS). Наприклад,

```
<?xml-stylesheet type="text/xml" href="http://www.ore.com/styles/people.xsl"?>
```

Ця інструкція говорить про те, що слід застосовувати таблицю стилів, яка знаходиться за абсолютною URL-адресою. Значенням атрибута href може бути і відносна адреса.

Для формування вихідних даних на основі початкового XML-документа в таблицю стилів XSLT включають шаблони. Кожен шаблон представлений елементом xsl:template. Цей елемент має атрибут match, який вказує на певну гілку дерева структури документа (атрибут match аналогічний селектору в правилі CSS). Значення атрибута match має назву зразка (pattern). Наприклад, шаблон, який містить інструкції для відображення всього XML-документа:

```
<xsl:template match="/">
<!-- дочірні елементи... -->
</xsl:template >
```

Зразок у цьому шаблоні (символ "/") представляє весь XML-документ. Зразок match="/" може бути тільки один. Вставка цього шаблона у вихідний документ здійснюється автоматично (завжди), для вставки інших шаблонів їх активують за допомогою спеціальних елементів (<xsl:apply-templates>).

Інший простий зразок – це ім'я елемента, наприклад, <xsl:template match="person">Людина</xsl:template>. Цей шаблон говорить про те, що кожного разу, коли зустрічається елемент person, процесор таблиць стилів повинен генерувати текст "Людина".

Складніші конструкції для зв'язування шаблону з гілкою дерева записуються за допомогою спеціальної мови XPath.

Шаблон може містити послідовність елементів двох видів:

XML-елементи, що представляють незмінну коректну розмітку (наприклад, HTML-розмітку, а може бути яку-небудь іншу, в яку виконується перетворення);

XSL-елементи, які задають порядок використання даних із початкового документа (інструкції по вибору і модифікації даних XML).

XSL-таблиця може виконувати на основі шаблонів перетворення в розмітку (XML-документ) будь-якого вигляду, у тому числі і в HTML. Для XML-документів, що перетворюються в HTML-розмітку, XSL-таблиця стилів повідомляє браузеру (або іншій програмі, що виконує перетворення), як відобразити XML-документ шляхом його вибіркового перетворення в блоки HTML-розмітки.

Приклад вставки незмінної розмітки:

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
 <xsl:template match="/">
        <HTML >
        <HEAD >
        <TITLE>pr10</TITLE>
        </HEAD >
        <BODY >
        <H2>Результат перетворення</H2>
        </BODY >
        </HTML >
 </xsl:template >
</xsl:stylesheet >
```

Якщо відкрити в браузері XML-документ із посиланням на таку XSLT, то буде відображений HTML-документ, що міститься в шаблоні.

Для зміни порядку використання даних і оброблення існують спеціальні елементи (елементи трансформації). Програми оброблення відрізняють ці елементи за префіксом xsl. Основні елементи для перетворення (всього їх близько 30) наведені в табл. 2.

Таблиця 2

Елементи трансформації

Елемент	Призначення	Атрибути
<xsl:template>	опис шаблона	match
<xsl:apply-templates>	вставка даних за шаблоном	select
<xsl:value-of>	вставка значень з елемента	select
<xsl:for-each>	повторення обробки	select, order-by
<xsl:sort>	сортування	select, order
<xsl:attribute>	установка значень атрибутів	name

Вказівка на певний XML-елемент, з яким виконуватимуться перетворення, проводиться завданням значення атрибута select. Це може бути шлях по дереву або вираз XPath. У разі завдання шляху проводиться орієнтація на уявлення структури документа у вигляді дерева ("/" – кореневий елемент), з урахуванням поточного місцеположення, що задається атрибутом match в xsl:template або select в xsl:for-each. Можна використовувати спеціальні символи ("**.**" і " ***** **"**). Крапка – поточний елемент, зірочка – будь-який елемент.

Якщо дані знаходяться в атрибутах, то посилання на них виглядає таким чином: шлях/@ім'я_атрибута. Наприклад, для тега <STUDENT GR='1979'>Степаненко </STUDENT> посилання на дані у атрибутах могло б бути таким select='DOCUMENT/STUDENT/@GR'.

Розглянемо вживання конструкцій XSLT для перетворення XML-документа в HTML на прикладі.

Приклад 5. Створимо таблицю стилів XSLT для перетворення документу з прикладу 2 у формат HTML:

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
 <xsl:template match="/">
<HTML>
<HEAD>
<TITLE>pr5</TITLE>
</HEAD>
<BODY>
<H2>Список студентів, які не здали книги</H2>
<xsl:apply-templates select="DOCUMENT/STUDENT"/>
```

```
</BODY>
</HTML>
 </xsl:template>
 <xsl:template match="STUDENT">
       <IMG>
       <xsl:attribute name="src"><xsl:value-of select="photo"/></xsl:attribute>
       </IMG>
       <SPAN STYLE="font-style:italic">Прізвище: </SPAN>
       <SPAN><xsl:value-of select="NAME"/></SPAN>
       <BR/>
       <xsl:apply-templates select="BOOK"/>
 </xsl:template>
 <xsl:template match="BOOK">
       <SPAN STYLE="font-style:italic">Назва: </SPAN>
       <SPAN><xsl:value-of select="AUTOR"/></SPAN>
       <SPAN><xsl:value-of select="TITLE"/></SPAN><BR/>
 </xsl:template>
</xsl:stylesheet>
```

Вона містить три шаблони (кореневий елемент, STUDENT, BOOK). Шаблон кореневого елемента активізується автоматично, а шаблони вузлів STUDENT і BOOK інструкціями apply-templates у відповідних місцях. Для відображення фотографії було використано елемент <xsl:attribute name="src" >, що привласнює значення атрибуту тегу. Саме значення міститься в тексті елемента photo.

Слід ще раз зазначити, що наведена таблиця перетворень не є єдиною можливою. Технологія достатньо гнучка і допускає різні рішення. Наприклад, аналогічного результату можна досягти за допомогою такого коду:

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
<HTML>
<HEAD>
<TITLE>pr5</TITLE>
</HEAD>
<BODY>
<H2>Список студентів, які не здали книги</H2>
   <xsl:for-each select="DOCUMENT/STUDENT">
    <IMG>
       <xsl:attribute name="src"><xsl:value-of select="photo"/></xsl:attribute>
       </IMG>
       <SPAN STYLE="font-style:italic">Прізвище:
```

```
      </SPAN>
      <SPAN><xsl:value-of select="NAME"/>
       </SPAN>
      <BR/>
   <xsl:for-each select="BOOK">
    <SPAN STYLE="font-style:italic">Назва: </SPAN>
      <SPAN><xsl:value-of select="AUTHOR"/></SPAN>
      <SPAN><xsl:value-of select="TITLE"/></SPAN><BR/>
   </xsl:for-each>
  </xsl:for-each>
 </BODY>
</HTML>
  </xsl:template>
</xsl:stylesheet>
```

Тут перебір вузлів із метою витягання даних здійснюється з використанням елементів for-each (циклів) замість звернення до шаблона.

Використання обох таблиць дозволяє получити однаковий HTML-документ:

```
<?xml version="1.0" encoding="UTF-8"?>
<HTML>
  <HEAD>
    <TITLE>pr1</TITLE>
  </HEAD>
  <BODY>
    <H2>Список студентів, які не здали книги</H2>
    <IMG src="im1.gif"/>
    <SPAN STYLE="font-style:italic">Прізвище:
    </SPAN>
    <SPAN>Іванов</SPAN>
    <BR/>
    <SPAN STYLE="font-style:italic">Назва:
     </SPAN>
    <SPAN>Пушкар </SPAN>
    <SPAN> Основи наукових досліджень</SPAN>
    <BR/>
    <SPAN STYLE="font-style:italic">Назва: </SPAN>
    <SPAN>Молчанов  </SPAN>
    <SPAN> Технології Web-дизайну</SPAN>
    <BR/>
  </BODY>
</HTML>
```

Відображення файла в браузері наведено на рис. 11.

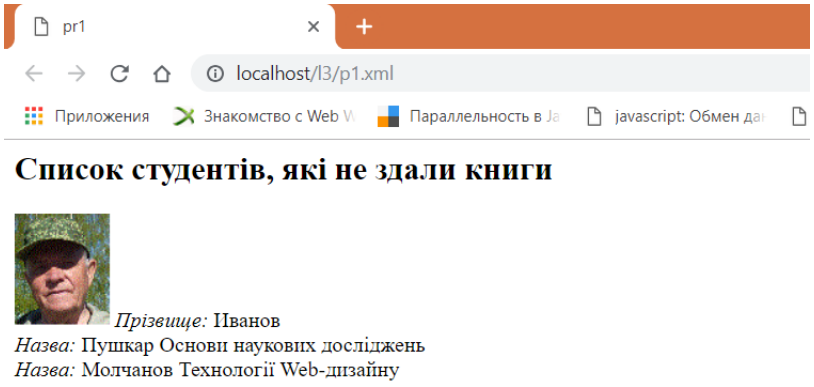


Рис. 11. **Відображення XML-файла після XSLT-перетворення в браузері**

У прикладі було розглянуто найпростіше перетворення. Окрім цього можна сортувати і фільтрувати дані для виведення, використовувати спеціальні функції і т. п.

XSL-перетворення для сортування. Для сортування використовується спеціальний елемент (інструкція) sort, що розміщується безпосередньо після for-each і apply-templates:

```
<xsl:sort select = "строкове_вираження "
 data-type = "text "або "number "
 lang = "код_мови "
 order = "ascending " або "descending "
 case-order = "upper-first " або "lower-first " />
```

Цей елемент змінює порядок проходження вузлів з порядку, прийнятого в документі, на інший, наприклад, алфавітний.

Атрибут select – це ключ сортування (ним ідентифікується вузол, за даними якого проводиться сортування). Якщо select відсутній, ключ сортування повинен дорівнювати значенню поточного вузла.

data-type – шаблон значення атрибута (як інтерпретувати дані, текст або числові значення). За замовчуванням вважається текст, і виконується сортування за абеткою.

Сортування залежить від мови. Встановивши атрибут lang відповідно до мовних кодів RFC 1766, можна змінити мову сортування.

order – визначає порядок, в якому сортуються рядки (descending – що убуває, ascending – що зростає). За замовчуванням сортування виконується в зростаючому порядку.

Атрибут case-order може бути встановлений або в upper-first, або в lower-first, указуючи, що або спочатку сортуються літери верхнього регістра, а потім нижнього, або навпаки. Значення за замовчуванням залежить від мови.

Наприклад, сортування за абеткою:

<xsl:sort order="ascending" select="AUTHOR/LASTNAME">

сортування за значенням

<xsl:sort order="ascending " select="number (PAGES)" data-type="number"/>

Можна виконувати сортування за декількома ключами (наприклад, спочатку сортувати за прізвищем, потім за іменем, потім по батькові), використовуючи декілька елементів xsl:sort у порядку убування важливості ключів.

XSL-перетворення для фільтрації. Значення, яке привласнюється атрибутам match або select (match використовується для елемента template, а select – для елементів value-of, for-each і apply-templates), є зразком, що відповідає одному або декільком елементам у XML-документі. Можна обмежити кількість елементів, що відповідають шаблону, ввівши фільтр. Фільтр – це вираз, поміщений у квадратні дужки ([]), який йде безпосередньо за послідовністю вузлів шляху. Наприклад, зразок, привласнений наступному атрибуту match, указує, що відповідний елемент повинен мати ім'я BOOK, і крім того, повинен мати дочірній елемент BINDING, який містить текст "trade paperback":

<xsl:template match="BOOK [BINDING='trade paperback ']">

Якщо у фільтр вміщено тільки ім'я елемента, то відповідний елемент повинен мати дочірній елемент з вказаним ім'ям. Наприклад:

match="ITEM[CD]" – будь-який елемент ITEM, що має дочірній елемент CD, незалежно від вмісту елемента CD;

match="SHIRT[COLOR='red']" – будь-який елемент SHIRT, що має дочірній елемент COLOR, що містить текст "red";

select="SHIRT[COLOR!='red']" – будь-який елемент SHIRT, що має дочірній елемент COLOR, який не містить текст "red".

Якщо елемент має більше ніж один дочірній елемент з ім'ям, вказаним в умові фільтрації, оператор порівняння застосовується тільки до першого дочірнього елемента. Наприклад, якщо елемент SHIRT має два дочірні елементи COLOR, зразок "SHIRT[COLOR='red']" відповідатиме елемента, тільки якщо перший елемент COLOR містить слово "red".

За необхідністю використання не строкових, а числових значень використовується такий синтаксис:

<xsl:for-each select="… [number(nam)$gt;10] " >,

де $gt – підстановка для символу ">" (у полі nam має бути значення більше, ніж 10).

Для виконання фільтрації можна також використовувати елемент <xsl:if test="number(nam)>10">…</xsl:if> Вкладені в цей тег елементи відображаються тільки тоді, якщо обчислена умова істинна.

Приклад 6. Створимо таблицю для фільтрації та сортування даних із прикладу 3 і перетворення їх у формат HTML:

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<HTML>
<HEAD>
<TITLE>Приклад</TITLE>
</HEAD>
<BODY>
<H2>Список літератури</H2>
<UL>
  <xsl:for-each select="список/джерело">
   <xsl:sort order="ascending" select="автор"/>
  <xsl:if test="number(рік)&gt;2002">
   <LI><SPAN STYLE="font-style:italic">
     <xsl:value-of select="автор"/>
     <xsl:value-of select="назва"/>
     <xsl:value-of select="видавництво"/>
     <xsl:value-of select="рік"/>
     </SPAN>
     <xsl:value-of select="кіл_сторінок"/></LI>
 </xsl:if>
   </xsl:for-each>
</UL>
```

```
</BODY>
</HTML>
</xsl:template>
</xsl:stylesheet>
```

Сортування виконано за абеткою за тегом *автор*, елемент <*xsl:sort order*… Фільтрація – за тегом *год*, елемент <*xsl:if test*… У результаті будуть відображатись тільки два джерела, які були видані після 2002 року (рис. 12).

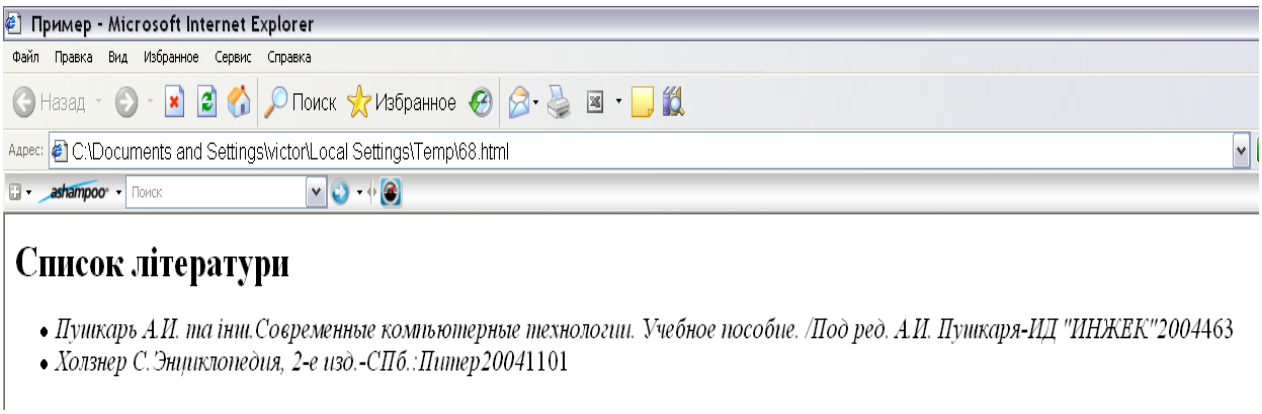


Рис. 12. **Відображення документа в браузері**

Переважна більшість браузерів мають підтримку XML і XSLT. Однак під час перетворення в браузері результуючий документ відображається у вікні, і отримати його в окремому файлі без програмування неможливе. Тому для таких дій зручніше користуватися редактором із вбудованим перетворювачем.

Тема 3. Мова програмування PHP

Існує декілька технологій створення і використання програм на боці сервера, що відрізняються мовами, способами взаємодії з сервером та іншими ознаками. До теперішнього часу склалися два підходи до створення програм на стороні сервера.

Перший – це розроблення програми, яка взаємодіє з сервером і вихідні дані якої (HTML-документ) прямує через сервер клієнтові. Приклад такого підходу було розглянуто в п. 1.1.

Другий – вбудовування програмного коду (скрипта) безпосередньо у сторінку. Серед цих технологій найбільшого поширення набули активні сторінки на мові PHP. Саме вона і буде розглянута.

 Сутність технології активних сторінок полягає у вбудовуванні програмних кодів у текст сторінки. Цей код призначено для виконання на сервері перед відправкою клієнтові (не плутати зі скриптами, що виконуються на клієнтському боці). Сервер виконує запрограмовані дії (звичайно, це вставка в сторінку тегів із даними, наприклад, бази і т. п.). Готова сторінка (без програми, яка вже виконана) відправляється клієнтові.

Серед реалізацій технології активних серверних сторінок найбільш відомі: PHP та ASP.NET.

PHP заснована на мові програмування PHP, що забезпечує генерування WEB-сторінок на сервері та роботу з базами даних. Зараз підтримується переважною більшістю хостінг-провайдерів. Мова заснована на C і Perl. Інтерпретатор PHP підключається до WEB-сервера або через модуль, створений спеціально для цього сервера (наприклад, для Apache або IIS), або як CGI-додаток. За статистикою більше 20 % доменів мережі працюють із PHP.

Програмний код поміщається всередину обмежувачів <?php.. ?> (допускається використання додаткових варіантів, таких як <?... ?> і <script language="php">.. </script>).

У HTML-документі це виглядає так:

```
<html>
<head>
<title>Тестуємо PHP</title>
</head>
<body>
<?php echo 'Hello, world!'; ?>
</body>
</html>
```

Бібліотеки містять велике число функцій, що полегшують розроблення. Найбільш масовими прикладами використання цієї технології є так звані системи управління контентом (SMC) Joomla, WordPres, Drupal та ін.

ASP.NET заснована на платформі .NET і, отже, використовує все нові можливості, що надаються цією платформою.

Завдяки Common Language Runtime (CLR), який є основою всіх додатків Microsoft .NET, розробники можуть писати код для ASP.NET, використовуючи практично будь-які мови програмування, зокрема, і що входять у комплект .NET Framework(C#, Visual Basic.NET, і JScript .NET).

ASP.NET має перевагу в швидкості порівняно зі скриптовими технологіями, оскільки під час першого звернення код компілюється і поміщається у спеціальний кеш, і згодом тільки виконується, не вимагаючи додаткових витрат часу.

Сторінки ASP.NET мають розширення .aspx, і відрізняються від HTML тим, що можуть містити програмні елементи, які виконуються на сервері (як ASP) і керуючі елементи, які можуть змінюватися на сервері перед відправкою. З керуючими елементами можуть зв'язуватися обробники подій, що виконуються на сервері. Усе це дає в руки розробника потужний інструмент для вирішення широкого кола завдань.

Програмний код може розміщуватися або в окремому файлі, або усередині спеціального тегу для сценаріїв. Файл із кодом звично має розширення *.aspx.cs (*.aspx.vb) і має ім'я, співпадаюче з ім'ям основного файла aspx.

Сторінки ASP.NET зазвичай містять директиви, які дозволяють вказати властивості і конфігурацію для сторінки. Директиви використовуються ASP.NET як інструкції, що визначають спосіб оброблення сторінки. Вони не відображаються в розмітці, яка відправляється клієнтові.

Головний недолік цієї технології – підтримка тільки серверами Microsoft.

3.1. Розроблення програм мовою PHP

Активна сторінка PHP є HTML-сторінкою, в текст якої вставлені інструкції мови PHP. Інструкції знаходяться всередині обмежувачів <?php... ?>. Файли сторінок PHP, зазвичай, мають розширення php, phtm або phtml. Під час оброблення сторінки, що містить код PHP, лексичного аналізу й інтерпретації піддається тільки вміст конструкції <?php ... ?>. Решта всього текстового змісту (код розмітки) прямує у вихідний потік без зміни.

Сценаріїв PHP на сторінці може бути декілька, вони можуть розташовуватися в декількох місцях, водночас простір імен залишається єдиним.

У сценаріях допустимі коментарі. Існують два формати коментарів: однорядкові (починається // або #, і закінчується кінцем рядка) і багаторядкові (ув'язнені між символами /* і */).

Приклад 7. Розробити активну сторінку, яка містить серверний і клієнтський сценарії.

```
<html>
<head>
<title>PHP сумісно з HTML</title>
</head>
<body>
  <?php
  $h="h3";            //  змінній привласнене значення
```

```
  echo "<",$h,">";  # використано значення змінної $h
  ?>
Сторінка сформована на сервері
  <?php
  /* У цьому фрагменті сформовано значення поточного часу на сервері*/
   echo date("H:i:j"),"</",$h,">";
  ?>
<script>
sDate=new Date();
s="<p>сторінка завантажена "+sDate.toLocaleString()+"</p>";
document.write(s);
</script>
</body>
</html>
```

Сторінка містить статичний зміст (HTML та клієнтський скрипт), два фрагменти скрипта PHP і коментарі. Інструкції PHP виконуються на сервері і клієнтові не відправляються.

Код, який отримав браузер після обробки на сервері, наведено на рис. 13.

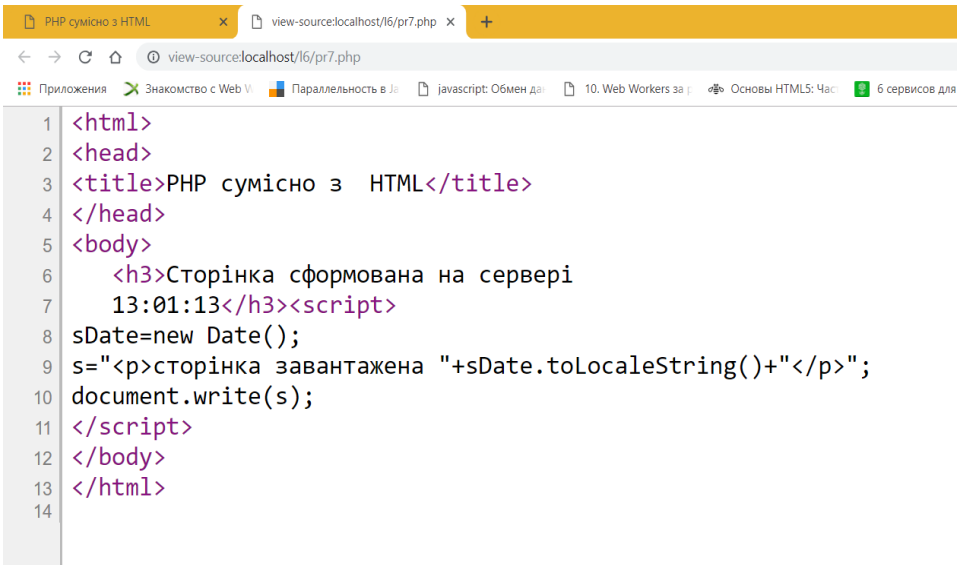


Рис. 13. **Код, який отримав браузер**

Розглянемо основні синтаксичні конструкції мови PHP.

Змінні. У PHP імена змінних починаються зі знаку $, за яким йде саме ім'я. Імена є послідовністю літер, цифр і символів підкреслення, яка починається з букви або підкреслення. Імена змінних в PHP чутливі до регістра, змінні $**user,** $**User** і $**USER** є різними.

Окремого оголошення змінних не вимагається, змінні створюються під час привласнення значень. Наприклад, оператором привласнення.

```
<?php
$var = "5"; $var1 = 5;
echo $var,$var1;
?>
```

Під час завдання змінних не потрібно явно указувати тип, водночас одна і та сама змінна може послідовно набувати значень різних типів. Найчастіше використовуються наступні типи: integer (цілий), string (строковий), boolean (логічний), double або float (речовий), array (масив), object (об'єкт). Під час привласнення тип змінної визначається за контекстом. Для явного перетворення типів використовується ім'я типу в дужках (як в C).

```
$var = "5";     // тип string
$var = (int)$var; // перетворене в integer
```

За необхідністю визначення типу змінної в ході виконання програми, можна використовувати функцію **gettype**(), вона повертає значення типу змінною:

```
<?php
$var = "5";
$var1 = 5;
echo gettype($var);
echo "<br>";
echo gettype($var1);
?>
```

Окрім змінних у PHP-скриптах використовуються константи – це імена, з якими пов'язані незмінні значення. Вони створюються за допомогою функції **define**(): define(CONSTANT, value). Перший параметр цієї функції – ім'я константи, другий – її значення. Для використання константи на неї посилаються за ім'ям:

```
<?php
define(CONSTANT1,15);
define(CONSTANT2,"\x20"); // код пропуску
define(CONSTANT3,"Hello");
echo(CONSTANT1);
echo(CONSTANT2);
```

```
echo(CONSTANT3);
?>
```

Для перевірки чи визначена константа, слугує функція defined(), true – визначена, false – ні.

```
<?php
define(CONSTANT,"Hello");
if(defined("CONSTANT"))
{
echo("<p>CONSTANT is defined</p>");
}
?>
```

Для забезпечення зручності в програмуванні оброблення запитів від клієнта в PHP встановлюють значення спеціальних змінних, які містять дані, що належать до запиту і доступні весь час його виконання. Ці змінні називаються зовнішніми, вони доступні в будь-якому місці скрипта.

$_SERVER – асоціативний масив (доступ за іменем змінної), містить значення змінних оточення операційної системи, що передаються програмі. У цьому масиві **$_SERVER[QUERY_STRING]** – рядок запиту (інформація, наступна за символом "**?**" у запитаному URL).

$_GET – асоціативний масив значень, отриманих із форми за методом передачі GET.

$_POST – асоціативний масив значень, отриманих із форми за методом передачі POST.

Оператори. Склад операцій для використання у виразах практично не відрізняється від мови JavaScript. Відзначимо лише деякі особливості.

Конкатенація рядків (об'єднання) позначається крапкою:

```
$st="string1".'string2'.
```

Операція виведення даних у вихідний потік echo, значення, що виводяться, можуть бути конкатенацією рядків, розділятися комою або пропуском.

Набір операторів управління (цикли, розгалуження) такий самий, як в JavaScript.

Як приклад розглянемо динамічне формування посилань для панелі навігації на сторінці. Динамічне в тому сенсі, що значення атрибуту

href формуватиметься на сервері у момент обробки сторінки, і може змінюватися від завантаження до завантаження.

```
…
<h3>одне посилання:</h3>
<?php
//$pr – логічна змінна, яка керує переходом
if ($pr) $link = "date1.php";
else   $link = "date2.php";
echo "<a href=$link>проглянуть цю сторінку</a><br>"
?>
<h3>Список посилань</h3>
<?php
…
```

PHP-код забезпечує формування посилання за адресою, яка встановлюється залежно від значення логічної змінної $pr.

Крім розглянутих, є кілька специфічних операторів для роботи з масивами й об'єктами.

Масиви. У PHP підтримується використання традиційних масивів – сукупності впорядкованих однотипних елементів. Створюються вони за допомогою конструктора array(). Доступ до елементів здійснюється за цілочисельним індексом. Наприклад:

```
$products=array("олія","хліб","чай");
echo $products[0]," ", $products[1]," ", $products[2];
```

Масив можна динамічне змінювати, додавати та видаляти елементи.

Крім традиційних масивів (із цілими індексами) можливе створення асоціативних масивів, в яких доступ забезпечується за ключем (рядком символів). Створення і доступ до елементів таких масивів виконують таким чином:

```
$prices=array("олія"=>21.50,"хліб"=>4.25,"чай"=>12.30);
echo $prices["олія"]," ", $prices["хліб"]," ", $prices["чай"],"<br>";
echo $prices[$products[0]]," ", $prices[$products[1]]," ", $prices[$products[2]].
```

Для роботи з масивами (як традиційними, так і асоціативними) передбачений спеціальний оператор циклу foreach:

```
foreach ($course as $key => $value) {  …};
```

У кожному циклі буде обрано один елемент, та привласнені значення:

$course – ім'я масиву, який обробляється;

$key – змінна, якої буде привласнено значення ключа;

$value – змінна, якої буде привласнено значення елемента.

Для забезпечення зручності роботи з асоціативними масивами використовуються також спеціальні функції. Частіше всього each(), яка повертає ключ і значення поточного елемента масиву і робить поточним наступний елемент. Завдяки цьому можна послідовно перебирати елементи асоціативного масиву:

```
while ($el=each($prices)){
echo $el["key"];
echo "-";
echo $el["value"];
echo "<br>";}
```

Крім того, для роботи з масивами є велике число функцій, за допомогою яких можна виконувати сортування, заповнення даними з файла, з рядка тощо. Наприклад, наступний скрипт формує масив із рядка за допомогою функції explode(), і використовує його значення для створення на сторінці текстового поля із списком, що розкривається.

```
<?php
$strarray=explode(":","Понеділок:Вівторок:Середа:Четвер:П'ятниця:Субота:Ніділя");
?>
<select name="day" size="1">
<?php for($i=0;$i<7;$i++){ ?>
<option><?php echo($strarray[$i]); ?></option>
<?php } ?>
</select>
```

Рядки. Особливістю рядків у мові PHP можна вважати два способи їх завдання: в одинарних лапках і подвійних.

Під час завдання рядка в подвійних лапках імена змінних всередині послідовності символів замінюються їхніми значеннями. Для відображення спеціальних символів (наприклад, лапок) необхідно перед ними записувати символ \:

```
$Count = "рядок";
$My_string = "рядок зі \"змінними\"$count";
```

У результаті змінна $my_string буде містити такий рядок: *рядок зі "змінними" рядок.*

Під час задавання рядків із використанням одинарних лапок імена змінних залишаються просто послідовністю символів, без оброблення. Наприклад, у результаті виконання такого фрагмента:

```
$count = 13;
$string = 'В рядку \'Hello, world!\' $count символів';
```

змінна $string буде містити: *У рядку 'Hello, world!' $count символів.*

Ще одної особливістю мові є відсутність окремого символьного типу. У мові PHP символ – це рядок із довжиною 1. Вибірка окремих символів із рядка може здійснюватися шляхом вказівки порядкового номера символу, який починається з нуля, і який повинен бути зазначений у фігурних дужках безпосередньо за ім'ям рядкової змінної. Наприклад, фрагмент програми, наведений нижче, робить вибір із рядка і виведення у вихідний потік по одному символу тексту.

```
$my_string = "Doubled";
for ($index = 0; $index <strlen($my_string); $index++)
{
   $char = $my_string{$index};
        print("$char");
}
```

У мові PHP передбачений ще один спосіб задавання рядка – так звана синтаксична структура вкладеного документа (*Heredoc*). Подібна синтаксична конструкція є зручним засобом задавання великих фрагментів тексту.

Знаком операції, що застосовується у синтаксичній структурі вкладеного документа, є <<<. За цим знаком повинна безпосередньо йти мітка (без лапок), яка позначає початок багаторядкового тексту. Мітка може мати будь-яке ім'я, відповідне звичайними правилами іменування змінних у мові PHP. Інтерпретатор PHP продовжує включати до складу значення змінної такі рядки до тих пір, поки знову не з'явиться та сама мітка, що й на початку рядка. За заключною міткою може слідувати необов'язкова крапка з комою.

Наприклад,

```
$string = <<<EOT
<form method="post" action="{$_SERVER['PHP_SELF']}">
<input type="text" name="login" placeholder="Введіть ім'я..."><br>
   <input type="password" name="password" placeholder="Введіть
пароль..."><br>
   <input type="submit" value="Відправить">
</form>
EOT;
echo $string;
```

Цей фрагмент програми (виводить найпростішу HTML-форму), забезпечує присвоєння змінної *$string* рядка, що містить HTML-код. Зручною особливістю є те, що у цьому тексті можуть бути присутніми знаки лапок (й інші спеціальні символи).

Для дій із рядками передбачені дві рядкові операції: операція конкатенації (зчеплення рядків, позначається крапкою) й операція конкатенації з привласненням (позначається . =). Наприклад,

```
$string_1 = "Це частина";
$string_2 = "рядки";
// Конкатенація рядків
echo $string_1." простий ".$string_2."<br>";     // "Це частина простого рядка"
// Конкатенація з привласненням
$string_1 .= " простий ";    // Еквівалентно $string_1 = $string_1." простий ";
$string_1 .= $string_2;
echo $string_1;                  // " Це частина простого рядка "
```

Крім цього у мові PHP передбачена велика кількість різноманітних функцій для оброблення і перетворення рядків:

strlen() – повертає довжину рядка у байтах;

mb_strlen() – повертає довжину у символах;

strpos() – повертає номер позиції конкретного символу або початку підрядка у рядку у байтах;

mb_strpos() – повертає номер позиції конкретного символу або початку підрядка у рядку у символах.

Структура коду. Під час запиту активних сторінок на сервері відбувається роздільне оброблення коду HTML і PHP (HTML-код прямує до клієнта без зміни, а PHP виконується). У зв'язку із цим є можливість

комбінувати розміщення коду як в одному, так і в декількох файлах, створюючи ефективні та компактні додатки. Для об'єднання фрагментів коду, розміщених у різних файлах, існує кілька функцій: include(); include_once(); require(); require_once().

Під час використання цих функцій файл, що включається, може мати довільне розширення. Крім того, якщо його вміст є PHP-кодом, то воно має бути поміщено всередину дескриптора <?php…?>.

Функція include(file) вставляє вміст файла з вказаним ім'ям у скрипт під час звернення до неї.

Функція include_once(file) виконує те саме, що і include(), але заздалегідь перевіряє, чи не був він включений раніше. Якщо файл вже був включений, виклик функції ігнорується.

Функція require(file) вставляє вміст у місці виклику, але вставка проводиться до виконання коду (отже, якщо помістити виклик в умовному операторі, код буде вставлений незалежно від результату перевірки умови). Require_once(file) – вставка проводиться тільки один раз.

З використанням цих функцій можна розділяти код розмітки і програмний код, забезпечуючи тим самим більш зручну взаємодію верстальника і програміста.

Приклад 8. Розробити активну сторінку з розділенням коду розмітки та коду скрипта.

Файл із кодом розмітки (heder.htm):

```
<!-- Шаблон розмітки -->
<!DOCTYPE html>
<html>
<head>
    <title>template: header</title>
    <meta http-equiv="content-type" content="charset=utf-8" />
    <link rel="stylesheet" type="text/css" href="pr4_css.css" />
</head>
<body>
<header class="stroke_1">Заголовок сайта</header>
<div class="main">
<div class="lc"><h2>Список замовлень</h2>
                <?php echo $content_left ?>
</div>
<div class="rc"><h2>Про нашу компанію</h2>
                 <?php echo $content_right ?>
```

```
</div>
</div>
<footer>
  <p>Copyright &copy; 2018 Your Name - All rights reserved </p>
</footer>
</body>
</html>
```

Усі дії з верстання цієї сторінки можна виконувати автономно. Вигляд зверстаного шаблона (heder.htm) наведено на рис. 14.

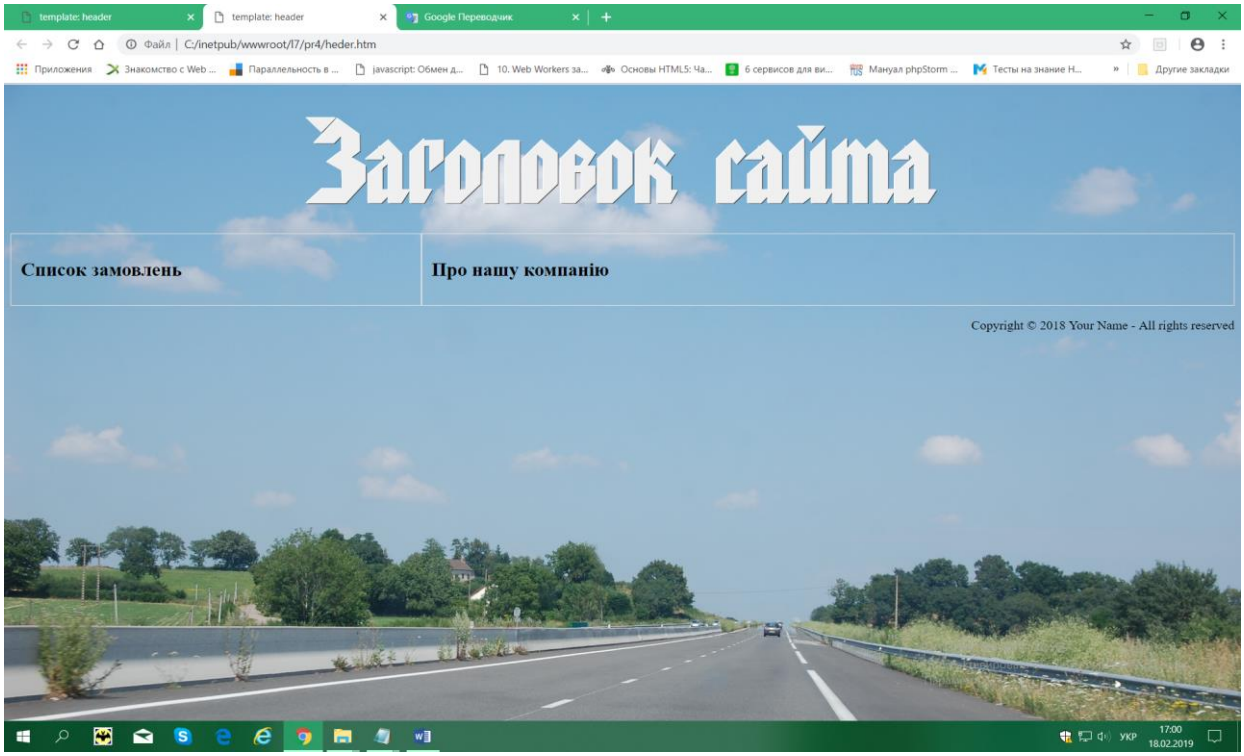


Рис. 14. **Вигляд шаблону сторінки у браузері**

Файл index.php містить код сценарію:

```
<?php
      //масив списку послуг:
$ordersList = array(
array('description' => 'Закупівля будівельних матеріалів','price' => 250000,'id'
=> 230565),
array('description' => 'Внутрішнє оздоблення будівлі','price' => 120000,'id' =>
230789),
array('description' => 'Складання проекту ландшафту','price' => 180000,'id' =>
247752));
     //формування рядка для лівої колонки:
```

```
foreach ($ordersList as $order) {
$o=$order['id']; $d=$order['description'];$p=$order['price'];
$content_left =$content_left."<div> <h3 class='c1'>Номер замовлення:  $o </h3>
                                        <p>Опис: $d </p>
                                        <p>Вартість: $p грн</p>
                                </div>";};
    //формування рядка для правої колонки:
$content_right=<<<c_l
Термін перебування на будівельному ринку
…
на сьогодні складає більше шістнадцяти років.
c_l;
     //завантаження шаблону розмітки:
include 'heder.htm';
?>
```

Дані для відображення в лівій колонці зберігаються у вигляді масиву ($ordersList), хоча можна використовувати будь-яке інше джерело (базу даних, файл на сервері). Із цих даних формується рядок ($content_left) для вставки в код розмітки.

Дані для правої колонки зберігаються у вигляді рядка Heredoc ($content_right).

Зверстаний шаблон завантажується директивою include 'heder.htm'.

На рис. 15 зображено вигляд сторінки, отриманої браузером із сервера у відповідь на запит (http://localhost/index.php).

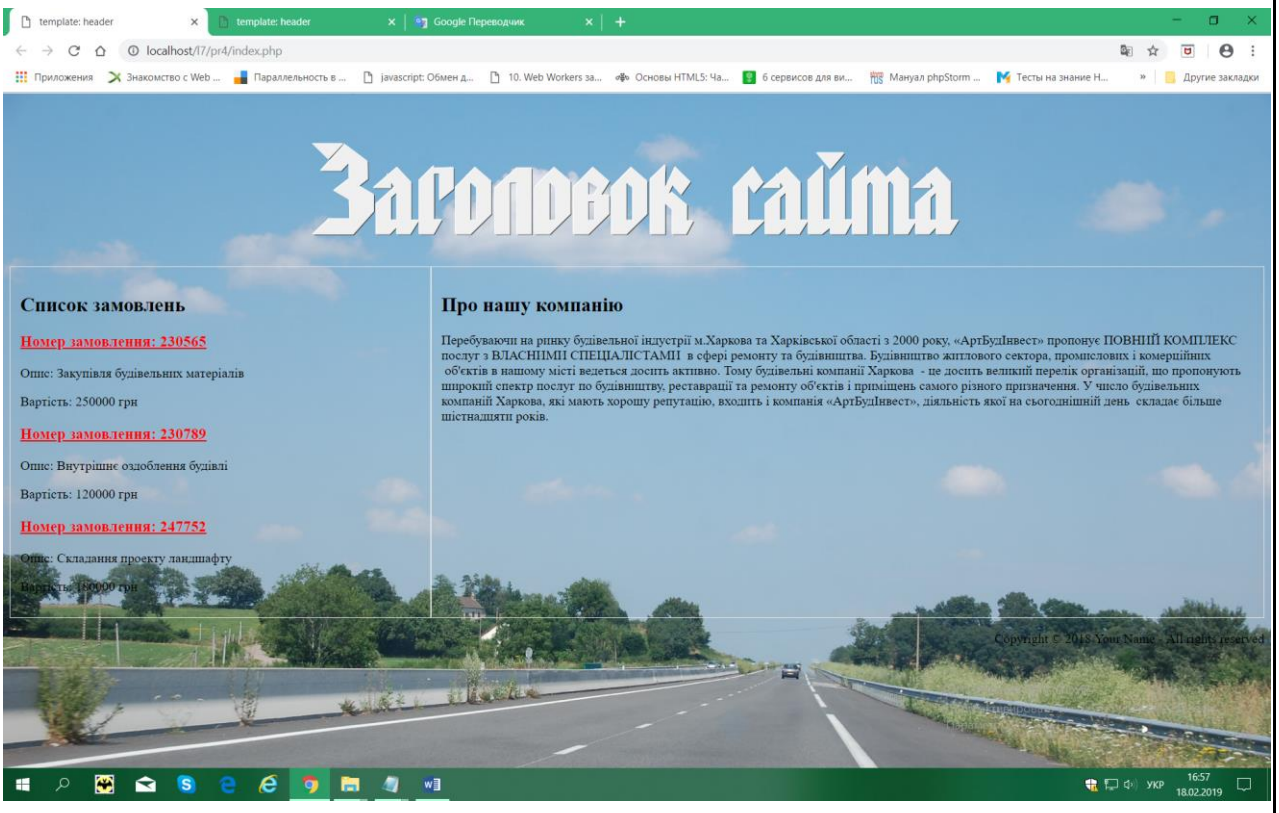


Рис. 15. **Сторінка у браузері**

3.2. Використання функцій

Функції в PHP, як і в інших мовах є фрагментами коду, якому привласнені імена. Імена функцій будуються за загальними правилами. Вони не чутливі до регістра (Func() і fUNC() – одна і та ж функція). Імена функцій доступні в будь-якому місці скрипта, незалежно від місця оголошення, тобто мають глобальну область бачення. Функція може повертати значення будь-якого типу (в тому числі і масив). Для привласнення імені функції значення використовується оператор return.

Перед використанням (виконанням) функція повинна бути оголошена. В оголошенні вказуються ім'я функції, аргументи та оператори, які повинні виконуватися (тіло функції). Оголошення функції виглядає таким чином: function Ім'я (аргументи){тіло_функції}.

Значення аргументів (параметрів) можуть передаватися у функцію трьома способами: за значенням, за посиланням і за замовчуванням. Імена параметрів, що передаються за значенням, записуються звичайним чином, імена, що передаються за посиланням, починаються з "&", імена параметрів за замовчуванням повинні знаходиться в кінці списку.

Наприклад:

```
function form_str($str_1, &$str_2, $str_3="значення за замовчуванням")
{
...
}
```

У даному прикладі функція має три параметри. Перший передається за значенням (зміна параметра усередині функції не приводить до зміни змінної, що підставляється під час звернення). Другий параметр передається за посиланням (символ & перед ім'ям параметра в оголошенні), зміна цього параметра усередині функції змінює значення змінної, що підставляється під час звернення. Третій параметр містить значення за замовчуванням (після знаку = у оголошенні), якщо під час звернення він не використаний, то використовується це значення.

У PHP функції мають властивість локалізації імен: всі змінні, використовувані усередині функції (включаючи параметри) є локальними і невідомі поза функцією незалежно від того, чи збігаються їхні імена із змінними поза функцією. За необхідністю розширити простір дії імен можна використовуючи ключове слово global.

Наприклад:

```
function func_1($str_1)
{
global $str_2=5;
…
}
func_1(4);
echo $str_1; // буде невизначене
echo $str_2; // буде виведено 5
```

Імена функцій можуть присвоюватися змінним:

```
function foo() {    echo "викликана функція foo"; }
$func = 'foo';
$func();       // буде викликана функція foo()
```

Також можна передавати їх в якості параметрів:

```
function f1() { echo "f1 function executed"; }
function f2($func) { $func(); }
$str = "f1";
f2($str); // буде виведено "f1 function executed"
```

Бібліотека PHP містить велику кількість різноманітних функцій, що полегшують написання додатків.

Приклад 9. Розробити активну сторінку з використанням функцій, які розроблені користувачем.

Шаблон розмітки розміщено у двох файлах (heder.htm та footer.htm).

heder.htm:

```
<!--статична частина сторінки-->
<!DOCTYPE html>
<html>
<head>
      <title>Приклад 9</title>
      <meta http-equiv="content-type" content="charset=utf-8" />
      <link rel="stylesheet" type="text/css" href="pr9_css.css" />
</head>
<body>
<h1>Таблиця множення</h1>
```

footer.htm:

```
<!--статична частина сторінки-->
<footer>
  <p>Copyright &copy; 2018 Your Name - All rights reserved </p>
</footer>
</body>
</html>
```

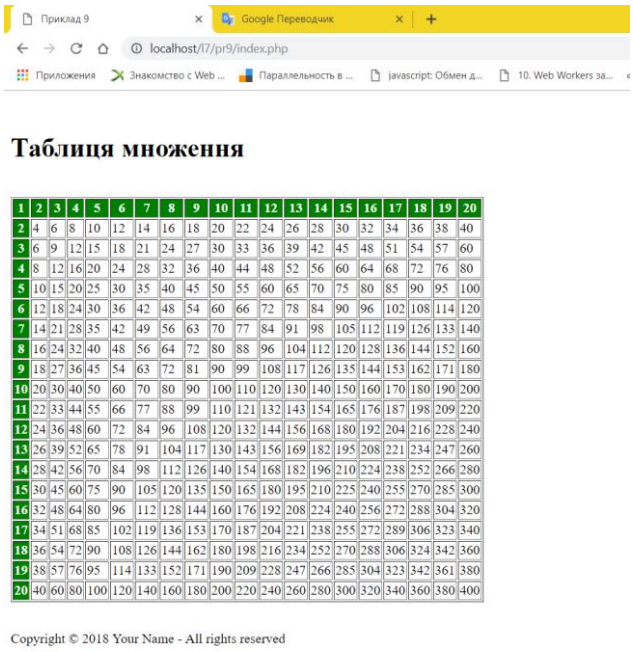
Серверний скрипт також поділено на два файли: bbl.con та index.php. Файл bbl.con містить код функції, яка створює код розмітки для таблиці множення:

```
<!--динамічна частина сторінки-->
<?php
function getTable($rows, $cols){
    $table = '<table border="1">';
   for ($tr=1; $tr<=$rows; $tr++){
      $table .= '<tr>';
      for ($td=1; $td<=$cols; $td++){
        if ($tr===1 or $td===1){
            $table .= '<th style="color:white;background-color:green;">'. $tr*$td
.'</th>';
         }else{
            $table .= '<td>'. $tr*$td .'</td>';
         }
      }
      $table .= '</tr>';
   }
   $table .= '</table>';
   echo $table;
}
?>
```

Файл index.php містить код, який забезпечує формування сторінки:

```
<?php
include ("head.con");
require ("bbl.con");
getTable(20,20);     // таблиця множення 20x20
include ("foot.con");
?>
```

На рис. 16 зображено вигляд сторінки, отриманої браузером із сервера у відповідь на запит (http://localhost/index.php).



1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	34	36	38	40
3	6	9	12	15	18	21	24	27	30	33	36	39	42	45	48	51	54	57	60
4	8	12	16	20	24	28	32	36	40	44	48	52	56	60	64	68	72	76	80
5	10	15	20	25	30	35	40	45	50	55	60	65	70	75	80	85	90	95	100
6	12	18	24	30	36	42	48	54	60	66	72	78	84	90	96	102	108	114	120
7	14	21	28	35	42	49	56	63	70	77	84	91	98	105	112	119	126	133	140
8	16	24	32	40	48	56	64	72	80	88	96	104	112	120	128	136	144	152	160
9	18	27	36	45	54	63	72	81	90	99	108	117	126	135	144	153	162	171	180
10	20	30	40	50	60	70	80	90	100	110	120	130	140	150	160	170	180	190	200
11	22	33	44	55	66	77	88	99	110	121	132	143	154	165	176	187	198	209	220
12	24	36	48	60	72	84	96	108	120	132	144	156	168	180	192	204	216	228	240
13	26	39	52	65	78	91	104	117	130	143	156	169	182	195	208	221	234	247	260
14	28	42	56	70	84	98	112	126	140	154	168	182	196	210	224	238	252	266	280
15	30	45	60	75	90	105	120	135	150	165	180	195	210	225	240	255	270	285	300
16	32	48	64	80	96	112	128	144	160	176	192	208	224	240	256	272	288	304	320
17	34	51	68	85	102	119	136	153	170	187	204	221	238	255	272	289	306	323	340
18	36	54	72	90	108	126	144	162	180	198	216	234	252	270	288	306	324	342	360
19	38	57	76	95	114	133	152	171	190	209	228	247	266	285	304	323	342	361	380
20	40	60	80	100	120	140	160	180	200	220	240	260	280	300	320	340	360	380	400

Рис. 16. **Сторінка в браузері**

3.3. Оброблення даних

Найчастіше серверні скрипти розробляються для отримання і оброблення даних із форм, пошуку і обробки даних для розміщення на сторінках, у відповідь на дії користувача.

3.3.1. Оброблення даних із форм

Основне призначення форм полягає в передачі серверу інформації від користувача. Наприклад, для реєстрації або збору особистих даних, даних різних опитувань тощо. У них можна вводити текст або вибирати відповідні варіанти із списку, а також встановлювати значення різних перемикачів. Дані, записані у форму, відправляються для оброблення на сервері. Залежно від введених значень сервер може формувати різні WEB-сторінки, відправляти запити до бази даних або виконувати пошук у файлах тощо.

Відправка даних форми відбувається після натиснення кнопки типу submit. Як приклад розглянемо форму для збору даних про інтереси відвідувача.

```
...
<h2> Заповніть форму відповідно до своїх уподобань </h2>
<form action="test.php" method=POST>
Ім'я <br><input type=text name="first_name" value=" Введіть ваше ім'я "><br>
E-mail <br><input type=text name="email"><br>
<p>Ваш рівень<br>
<input type=radio name="lev" value="1"> Топ-менеджер <br>
<input type=radio name="lev" value="2"> Керівник проекту <br>
<input type=radio name="lev" value="3"> WEB-майстер <br>
<P>Ваш вік <BR>
<input type=radio name="ag" value="1"> 18-24 <br>
<input type=radio name="ag" value="2"> 25-40 <br>
<input type=radio name="ag" value="3"> 41-55 <br>
<P> Що ще Ви хочете повідомити про себе
<textarea name="comment" cols=32 rows=5></textarea>
<P><input name="confirm" type=checkbox checked>Підтвердити отримання<br>
<input type=submit value="Надіслати">
<input type=reset value="Скасувати">
</form>
...
```

Під час відправки даних форми за допомогою методу **GET** вміст елементів форми додається до URL після знаку питання у вигляді пар ім'я=значення, об'єднаних за допомогою символу &:

action?name=...&email=...&lev=1&ag=1&comment=...&confirm=on

Action – це URL-адреса програми, яка повинна обробляти форму. Імена name, e-mail, lev, ag, comment, confirm відповідають іменам елементів форми, а символи, що стоять після знаку рівняння, – значення, встановлені для цих елементів.

Взагалі кажучи, можна створити рядок запиту в адресному рядку браузера і без форми, набравши вручну наведений вище рядок.

Метод GET зручний, якщо параметрів небагато, а також під час відлагодження скриптів (тоді можна бачити значення та імена змінних, які передані).

За методом POST вміст форми кодується точно так, як і для методу GET, але сам запит посилається блоком даних як частина операції POST.

Цей метод рекомендовано для передачі великих за об'ємом блоків даних. Інформація, яка введена користувачем і відправлена серверу за допомогою методу POST, подається на стандартне введення програми, яка вказана в атрибуті action. Довжина блоку передається в змінній оточення CONTENT_LENGTH, а тип даних – у змінній CONTENT_TYPE.

 Під час відправки даних на сервер будь-яким методом передаються не тільки самі дані, введені користувачем, але і ряд змінних, що називаються змінними оточення, які характеризують клієнта, історію його роботи, шляхи до файлів тощо. Ось деякі зі змінних оточення:

REMOTE_ADDR – IP-адрес хоста (комп'ютера), що відправляє запит;
REMOTE_HOST – ім'я хоста, з якого відправлений запит;
HTTP_REFERER – адреса сторінки, що посилається на поточний скрипт;
REQUEST_METHOD – метод, який був використаний для відправки запиту;
QUERY_STRING – інформація, що знаходиться в URL після знаку питання;
SCRIPT_NAME – віртуальний шлях до програми, яка повинна виконуватися;
HTTP_USER_AGENT – інформація про браузери, які використовує клієнт.

Для доступу до змінних оточення можна використовувати масив $_SERVER або функцію getenv(). Наприклад, $_SERVER["QUERY_STRING"] або getenv("QUERY_STRING") повернуть рядок запиту, а $_SERVER["REMOTE_ADDR"] або getenv("REMOTE_ADDR") – IP-адрес користувача, що послав запит.

Усередині PHP-скрипта існує декілька способів доступу до даних, які передані клієнтом за протоколом HTTP. По-перше, можна використовувати асоціативні масиви $_POST і $_GET, звернення до значень яких проводиться за іменами змінних форми. Наприклад, якщо пара name=Nina передана методом GET, то $_GET["name"] поверне "Nina".

По-друге, для звернення до змінних, переданих за допомогою HTTP-запитів, існує спеціальний масив $_REQUEST. Він містить дані, передані методами POST і GET, а також за допомогою HTTP cookies. Його значення можна отримати, використовуючи як індекс ім'я відповідної змінної (елемента форми), наприклад $_REQUEST["name"].

У третіх, в разі використання відповідних налаштувань (register_globals=On у файлі налаштувань) доступ до змінних форми може здійснюватися безпосередньо за їх іменами, тобто можна використовувати просто $name.

Приклад 10. Розробити сторінку з формою для голосування і скрипт для збереження, оброблення і пред'явлення результатів. Форму і виведення результатів вбудувати в довільну сторінку.

І форма і результати поміщені всередину блоку div з id = "gol" для забезпечення можливості форматування і верстки. Код форми:

```
<div id="gol">
<form action="golos.php" metod="GET">
<h2> Чи згоден ти з тим, що ...</h2>
<input type="radio" name="g" value="1" /><span>ТАК</span><br />
<input type="radio" name="g" value="2" /><span>НІ</span><br />
<input type="radio" name="g" value="3"/><span> Важко відповісти </span><br
/>
<input type="submit" />
</form>
</div>
```

На рис. 17 наведено фрагмент сторінки з формою.

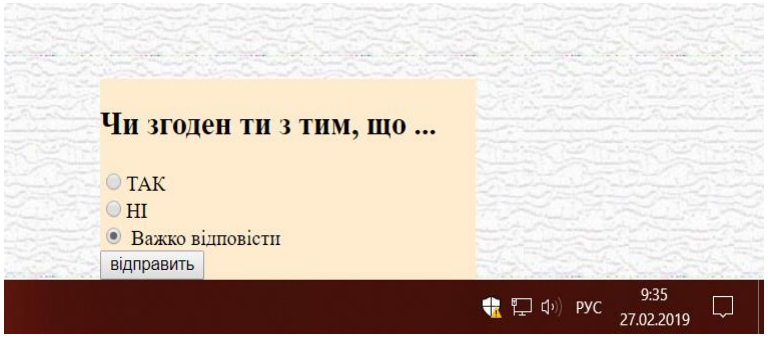


Рис. 17. **Фрагмент сторінки з формою для голосування**

Результати голосування накопичуються в текстовому файлі golos.txt.

Скрипт для оброблення і видачі результатів (файл golos.php):

```
<?php
$fp=fopen("golos.txt","r");// відкрити для читання файл з даними
$y=(int)fgets($fp);// читання числа тих, хто проголосував ЗА
$n=(int)fgets($fp);// читання числа тих, хто проголосував ПРОТИ
$v=(int)fgets($fp);// читання числа тих, хто УТРИМАВСЯ
$g=(int)fgets($fp);// читання числа тих, хто голосував
fclose($fp);                    // закрити файл
$g++;                           // зміна числа тих, хто голосував
if ($_GET["gl"]==1) $y++;// аналіз даних з форми
if ($_GET["gl"]==2) $n++;// і підрахунок результатів
if ($_GET["gl"]==3) $v++;// голосування
```

```
$s=$y."\n".$n."\n".$v."\n".$g."\n"; // формування рядка для запису у файл
$fp=fopen("golos.txt","w");         // відкрити для запису файл з даними
fwrite($fp,$s);                     // запис даних у файл
fclose($fp);                        // закрити файл
include 'otv.html';// вставити текст сторінки відповіді
?>
```

Сторінка відповіді є тією самою, яка пред'являлася користувачеві з формою. Однак замість форми блок з id="gol" містить таблицю, в яку вставлені змінні з даними ($y,$n,$v,$g):

```
<div id="gol">
<table width="100%">
<caption>Ми раді повідомити вам результати голосування:</caption>
<tr align="left"><th>ЗА<td><?php echo $y ?>
<tr align="left"><th>ПРОТИ<td><?php echo $n ?>
<tr align="left"><th>УТРИМАЛИСЬ<td><?php echo $v ?>
<tr align="left"><th>УСЬОГО ПРОГОЛОСУВАЛО<td><?php echo $g ?>
</table>
</div>
```

На рис. 18 наведено фрагмент сторінки з відповіддю (результатами голосування).

Рис. 18. **Фрагмент сторінки з результатами голосування**

3.3.2. Робота з файлами

Файлова система сервера являє собою зручне сховище даних для WEB-додатків, забезпечуючи спільне використання цих даних користувачами. Водночас PHP підтримує повноцінну роботу з файлами на сервері і надає засоби пересилки даних із машини користувача на сервер,

а пересилання файлів із сервера клієнту забезпечується стандартно протоколом HTTP.

Робота з файлами в PHP майже не відрізняється від інших мов, ті самі функції та послідовності дій:

відкриття файла з вказівкою режиму оброблення;

читання або запис даних;

закриття файла.

Для відкриття файлів використовують функцію fopen("файл","реж"), файл – це шлях і ім'я файла або URL, реж – режим відкриття. Шлях може бути як абсолютним, так і відносним. Список режимів наведено в табл. 3.

Таблиця 3

Режими відкриття файла

Режим	Використання
r	Для читання з початку файла
r+	Для читання і запису з початку файла
w	Для запису з початку файла, існуючий файл очищується
w+	Для запису і читання з початку файла, існуючий файл очищується
a	Для додавання в кінець файла
a+	Для додавання і читання з кінця файла
b	Двійковий режим, використовується у поєднанні з останніми

Таким чином, допустимими є, наприклад, такі конструкції:

```
$fp = fopen( "C:/root/fm.txt", "r" );
$fp = fopen( "fm.txt", "w" );
$fp = fopen( "http://www.soft.ru/", "r" );
$fp = fopen( "ftp://www.soft.ru/", "w+" );
```

Під час використання URL знак "/" наприкінці є обов'язковим. Для виконання всіх дій з файлами користувач повинен володіти відповідними правами відносно до даного файла (каталогу). У разі успішного відкриття файла функція fopen повертає посилання на відкритий файл, у разі

помилки – false. Повідомлення про помилку можна подавити (щоб не переривати виконання скрипта) і проаналізувати код повернення:

```
@ $fp=fopen("root/dat.txt","a");
if (!$fp) {echo "<p>Проблема з відкриттям файла dat.txt"."</p></body></html>;
exit;}
```

Після відкриття виконуються дії з оброблення файла, вони полягають у послідовності операцій із запису або читання даних. Для запису використовуються функція fwrite() або її синонім fputs(). Наприклад:

fwrite(fp,str), тут fp – посилання на файл, отримане під час відкриття, str – записуваний рядок. У рядку можна використовувати різні роздільники для подальшого (під час читання) розпаковування даних. Приклад формування рядка для запису з використанням символу табуляції (запис із чотирма полями):

```
$outstr=$date1."\t".$date2."\t".$date3."\t".sdate4."\n";
```

Число записуваних рядків визначається логікою програми.

Для читання даних із файла слугує група різних функцій, найчастіше використовуються fgets(), fgetcsv(), file(). Операції читання зазвичай закінчуються після досягнення кінця файла. Перевірка кінця файла здійснюється за допомогою функції feof().

fgets($fp) – читання рядка з файла, $fp – покажчик на файл, повертає прочитаний рядок. Читання йде до символу кінця рядка, або до кінця файла.

fgetcsv($fp [,delimiter]) – схожа на fgets(), але прочитаний рядок розбивається на поля з урахуванням вказаного роздільника (delimiter) і записується в масив.

file($fp) – читає весь файл у масив рядків, кожен рядок – елемент масиву. Можна використовувати не відкриваючи файла, тоді як параметр указується шлях та ім'я файла або URL.

У ході оброблення файла можна використовувати ще декілька корисних функцій file_exists() – перевірка існування файла, filesize() – визначення розміру файла в байтах, flock() – блокування файла для інших користувачів і т. д.

Після завершення роботи з файлом його слід закрити за допомогою функції fclose($fp), вона повертає true в разі успішного закриття файла і false інакше.

Приклад 11. Як приклад збереження даних у файлі з подальшим переглядом розглянемо ведення гостьової книги (збереження у файлі текстів, що вводяться користувачами під час роботи зі сторінкою).

Файл index.php виводить у вікно браузера форму для заповнення користувачем. Після заповнення полів формі і відправки проводиться перевірка наявності даних хоча б в одному полі форми. Якщо є непусті поля, то проводиться заміна у введеному тексті символів кінця рядка на табуляцію та формування рядка даних ($out_str), який дописують у файл guest.txt. Якщо файла немає, він буде створений.

Файл index.php:

```
<html>
<head>
<title>Сторінка відгуків</title>
</head>
<body>
<form action='index.php' method=POST>
<p>Залиште свій відгук</p>
<p>Введіть своє ім'я</p><input type=text name='name'>
<p>Введіть адресу електронної пошти</p><input type=text name='mail'>
<p>Введіть свій відгук</p><TEXTAREA name='cont' rows=5
COLS=72></TEXTAREA>
<input type=submit value='Надіслати'>
<input type=reset value='зброс'>
</form>
<?php
if ($_REQUEST["name"]|$_REQUEST["mail"]|$_REQUEST["cont"]) {
        $my_name=$_REQUEST["name"];
        $my_mail=$_REQUEST["mail"];
        $my_cont=$_REQUEST["cont"];
        $my_cont=str_replace("\n","\t",$my_cont);
        $out_str=$my_name."\t".$my_mail."\t".$my_cont."\n";
        $fp=fopen("guest.txt","a");
        fwrite($fp,$out_str);
        fclose($fp);                                                    }
?>
<a href="guest_1.php" target="new">Подивитися усі відгуки</a>
</body>
</html>
```

```
</body>
</html>
```

Посилання Подивитися усі відгуки відкриває сторінку зі змістом файла guest.txt у вигляді таблиці. Файл guest_1.php:

```
<html>
<head>
<title>Зміст відгуків</title>
</head>
<body>
<?php
if (file_exists("guest.txt")) $m_str=file ("guest.txt");
else {echo "<p>Проблема з файлом guest.txt</p></body></html>"; exit; };
echo "<table border=1>";
echo "<TR><TD>ім'я<TD>ел. пошта<TD>відгук";
foreach($m_str as $st) {$s=explode ("\t",
$st);echo"<TR><TD>".$s[0]."<TD>".$s[1]."<TD>".$s[2];};
echo "</table>";
?>
</body>
</html>
```

З клієнта (браузера) на сервер протоколом HTTP можуть бути відправлені не тільки дані з полів форми, але і файли цілком.

Оскільки можливості браузера з доступу до файлової системи клієнта обмежені, то для вибору файла на відправку в HTML є спеціальний елемент <input type="file"/>. Під час натисканні мишею відкривається вікно для вибору файла. Після натискання кнопки submit файл буде відправлений на сервер. Відправлення здійснюється методом POST, форма повинна містити атрибут enctype = "multipart / form-data".

Приклад форми для відправки файла:

```
<form name="myform_1" action="send_file.php" enctype="multipart/form-data"
method="post">
<fieldset style="border:1px solid #ccc; width:450px; background:#f9f9f9; text-
align:left;">
<legend style="font-size:14pt">Відправка файла на сервер</legend>
<input name="my_file" type="file">
<input type="image" src="imgbutton.gif">
</fieldset>
</form>
```

На сервері файл поміщується в папку для тимчасових файлів, а вся інформація про нього – в змінну $ _FILES:

$_FILE['my_file']['name'] – ім'я обраного файла (наприклад, uploadme.txt);

$_FILE['my_file']['type'] – тип обраного файла (наприклад, image/jpg);

$_FILE['my_file']['size'] – розмір обраного файла в байтах;

$_FILE['my_file']['tmp_name'] – ім'я і каталог для тимчасового розміщення файла на сервері;

$_FILE['my_file']['error'] – код помилки, виникає в тому випадку, якщо спроба завантаження була невдалою.

З урахуванням цієї інформації скрипт для копіювання файла в папку сайта для подальшого використання може виглядати так:

```
<?php
$myfile = $_FILES["my_file"]["tmp_name"];
$myfile_name = basename($_FILES["my_file"]["name"]);
 if(!copy($myfile, $myfile_name)) echo 'Помилка під час копіювання файла ';
 else                             echo " Файл успішно скопійований ";
?>
```

Функція basename повертає оригінальне ім'я файла без зайвих символів, а copy копіює файл із тимчасової папки в поточну папку (папку сайта).

Таким чином PHP дозволяє досить ефективно використовувати сервер для зберігання даних, створювати різни сховища (наприклад, фото або аудіо).

Висновки й узагальнення

Розвиток технологій мережі Інтернет привело до масового розроблення і використання WEB-додатків – додатків, які виконуються на двох взаємодіючих у мережі машинах (клієнта і сервера). Частина логіки додатка виконується на машині клієнта з використанням скриптів, а частина – на сервері. Для сервера запропоновано багато технологій, найбільшого поширення набули три: PHP, ASP.NET, Node.js. У розділі було розглянуто основи технології PHP, як найбільш поширеної та доступної.

Рознесені частини додатків, а також додатки між собою обмінюються даними. Для опису таких структурованих даних розглянута мова XML.

Для налагодження WEB-додатків, що розробляються з використанням PHP, на комп'ютері повинен бути встановлений WEB-сервер с інтерпретатором PHP та виконані відповідні налаштування.

Теоретичні запитання

1. Який із підходів до створення динамічних сторінок представляється вам найбільш універсальним? Чому?
2. Який із підходів до створення динамічних сторінок представляється вам найбільш простим в реалізації? Чому?
3. Поясните зв'язок між розширеннями сервера і технологіями, заснованими на скриптах, вбудованих в сторінку.
4. Дайте характеристику інтерфейсу CGI?
5. У чому полягає відмінність технологій CGI і ISAPI?
6. Як здійснюється доступ до змінних оточення в середовищі VS?
7. Чим відрізняється передача даних під час використання методів GET і POST?
8. Які цілі ставили перед собою розробники XML?
9. Сформулюйте основні правила розмітки з використанням XML.
10. Для чого служить секція <! CDATA [.....]]>?
11. Які частини можна виділити в структурі XML-документа?
12. Які атрибути можуть бути присутніми в XML-оголошенні?
13. Як виглядають інструкції з оброблення XML-документів, для чого вони призначені?
14. Назвіть області, де може знайти застосування мову XML.
15. Які дії виконує браузер під час відображення XML-документів?
16. Порівняйте JSON і XML як способи обміну даними між додатками.
17. Для чого було розроблено XSL?
18. Як здійснюється XSLT-перетворення?
19. Що таке шаблон в XSL-таблиці і як він використовується?
20. Які компоненти забезпечують опис структури XML-документа?
21. Що таке валідність? Для чого створюються DTD?
22. Сформулюйте переваги мови PHP для створення динамічних сторінок.
23. Сформулюйте правила розміщення PHP-скриптів на сторінках.

24. Сформулюйте відмінності в синтаксисі PHP і знайомих вам мов C і JavaScript.

25. Назвіть способи отримання доступу до даних із форм у PHP.

26. Перерахуйте послідовність дій для отримання доступу до даних у БД. Назвіть функції для підтримки цих дій.

27. Дайте характеристику середовищ для розроблення додатків на PHP.

28. Дайте характеристику області дії імен в мові PHP.

29. Як можуть передаватися параметри у функції на PHP?

Контрольні завдання

1. Доповнити необхідними даними та відобразити такий XML-документ у вікні браузера за допомогою XSLT:

```
<DOCUMENT> Список студентів, що не здали книги
  <STUDENT>
   <NAME>Іванов</NAME>
   <photo>im1.gif</photo>
   <DATA>10.04.176</DATA>
  </STUDENT>
<STUDENT>
   <NAME>Петренко</NAME>
   <photo>im2.gif</photo>
   <DATA>10.04.176</DATA>
  </STUDENT>
</DOCUMENT>
```

2. Доповнити необхідними даними та відобразити такий XML-документ у вікні браузера за допомогою XSLT:

```
<INVENTORY>
  <BOOK>
    <TITLE>Приключения Гекельберри Финна</TITLE>
    <AUTHOR>
      <FIRSTNAME>Марк</FIRSTNAME>
      <LASTNAME>Твен</LASTNAME>
    </AUTHOR>
    <BINDING>м'який</BINDING>
    <PAGES>298</PAGES>
```

```
    <PRICE>49</PRICE>
  </BOOK>
  <BOOK>
    <TITLE>Приключения Тома Сойера</TITLE>
    <AUTHOR>
      <FIRSTNAME>Марк</FIRSTNAME>
      <LASTNAME>Твен</LASTNAME>
    </AUTHOR>
    <BINDING>твердий</BINDING>
    <PAGES>205</PAGES>
    <PRICE>40</PRICE>
  </BOOK>
 </INVENTORY>
```

3. Доповнити необхідними даними та відобразити такий XML-документ у вікні браузера за допомогою XSLT:

```
<report>
<header>
<supplier>Mic Co.</supplier>
<client>Labaz inc.</client>
</header>
<rows>
<row price="1" quantity="2" total="2"><name>минтай</name></row>
<row price="2" quantity="1" total="2"><name>оселедець</name></row>
<row price="3" quantity="4" total="12"><name>горбуша</name></row>
</rows>
<rows>
<row price="1" quantity="1" total="3"><name>трепанги</name></row>
<row price="2" quantity="1" total="3"><name>корюшка</name></row>
<row price="3" quantity="1" total="11"><name>тунець</name></row>
</rows>
</report>
```

4. Розробити додаток за технологією PHP, що забезпечує завдання фону сторінки відповідно до вибору користувача (обрано за допомогою перемикачів).

5. Розробити додаток за технологією PHP, що забезпечує зміну надпису на кнопці відповідно до вибору користувача (обрання зі списку).

6. Розробити додаток за технологією PHP, що забезпечує зміну малюнка на сторінці відповідно до вибору користувача зі списку.

Розділ 2. Технологія PHP та її використання

Розділ присвячено поширенню можливостей використання мови PHP для створення серверних додатків. У ньому розглядаються різні підходи і технології взаємодії серверної та клієнтської частин WEB-додатків, застосування масштабуємої (SVG) та динамічної серверної графіки.

Завдяки поданим матеріалам студенти мають ознайомитися з розвинутими технологіями взаємодії браузера на машині користувача із сервером, фреймворками можливо використати для високорівневої взаємодії серверної та клієнтської частин WEB-додатків, зокрема серверною графікою, що масштабується та є динамічною, яке програмне забезпечення повинно бути використане для розробки розподілених програмних додатків, зокрема (**IDE**-Integrated development environment) для створення конкретних різновидів додатків.

Ключові поняття розділу: інтегровані системи розробки WEB-сервісів, синхронна та асинхронна взаємодія клієнтської та серверної частин додатку, високорівневий інтерфейс взаємодії, фреймворки для забезпечення взаємодії.

Питання розділу:

Тема 4. Технології активних сторінок PHP

- 4.1. Загальна характеристика технології PHP
- 4.2. Робота з базами даних
- 4.3. Асинхронна взаємодія клієнта із сервером
- 4.4. Використання фреймворків

Тема 5. Динамічна графіка на WEB-сторінках

- 5.1. Графічні елементи HTML5
- 5.2. Створення зображень на сервері

Цілі вивчення розділу

Інформація, що подана в розділі, надає студентові можливість сформувати такі **компетентності:**

здатність обґрунтувати вибір технології високорівневої взаємодії серверної та клієнтської частин WEB-додатків;

здатність створювати WEB-додатки з використанням високорівневої взаємодії серверної та клієнтської частин;

здатність виконувати розроблення WEB-додатків із динамічною серверною графікою;

здатність виконувати розробку та відлагодження додатків мовою PHP;

знання:

основних принципів функціонування найбільш поширених технологій високорівневої взаємодії серверної та клієнтської частин серверів;

основні технології виконання програм на з розподіленою архітектурою;

особливості високорівневих технологій активних сторінок, засоби їх створення;

високорівневі засоби фреймворків забезпечення програмування мовою PHP;

уміння:

обирати засоби, методи і технології для створення WEB-ресурсів із засобами високорівневої взаємодії серверної та клієнтської частин додатків;

використовувати додатки мовою PHP на WEB-ресурсах;

створювати динамічні сторінки і обробляти дані з клієнтських форм, використовуючи фреймворки;

комунікації:

доведення своїх висновків і результатів роботи до учасників команди;

уміння користуватися консультацією фахівців для вирішення проблемних питань;

автономність і відповідальність:

пошук шляхів вирішення проблем, що виникають під час функціонування високорівневих компонент сервісу WWW;

пошук альтернативних засобів для роботи з WEB-ресурсами;

самостійний пошук та вибір засобів для роботи з векторною та динамічною графікою на сервері.

Вступ

Широке поширення мобільних пристроїв зажадало прискорити перехід до нових архітектурних принципів і інтернет-технологій. В області організації комунікації "інтернет-клієнт – веб-сервер" – це, перш за все, асинхронна взаємодія. В області побудови додатків з насичених графікою це фреймворки для використання масштабованої векторної і динамічної графіки, саме такі технології буде розглянуто в цьому розділі.

Тема 4. Технології активних сторінок PHP

4.1. Загальна характеристика технології PHP

4.1.1. Взаємодія клієнта із сервером

Крім засобів взаємодії клієнта із сервером існує ціла низка методів, технологій та архітектур.

Зупинимося докладно на архітектурі **REST** (Representational State Transfer). Це скорочення позначає в перекладі: подання даних для клієнта у форматі зручному для нього. REST-підхід, архітектурний стиль до написання прикладних інтерфейсів.

RESTful API зводиться до чотирьох базових операцій:

отримання даних у зручному для клієнта форматі;

створення нових даних;

оновлення даних;

видалення даних.

REST функціонує поверх протоколу **HTTP**, тому варто згадати про його основні особливості. Для кожної операції зазначеної вище використовується свій власний **HTTP** метод:

GET-отримання;

POST-створення;

PUT-оновлення, модифікація;

DELETE-видалення.

Усі ці методи в сукупності називають **CRUD** (Create, Read, Update, Delete) – (створити, прочитати, оновити, видалити) операціями.

Вимоги до архітектури REST:

модель клієнт-сервер,

відсутність стану,

кешування,

однаковість інтерфейсу,

шарова побудова.

Протокол **SOAP** (від англ. **S**imple **O**bject **A**ccess **P**rotocol – простий протокол доступу до об'єктів) – протокол обміну структурованими повідомленнями в розподіленому обчислювальному середовищі. Спочатку SOAP призначався в основному для реалізації віддаленого виклику про-

цедур (RPC). Зараз протокол використовується для обміну довільними повідомленнями у форматі XML, а не тільки для виклику процедур.

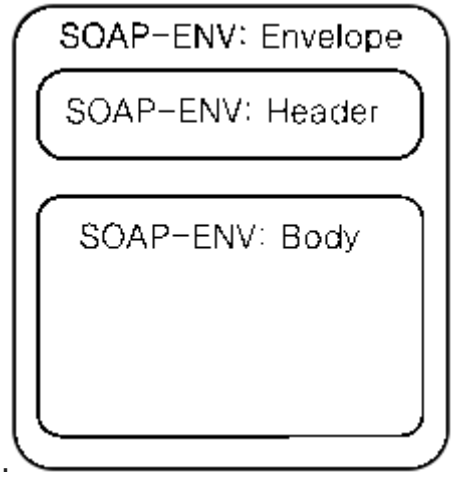


Рис. 19. **Структура SOAP-повідомлення**

SOAP може використовуватися з будь-яким протоколом прикладного рівня: SMTP, FTP, HTTP, HTTPS тощо. Однак його взаємодія із кожним із цих протоколів має свої особливості, які повинні бути визначені окремо. Найчастіше SOAP використовується поверх HTTP.

SOAP є одним із стандартів, на яких базуються технології веб-служб.

SOAP використовується зараз у деяких консервативних сферах.

WEB-служба, WEB-сервіс (англ. web service) – ідентифікована унікальною WEB-адресою (URL-адресою) програмна система зі стандартизованими інтерфейсами, а також HTML-документ сайта, що відображається браузером користувача.

WEB-служби можуть взаємодіяти один з одним і зі сторонніми додатками за допомогою повідомлень, заснованих на певних протоколах (SOAP, XML-RPC і т. д.) і угодах (REST). WEB-служба є одиницею модульності під час використання сервіс-орієнтованої архітектури.

AJAX, Ajax (Asynchronous **Ja**vascript and **X**ML – "асинхронний JavaScript і XML") – підхід до побудови інтерактивних інтерфейсів вебдодатків, що полягає в фоновому обміні даними браузера з веб-сервером та буде детально розглянуто в наступному розділі.

WebSocket – протокол зв'язку поверх TCP-з'єднання, призначений для обміну повідомленнями між браузером і WEB-сервером у режимі реального часу.

У даний час в W3C здійснюється стандартизація API WEB Sockets. Чорновий варіант стандарту цього протоколу затверджено IETF.

WebSocket розроблений для втілення у веб-браузерах і веб-серверах, але він може бути використаний для будь-якого клієнтського або серверного додатка. Протокол WebSocket – це незалежний протокол, заснований на протоколі TCP. Він робить можливим більш тісну взаємодію між браузером і WEB-сайтом, сприяючи поширенню інтерактивного вмісту і створенню додатків реального часу.

Для установки з'єднання клієнтський скрипт створює об'єкт WebSocket, в конструктор якого передає параметр WebSocket URI, і визначає функції зворотного виклику під час з'єднання, отримання повідомлення і розриві з'єднання.

Приклад 12. Створення об'єкта WebSocket, в конструктор якого передає параметр WebSocket URI, і визначення функції зворотного виклику під час з'єднання, отриманні повідомлення і розриві з'єднання.

```
<html>
   <head>
      <script>
         const webSocket = new WebSocket('ws://localhost/echo');

         webSocket.onopen = event => {
            alert('onopen');
            webSocket.send("Hello Web Socket!");
         };

         webSocket.onmessage = event => {
            alert('onmessage, ' + event.data);
         };

         webSocket.onclose = event => {
            alert('onclose');
         };
      </script>
   </head>
   <body>
   </body>
</html>
```

4.1.2. Обробка XML-документів

Нехай є XML. Він може бути в рядку, або зберігається у файлі чи віддаватися за запитом до певного URL.

Нехай XML зберігається в рядку. У цьому випадку із цього рядка потрібно створити об'єкт за допомогою new SimpleXMLElement:

```
$str = "<?xml version='1.0'?>
<worker>
        <name>Коля</name>
        <age>25</age>
        <salary>1000</salary>
</worker>";

$xml = new SimpleXMLElement($str);
```

Зараз у змінній $ xml зберігається об'єкт з розібраним XML. Звертаючись до властивостей цього об'єкта можна отримувати доступ до вмісту тегів XML.

Якщо ж XML зберігається у файлі або віддається зі зверненням до URL (що найчастіше і відбувається), то слід використовувати функцію simplexml_load_file, яка робить той же самий об'єкт $xml:

```
<?xml version='1.0'?>
<worker>
        <name>Коля</name>
        <age>25</age>
        <salary>1000</salary>
</worker>

$ xml = simplexml_load_file (шлях до файла або урл);
```

Прийоми роботи. У прикладах нижче XML зберігається у файлі або по URL.

Нехай дано наступний XML:

```
<?xml version='1.0'?>
<worker>
        <name>Коля</name>
        <age>25</age>
        <salary>1000</salary>
```

```
</worker>
Давайте отримаємо ім'я, вік і зарплату працівника:

$ xml = simplexml_load_file (шлях до файла або урл);
echo $xml->name; //виведе 'Коля'
echo $xml->age; //виведе 25
echo $xml->salary; //виведе 1000
```

Об'єкт $xml має властивості, що відповідають тегам.

Тег <worker> ніде не фігурує під час звернення. Це тому, що він кореневий тег. Можна перейменувати його, наприклад, на <root> – і нічого не зміниться:

```
<?xml version='1.0'?>
<root>
	<name>Коля</name>
	<age>25</age>
	<salary>1000</salary>
</root>
$ xml = simplexml_load_file (шлях до файла або урл);
echo $xml->name; //виведе 'Коля'
echo $xml->age; //виведе 25
echo $xml->salary; //виведе 1000
```

Кореневий тег в XML може бути тільки один, так само, як і тег <html> у звичайному HTML.

Трохи модифікуємо наш XML:

```
<?xml version='1.0'?>
<root>
	<worker>
		<name>Коля</name>
		<age>25</age>
		<salary>1000</salary>
	</worker>
</root>
```

У цьому випадку вийде ланцюжок звернень:

```
$ xml = simplexml_load_file (шлях до файла або урл);
echo $xml->worker->name; //виведе 'Коля'
echo $xml->worker->age; //виведе 25
echo $xml->worker->salary; //виведе 1000
```

Робота з атрибутами.

Нехай деякі дані зберігаються в атрибутах:

```
<?xml version='1.0'?>
<root>
      < worker name= "Коля" age= " 25 "salary=" 1000 " >Номер 1< / worker>
</root>
$ xml = simplexml_load_file (шлях до файла або урл);
echo $xml->worker['name']; //виведе 'Коля'
echo $xml->worker['age']; //виведе 25
echo $xml->worker['salary']; //виведе 1000
echo $xml->worker; //виведе 'Номер 1
```

Використання шаблонів. Шаблони у PHP використаються як інструмент для об'єднання даних та інтерфейса (відокремити бізнес-логіку і логіку уявлення), так і для маніпулювання текстами.

Шаблонізатор (у WEB) – програмне забезпечення, що дозволяє використовувати html-шаблони для генерації кінцевих html-сторінок. Основна мета використання шаблонізаторів – це відділення представлення даних від виконуваного коду. Часто це необхідно для забезпечення можливості паралельної роботи програміста і дизайнера-верстальника. Використання шаблонізаторів часто покращує читаність коду і внесення змін у зовнішній вигляд, коли проєкт повністю виконує одна людина.

Для цього потрібно дотримуватись наступних умов:

не використовувати в шаблонах логіку додатка, передавати в шаблони тільки дані, отримані зі скрипта – скаляри, масиви, об'єкти. Ніяких викликів до бази, алгоритмів не пов'язаних із логікою відображення тощо;

використовувати в шаблонах структури управління PHP, необхідні для логіки відображення – IF / ELSEIF / ELSE, FOR / FOREACH, INCLUDE / REQUIRE;

використовувати для структур управління альтернативний синтаксис – він дуже легко розпізнати об'єкти HTML-шаблонів;

намагатися не використовувати вбудовані функції в шаблонах. Якщо навіть вам потрібно застосувати в шаблоні досить часто використовувану функцію htmlspecialchars, то не полінуйтеся обернути її в статичний метод класу-помічника або у функцію. Це в подальшому дасть більший простір для рефакторинга і просто створить стиль вашого API.

Такий стиль шаблонізації на PHP називається pure-шаблонізації, тобто чиста шаблонізація, заснована на можливостях самого PHP.

Під час використання pure-шаблонізації код нашого скрипта і шаблону міг би виглядати так:

```
<? Php
$ A = isset ($ _ GET [ 'a']) && is_numeric ($ _ GET [ 'a'])? $ _GET [ 'a']: 0;
$ B = isset ($ _ GET [ 'b']) && is_numeric ($ _ GET [ 'b'])? $ _GET [ 'b']: 0;

$ Result = $ a + $ b;

// завантажуємо шаблон
include ( 'template.html');

Шаблон:

<! DOCTYPE HTML PUBLIC "- // W3C // DTD HTML 4.0 Transitional // EN">
<Html>
<Head>
   <Title> Основний шаблон HTML-сторінки </ title>
</ Head>
<Body>
   <? Php if ($ result):?>
      <Span style = "color: blue; font-weight: bold"> Результат: <? = $ Result?> </ Span>
   <? else:?>
      <Span style = "color: red; font-weight: bold"> Результат дорівнює нулю </ span>
   <? endif; ?>
</ Body>
</ Html>
```

Так можливо розділити логіку програми та логіку відображення. Тепер верстальник, знайомий з тривіальними керуючими конструкціями будь-якої мови програмування, може з легкістю підтримувати HTML-код, а програмісту немає необхідності що-небудь знати про те, як і де буде виведений результат роботи програми. Так розділюються обов'язки і створюється код, який легко модифікувати.

Найчастіше шаблони для обробки текстових даних задаються у вигляді регулярних виразів.

Функції регулярних виразів. У PHP існує кілька функцій для роботи з регулярними виразами. Усі вони використовують один і той самий парсер регулярних виразів для своєї роботи, але водночас переслідують різні цілі. Розглянемо всі ці функції. Опис синтаксису кожної функції в тому вигляді, в якому вона описана в PHP Manual.

Функція preg_match()
Синтаксис:

```
int preg_match (string pattern, string subject [, array matches])
```

Ця функція призначена для перевірки того, чи збігається заданий рядок (subject) із заданим регулярним виразом (pattern). Як результат функція повертає 1, якщо збіги були знайдені і 0, якщо немає. Якщо під час виклику функції було задано необов'язковий параметр matches, то після роботи функції йому буде присвоєно масив, що містить результати пошуку за заданим регулярним виразом. Незалежно від того, скільки саме збігів було знайдено при пошуку повернуто тільки перше збіг. Розглянемо приклад того, як це працює:

```
<?php
$str = "123 234 345 456 567"; // Рядок для пошуку
$result = preg_match('/\d{3}/',$str,$found); // Робимо пошук
echo "Matches: $result<br>"; // Виводимо кількість знайдених збігів
print_r($found); // Виводимо результат пошуку
?>
```

Результатом роботи цієї програми буде:

```
Matches: 1

Array
(
    [0] => 123
)
```

Реально функція preg_match() виявила у зазначеному рядку п'ять збігів із заданим виразом, але повернула лише перше з них. Здавалося б,

що в цьому випадку було б логічніше повертати результати пошуку у вигляді рядка, а не у вигляді масиву, але це не так. Регулярний вираз може містити в собі внутрішні регулярні вирази, які також повертають результат. А для того, щоб повернути результати пошуку за всіма регулярними виразами і потрібен масив. Змінимо регулярний вираз і подивимося на результат:

```
<?php
$str = "123 234 345 456 567";
// Тепер ми не просто шукаємо тризначне число,
// але і отримуємо його середню цифру
$result = preg_match('/\d\d\d/',$str,$found);
echo "Matches: $result<br>";
print_r($found);
?>
```

Результат буде наступним:

```
Matches: 1

Array
(
   [0] => 123
   [1] => 2
```

Присутні результати пошуку за всіма наявними регулярними виразами.

Функція preg_match_all()
Синтаксис:

```
int preg_match_all (string pattern, string subject, array matches [, int order])
```

Ця функція дуже схожа на попередню і призначена для тих самих цілей. Єдина її відмінність від preg_match() полягає в тому, що вона здійснює

"глобальний" пошук в заданому тексті за заданим регулярним виразом і, відповідно, знаходить і повертає всі наявні збігу.

Як відрізняється робота цієї функції на тому самому прикладі:

```
<?php
$str = "123 234 345 456 567";
$result = preg_match_all('/\d{3}/',$str,$found);
echo "Matches: $result<br>";
print_r($found);
?>
```

Результат роботи:

```
Matches: 5
  Array
  (
     [0] => Array
        (
           [0] => 123
           [1] => 234
           [2] => 345
           [3] => 456
           [4] => 567
        )
)
```

Отримано всі знайдені збіги та їх кількість в якості результату.

Необхідно звернути увагу на додатковий параметр, що з'явився в цій функції порівняно з preg_match (): order. Значення цього параметра визначає структуру вихідного масиву зі знайденими збігами. Його значення може бути одним із перерахованих нижче.

PREG_PATTERN_ORDER – результати пошуку будуть згруповані за номером регулярного виразу, яке повернуло цей результат (це значення використовується за замовчуванням).

PREG_SET_ORDER – результати пошуку будуть згруповані за місцем їх знаходження в тексті.

Для того, щоб краще зрозуміти різницю між цими значеннями, подивимося на результат роботи одного і того самого скрипта під час використання кожного з них:

Спочатку подивимося на те, як виглядає результат під час використання PREG_PATTERN_ORDER,

```
<?php
$str = "123 234 345 456 567";
$ order = PREG_PATTERN_ORDER;
$result = preg_match_all('/\d\d\d/',$str,$found,$order);
print_r($found);
?>
```

Результат:

```
Array
 (
    [0] => Array
       (
          [0] => 123
          [1] => 234
          [2] => 345
          [3] => 456
          [4] => 567
       )
    [1] => Array
       (
          [0] => 2
          [1] => 3
          [2] => 4
          [3] => 5
          [4] => 6
       )
)
```

Масив результатів містить зовнішні індекси, відповідні номерам регулярних виразів, від яких отримано результат (індекс 0 має основне регулярний вираз). За цими індексами в масиві розташовані масиви, що містять безпосередньо знайдену інформацію, причому індекс у цьому

внутрішньому масиві відповідає порядковому номеру" даного фрагмента у вихідному тексті.

Тепер спробуємо те саме, але з PREG_SET_ORDER:

```
<?php
$str = "123 234 345 456 567";
$ order = PREG_SET_ORDER;
$result = preg_match_all('/\d\d\d/',$str,$found,$order);
print_r($found);
?>
```

Результат:

```
Array
 (
    [0] => Array
        (
            [0] => 123
            [1] => 2
        )
    [1] => Array
        (
            [0] => 234
            [1] => 3
        )
    [2] => Array
        (
            [0] => 345
            [1] => 4
        )
    [3] => Array
        (
            [0] => 456
            [1] => 5
        )
    [4] => Array
        (
            [0] => 567
            [1] => 6
        )
)
```

Тут основний масив містить результати пошуку, згруповані за порядком їх знаходження в тексті, причому кожен результат являє собою масив з результатами пошуку за цим знайденого фрагмента для всіх наявних регулярних виразів.

Функція preg_replace()
Синтаксис:

mixed preg_replace (mixed pattern, mixed replacement, mixed subject [, int limit])

Ця функція дозволить зробити заміну тексту за регулярним виразом. Як і в попередніх функціях, тут проводиться пошук за регулярним виразом pattern в тексті subject, і кожен знайдений фрагмент тексту замінюється на текст, заданий у replacement. Завдання необов'язкового параметра limit дозволить обмежити кількість замінних фрагментів у тексті.

Наприклад, нам необхідно "стиснути" текст, прибравши з нього всі зайві прогалини і символи перекладу рядка.

```
<?php
$text = "there is\t\n\t\t some text \n \t just \n\n\n for test";
echo "<b>Перед заміною:</b>\n$text\n\n";
$text = preg_replace("/(\n \s{2,})/"," ",$text);
echo "<b>Після заміни:</b>\n$text";
?>
```

Результатом роботи даної програми буде наступний текст.

Перед заміною:

```
there is
        some text
    just

 for test
```

Після заміни.

there is some text just for test

Всього один рядок дозволив вирішити досить нетривіальну у звичайній практиці завдання.

Однак основна принадність цієї функції, яка і надає їй всю її міць – це той факт, що можливо посилатися на результати пошуку під час генерації заміщає текст. Приклад – конвертація дат з одного формату в інший. На Заході зазвичай використовується формат mm/dd/yyyy, тоді як у нас зазвичай – dd.mm.yyyy. Наступний приклад здійснює конвертацію дат між цими форматами в заданому тексті.

```
<?php
$text = 'Today is 11/16/2001';
$text = preg_replace("/(\d{2})\/(\d{2})\/(\d{4})/","\\2.\\1.\\3",$text);
echo $text;
?>
```

Результат роботи цієї програми.

```
Today is 16.11.2001
```

4.2. Робота з базами даних

Характеристика баз даних, існують різноманітні сервери баз даних, та достатньо багато принципів типізації – за архітектурою, за типом контенту, за операціями, які виконуються найбільш час.

Для побудови серверних додатків використаються безкоштовних систем керування контентом, найчастіше з кодом написаним на PHP. Як сервер бази даних часто використовуються такі СКБД як пропрієтарні MS SQL Server, Oracle, DB2, Informix або безкоштовні MySQL, PostgreSQL, SQLite, Firebird.

Останнім часом під час роботи з мультимедійним контентом набувають більшого поширення СКБД з реляційною, постреляційною і нереляційною архітектурами.

Налаштування доступу. Щоб php скрипт міг працювати з базою даних, йому необхідно встановити з'єднання з базою. Для цього скрипт повинен розуміти, за якою адресою встановлювати з'єднання, яку базу шукати та який пароль надати серверу бази даних, щоб одержати доступ

до інформації. Задамо константи необхідні для установки з'єднання із сервером баз даних.

Приклад оголошення констант для підключення до бази.

```
<?php
    define('HOST', 'localhost');
    define('USER', 'root');
    define ('PASSWORD','');
    define('NAME_BD', 'myBase');
?>
```

Тепер необхідно скористаємося командою mysql_connect(). У разі успішного з'єднання функція поверне покажчик на з'єднання з MySQL, в іншому випадку повернеться прапор FALSE.

Приклад підключення до бази даних.

```
<?php
   $ connect = mysql_connect (HOST, USER, PASSWORD)
      or die("Неможливо встановити з'єднання"
           .mysql_error( ));
   print ("З'єднання з базою встановлено.");
?>
```

У цьому прикладі задіяна ще одна команда mysql_error (), що належить до роботи СБД. З назви видно, що вона повертає помилку через, яку не вдалося з'єднатися з базою. Іноді буває корисним дізнатися подробиці помилки, тому не варто нехтувати такою можливістю.

Підключившись до сервера бази, потрібно повідомити йому про те, з якою базою ми збираємося працювати, для цього нам доведеться виконати ще один код.

Але спочатку слід обрати базу.

```
<?php
    mysql_select_db(NAME_BD, $connect)
       or die ("Неможливо вибрати зазначену базу"
            .mysql_error( ));
?>
```

Базу даних обрано і тепер можемо робити з нею все, що нам заманеться, наприклад, виконати найпростіший запит до БД.

Як виконати запит до бази.

Для початку придумаємо SQL запит, наприклад такий.

```
SELECT COUNT (*) FROM myBase
```

Запит, що наведено вище, повинен повернути нам кількість записів у таблиці myBase. Для того, щоб виконати запит потрібно скористатися функцією mysql_query(). Взагалі функція приймає два параметри: перший – сам запит, другий – ідентифікатор підключення до бази, але так як ми використовуємо тільки одне підключення, то другий параметр можна опустити.

Приклад виконання запиту.

```
<?php
    $query = mysql_query("SELECT COUNT(*) FROM myBase")
    or die ("Помилка виконання запиту:".mysql_error( ));
    echo " запит виконано успішно.";
?>
```

Приклад запиту додавання таблиці.

```
CREATE TABLE myNewBase (id INTEGER, content CHAR (10));
```

Результатом такого запиту буде створена таблиця з ім'ям myNewBase, а також полями id і content. Виконаємо запит за допомогою команди mysql_query(), як показано в прикладі.

Приклад виконання запиту MySQL.

```
<?php
    $query = mysql_query("CREATE TABLE myNewBase (id integer, content
char(10));")
    or die ("Помилка виконання запиту:".mysql_error());
     echo " запит виконано успішно.";
?>
```

Після відкриття з'єднання з базою функцією mysql_connect(), необхідно закрити це з'єднання за допомогою відповідної функції mysql_close().

Приклад завершення роботи з базою.

```
<?PHP
    mysql_close($connect);
?>
```

Для того, щоб отримати інформацію з таблиць складемо запит MySQL вибірки, виконаємо його функцією mysql_query() і запишемо з змінну для подальшої роботи з отриманою інформацією:

Запис інформації з таблиці в БД змінну.

```
<? PHP
    $result=mysql_query ("виберіть * від користувачів");
?>
```

Тепер змінна $result результат містить у собі всю інформацію з таблиці mytable. Виведемо отримані дані у вигляді таблиці у форматі HTML.

Виведення даних із таблиці MySQL.

```
<?PHP
    $ sql = " SELECT * FROM users";
    $result = mysql_query ($sql) або die (mysql_error());
    $table = " <table>";
    while ($row = mysql_fetch_assoc($result))
    {
      $таблиця. = "<tr>";
      $таблиця. = "<ТД.">$row ['id']."</ТД>";
      $таблиця. = "<ТД.">$row ['login']."</ТД>";
      $таблиця. = "<ТД.">$row ['email']."</ТД>";
      $таблиця. = "</tr>";
     }
      $таблиця. = "</таблиця> ";
     таблиця echo $;
?>
```

4.3. Асинхронна взаємодія клієнта із сервером

Технологія AJAX, загальна характеристика.

AJAX (від англ. Asynchronous JavaScript and XML – асинхронний JavaScript і XML) – підхід до побудови інтерфейсів інтерактивних призначених для користувача WEB-серверних додатків, що полягає у "фоновому" обміні даними браузера з WEB-сервером. У результаті під час оновлення даних, WEB-сторінка не перезавантажується повністю і WEB-серверні додатки стають швидшими і зручнішими.

Сутність технології AJAX полягає в зміні вмісту завантаженої вебсторінки без її повного перезавантаження, завдяки чому досягається висока динамічність сайтів. Технологія ґрунтується на розділенні даних і перезавантаження тих або інших компонентів у разі потреби.

Застосування AJAX PHP може бути різноманітним, єдине, що можна конструювати елементи сторінки за допомогою даної технології, які несуть у собі релевантність для пошукових систем. Тому що AJAX довантажує елементи сторінки після її завантаження під час виклику js подій, але, як нам відомо, пошукові системи не вміють читати JavaScript код, тому потрібно ретельно вибирати де треба, а де не потрібно застосовувати AJAX PHP.

Засоби реалізації XMLHttpRequest набули популярності завдяки тому, що значно прискорював завантаження даних, дозволяючи включати у вже завантажену сторінку дані будь-якого типа. Його вживання дозволяло без особливих хитрувань вставляти в сторінку відповіді сервера і, таким чином, використовувати технологію AJAX змогли навіть люди, що не відрізняються особливими талантами. AJAX – не самостійна технологія, а концепція використання декількох суміжних технологій. AJAX базується на наступних основних принципах.

Використання технології динамічного звернення до сервера "на льоту", без перезавантаження всієї сторінки повністю, наприклад: із використанням XMLHttpRequest (основний об'єкт); через динамічне створення дочірніх фреймів; через динамічне створення тега <script>, використання DHTML для динамічної зміни вмісту сторінки.

Переваги використання AJAX наступні.

Економія трафіку. Використання AJAX дозволяє значно скоротити трафік під час роботи з WEB-серверним додатком завдяки тому, що часто замість завантаження всієї сторінки досить завантажити частину, яка змінилася, як правило, досить невелику.

Зменшення навантаження на сервер. AJAX дозволяє трохи понизити навантаження на сервер. Наприклад, на сторінці роботи з поштою, коли ви відзначаєте прочитані листи, серверу досить внести зміни в базу даних і відправити клієнтському скрипту повідомлення про успішне виконання операції без необхідності повторно створювати сторінку і передавати її клієнту.

Прискорення реакції інтерфейсу. Оскільки потрібно завантажити лише ту частину, що змінилася, користувач бачить результат своїх дій швидше.

Недоліки технології такі.

Відсутність інтеграції із стандартними інструментами браузера. Динамічно створювані сторінки не реєструються браузером в історії переглядів сторінок, тому не працює кнопка "Назад", що надає користувачам можливість повернутися до сторінок що, були переглянуті раніше, але існують скрипти, які можуть вирішити цю проблему.

Інший недолік зміни вмісту сторінки за постійного URL полягає в **неможливості збереження закладки на бажаний матеріал**. Частково вирішити ці проблеми можна за допомогою динамічної зміни ідентифікатора фрагмента (частини URL після #), що дозволяють багато браузерів.

Вміст, що динамічно завантажено, **недоступний пошуковим системам** (якщо не перевіряти запит, звичайний він або XMLHttpRequest). Пошукові машини не можуть виконувати JavaScript, тому розробники повинні поклопотатися про альтернативні способи доступу до змісту сайту. Старі методи обліку статистики сайтів стають неактуальними.

Багато сервісів статистики ведуть облік переглядів нових сторінок сайта. Для сайтів, сторінки яких широко використовують AJAX, така **статистика втрачає актуальність**.

Ускладнення проєкту. Перерозподіляється логіка оброблення даних – відбувається виділення і часткове перенесення на сторону клієнта процесів первинного форматування даних. Це ускладнює контроль цілісності форматів і типів. Кінцевий ефект технології може знівелювати необґрунтованим зростанням витрат на кодінг і управлінні проєктом, а також ризиком зниження доступності сервісу для кінцевих користувачів.

Потрібний включений JavaScript в браузері.

Альтернативами до технології є: Java-аплети, технологія JAVAFX, стек технологій Flash-ActionScript, Adobe Flex і Flash Remoting, Технологія Silverlight фірми Microsoft.

Як формат передачі даних зазвичай використовуються JSON або XML. Оскільки AJAX – це, по суті, лише всіляке вживання компонента XMLHttpRequest, то щоб зрозуміти роботу AJAX, потрібно розібратися з синтаксисом XMLHttpRequest. Зробимо це на прикладі.

```
<script type="text/javascript" language="JavaScript">
function doLoad() {
req=false;
try { // визначити метод підтримки
req=new ActiveXObject('Msxml2.XMLHTTP');
} catch (e) {
try {
req=new ActiveXObject('Microsoft.XMLHTTP'); // спрацює в Internet Explorer
} catch (e) {
if(window.XMLHttpRequest){ // спрацює в Mozilla і Safari
req=new XMLHttpRequest();
}
}
}
if (req){// якщо якийсь з варіантів підтримується
req.onreadystatechange = readystate; // призначимо обробник події об'єкта
req.open("GET", document.getElementById('edit1').value, true); // задати парамет-
ри методу open
req.send(null); // відправити запит
}
}
function readystate() {
if (req.readyState == 4){// якщо запит завершено
if (req.status == 200) { // якщо він завершений без помилок
document.getElementById('content').innerHTML
='<pre>'+req.responseText+'</pre>';
} else {
alert("Сталася помилка "+ req.status+":\n" + req.statusText);
}
}
}
</script>
<input type="TEXT" size="50" id="edit1">
<input type="button" value="Через ACTIVEX" onclick="doLoad()">
<div id=content style="white-space:pre"></div>
```

Розберемо приклад. Умова **window.XMLHttpRequest** визначає, яким чином працює об'єкт у даному браузері (можна додати перевірку, чи працює він взагалі і якщо **window.ActiveXObject** так само дає помилковий результат, даний метод взагалі не підтримується браузером). Потім створюється об'єкт один з підтримуваних типів, після чого події **onreadystatechange** (зміна стану) призначається обробник. Так само, призначаються параметри методу **open**. Перший з них визначає типа запиту: POST або **GET**, другий визначає адресу (в даному випадку адреса береться із значення текстового поля), третю, якщо достеменний, визначає що, запит повинен виконаються асинхронно, тобто, відправивши запит не чекати відповіді сервера, а продовжувати роботу, чекаючи відповіді у фоновому режимі.

Після всього цього методом **send** відбувається фактична відправка запиту і, як тільки від сервера приходить відповідь, спрацьовує обробник події onreadystatechange.

Під час обробки події onreadystatechange потрібно перевірити код стану об'єкта (властивість readyState). Визначено 4 стани: 0 – об'єкт не ініціалізував; 1 – йде завантаження; 2 – об'єкт вже завантажений; 3 – завантажений частково; 4 – завершення завантаження.

Як тільки код буде дорівнювати 4, дані сервера можна використовувати і виводити у браузер. Потрібно перевірити, що саме витягнули наші мережі, використовуючи код відповіді сервера, що зберігається в req.status. Якщо код дорівнює 200 (код помилок WEB-сервера), дані дійсно можна виводити, інакше, можна вивести повідомлення про помилку, що зберігається у властивості req.statusText. Замість стандартного тексту помилки, можна вивести свій, що зручно зробити, створивши масив визначень, ключі якого відповідатимуть номерам помилок: err[400]="Запит містить синтаксичну помилку"; err[401]="Для доступу потрібна авторизація"; err[403]="Доступ заборонений"і т. д.

Цим методом можна користуватися лише на ресурсах того самого сайта, з якого було закачано сторінку, що запрошено. Щоб користуватися іншими інтернет-ресурсами, доведеться використовувати серверний скрипт.

```
<?php
if (! @readfile ($_GET['file'])){
echo '<font color=red>Файл недоступний</font>';
}
?>
```

AJAX у цьому випадку виявляється складнішим, простіше використовувати IFRAME, але це не можливо використовувати безпосередньо усередині форм. У цьому випадку вживання AJAX виглядатиме так:

```
<script type="text/javascript" language="JavaScript">
function doLoad2 () {
req=false;
try { // визначити метод підтримки
req=new ActiveXObject('Msxml2.XMLHTTP');
} catch (e) {
try {
req=new ActiveXObject('Microsoft.XMLHTTP'); // спрацює в Internet Explorer
} catch (e) {
if(window.XMLHttpRequest){
// спрацює в Mozilla і Safari
req=new XMLHttpRequest();}
}
}
if (req){
// якщо якийсь з варіантів підтримується
req.onreadystatechange = readystate;
// призначимо обробник події об'єкта
req.open("GET", 'myscript.php?file='+document.
⊕getElementById('edit2').value, true);
// myscript.php – це наведений вище серверний скрипт
req.send(null);}
}
function readystate() {
if (req.readyState == 4){
// якщо запит завершений
if (req.status == 200) {
// якщо він завершений без помилок
document.getElementById('content').innerHTML
='<pre>'+req.responseText+'</pre>';
} else {
alert("Сталася помилка "+ req.status+":\n" + req.statusText);}
}
}
</script>
<input type="TEXT" size="50" id="edit2">
<input type="button" value="Через ACTIVEX і серверний скрипт"
onclick="doLoad2()">
<div id=content style="white-space:pre"></div>
```

Приклад таблиці заповнення списку користувачів в базі даних MYSQL

id	FirstName	LastName	Age	Hometown	Job
1	Peter	Griffin	41	Quahog	Brewery
2	Lois	Griffin	40	Newport	Piano Teacher
3	Joseph	Swanson	39	Quahog	Police Officer
4	Glenn	Quagmire	41	Quahog	Pilot

Коли користувач вибирає користувача у спливаючому списку вище, функція під назвою "ShowUser()" виконується. Функція запускається подією "Onchange".

```
<html>
<head>
<script type="text/javascript">
function showUser(str)
{
if (str=="")
{
document.getElementById("txtHint").innerHTML="";
return;
}
if (window.XMLHttpRequest)
{// code for IE7+, Firefox, Chrome, Opera, Safari
xmlhttp=new XMLHttpRequest();
}
else
{// code for IE6, IE5
xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
}
xmlhttp.onreadystatechange=function()
{
if (xmlhttp.readyState==4 && xmlhttp.status==200)
{
document.getElementById("txtHint").innerHTML=xmlhttp.responseText;
}
}
xmlhttp.open("GET","getuser.php?q="+str,true);
xmlhttp.send();
}
```

```
</script>
</head>
<body>

<form>
<select name="users" onchange="showUser(this.value)">
<option value="">Select a person:</option>
<option value="1">Peter Griffin</option>
<option value="2">Lois Griffin</option>
<option value="3">Glenn Quagmire</option>
<option value="4">Joseph Swanson</option>
</select>
</form>
<br />
<div id="txtHint"><b>Person info will be listed here.</b></div>
</body>
</html>
```

Функція **ShowUser()** виконує наступні дії.

Перевіряє, чи вибрана особа.

Створює об'єкт **XMLHttpRequest.**

Створює функцію, яка виконується, коли серверна відповідь готова.

Відсилає запит до файла на сервері.

Сторінка на сервері, до якої звертається JavaScript, файл PHP під назвою "Getuser.php". Вихідний код в "getuser.php" управляє запитом до бази даних MYSQL, повертає результат у таблиці HTML:

```
<?php
$q=$_GET["q"];

$con = mysql_connect('localhost', 'peter', 'abc123');
if (!$con)
{
die('Could not connect: ' . mysql_error());
}
mysql_select_db("ajax_demo", $con);
$sql="SELECT * FROM user WHERE id = '".$q."'";
$result = mysql_query($sql);
echo "<table border='1'>
<tr>
<th>Firstname</th>
<th>Lastname</th>
```

```
<th>Age</th>
<th>Hometown</th>
<th>Job</th>
</tr>";
while($row = mysql_fetch_array($result))
{
echo "<tr>";
echo "<td>" . $row['FirstName'] . "</td>";
echo "<td>" . $row['LastName'] . "</td>";
echo "<td>" . $row['Age'] . "</td>";
echo "<td>" . $row['Hometown'] . "</td>";
echo "<td>" . $row['Job'] . "</td>";
echo "</tr>";
}
echo "</table>";
mysql_close($con);
?>
```

Коли запит посилається з JavaScript до файла PHP, трапляються наступні події.

PHP відкриває під'єднування до сервера MYSQL.

Правильна особа знайдена.

Сторінка HTML створена, наповнена даними, і послана назад до замінника "TxtHint".

У результаті, при оновленні даних веб-сторінка не перезавантажується повністю, і веб-додатки стають швидше і зручніше.

По суті, технологія AJAX не може існувати без php скриптів, так як AJAX, тільки відправляє дані на сервер і приймає зворотний відповідь, водночас без перезавантаження сторінки. Тому правильніше задати питання, як зв'язати роботу AJAX і PHP. Але про це ми поговоримо далі (AJAX PHP приклад), а зараз розберемося зі специфікою роботи AJAX.

Для відправки даних на сервер, потрібно створити об'єкт XMLHTTPRequest. З його допомогою відкрити URL (PHP скрипт), послати на нього дані (POST або GET метод), отримати відповідь, та засобами знань мови js вивести отриманий відповідь сервера на монітор (відповіддю може бути будь-який фрагмент або елемент сторінки сайта).

Для прояснення подивіться нижче надану схему ілюструючу взаємодію AJAX з PHP.

Для прикладу взаємодії AJAX з PHP, створимо два файли, ajax_page.html, get_ajax.php.

Спочатку розглянемо користувацьку сторону програми, тобто ajax_page.html.

```
<html>
<head>
<title>Ajax PHP: приклад | sitear.ru</title>
<script Language="JavaScript">
function XmlHttp()
{
var xmlhttp;
try{xmlhttp = new ActiveXObject("Msxml2.XMLHTTP");}
catch(e)
{
 try {xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");}
 catch (E) {xmlhttp = false;}
}
if (!xmlhttp && typeof XMLHttpRequest!='undefined')
{
 xmlhttp = new XMLHttpRequest();
}
  return xmlhttp;
}

function ajax(param)
{
        if (window.XMLHttpRequest) req = new XmlHttp();
        method=(!param.method ? "POST" : param.method.toUpperCase());

        if(method=="GET")
        {
                send=null;
                param.url=param.url+"&ajax=true";
        }
        else
        {
                send="";
                for (var i in param.data) send+= i+"="+param.data[i]+"&";
                send=send+"ajax=true";
        }

        req.open(method, param.url, true);
        if(param.statbox)document.getElementByld(param.statbox).innerHTML =
'<img src="images/wait.gif">';
        req.setRequestHeader("Content-Type",          "application/x-www-form-
urlencoded");
```

```
            req.send(send);
            req.onreadystatechange = function()
            {
                        if (req.readyState == 4 && req.status = = 200) / / якщо відповідь позитивна
                        {
                                    if(param.success)param.success(req.responseText);
                        }
            }
}
</script>
</head>
<body>
<div id="status">
Тут будемо приймати звіти про роботу ajax додатка і відповідь сервера.
</div>
            <form action="get_ajax.php">
            <p><b>Поле введення 1</b></p>
            <p><textarea id="area_1" name="area_1" style="height:50px; width:500px; ">Введіть свій текст. Наприклад: Я люблю sitear.ru!</textarea></p>
            <p><b>Поле введення 2</b></p>
            <p><textarea id="area_2" name="area_1" style="height:100px; width:500px; ">Произвольный текст... Я в захваті від цієї статті, тому хочу підписатися на RSS, щоб читати такі статті частіше!!!</textarea></p>
            <input type='button' value='TEST AJAX' onclick='
                        ajax({
                                                url:"get_ajax.php",
                                                statbox:"status",
                                                method: "POST",
                                                data:
                                                {

first_area:document.getElementById("area_1").value,

second_area:document.getElementById("area_2").value
                                                },

success:function(data){document.getElementById("status").innerHTML=data;}
                                    Арсен Мірзоян-Зимовий пляж
            >
            </form>
</body>
</html>
```

Розберемо JavaScript сторону даного прикладу.

XmlHttp() – функція яка створює об'єкт XMLHttpRequest(), вона написана максимально компактно і кроссбраузерно.

AJAX (param) – наш обробник при виклику подій (onclick), приймає в масиві param необхідні дані.

URL – куди відсилати дані, причому він може бути в такому вигляді page.php?parameter=value, тобто інформація може передаватися за методом GET.

statbox – ID html блоку який буде приймати результати роботи AJAX PHP програми.

method – метод відправки даних, може бути POST або GET. У нашому прикладі ми використовуємо POST метод, але в той же час через url можна передавати інформацію GET методом.

data – масив переданих даних. У нашому прикладі, дані автоматично беруться з поля 1 і 2, хоча можна просто писати data: {name: "value"}.

success – ім'я функції або сама функція, яка буде обробляти отримані дані (текст).

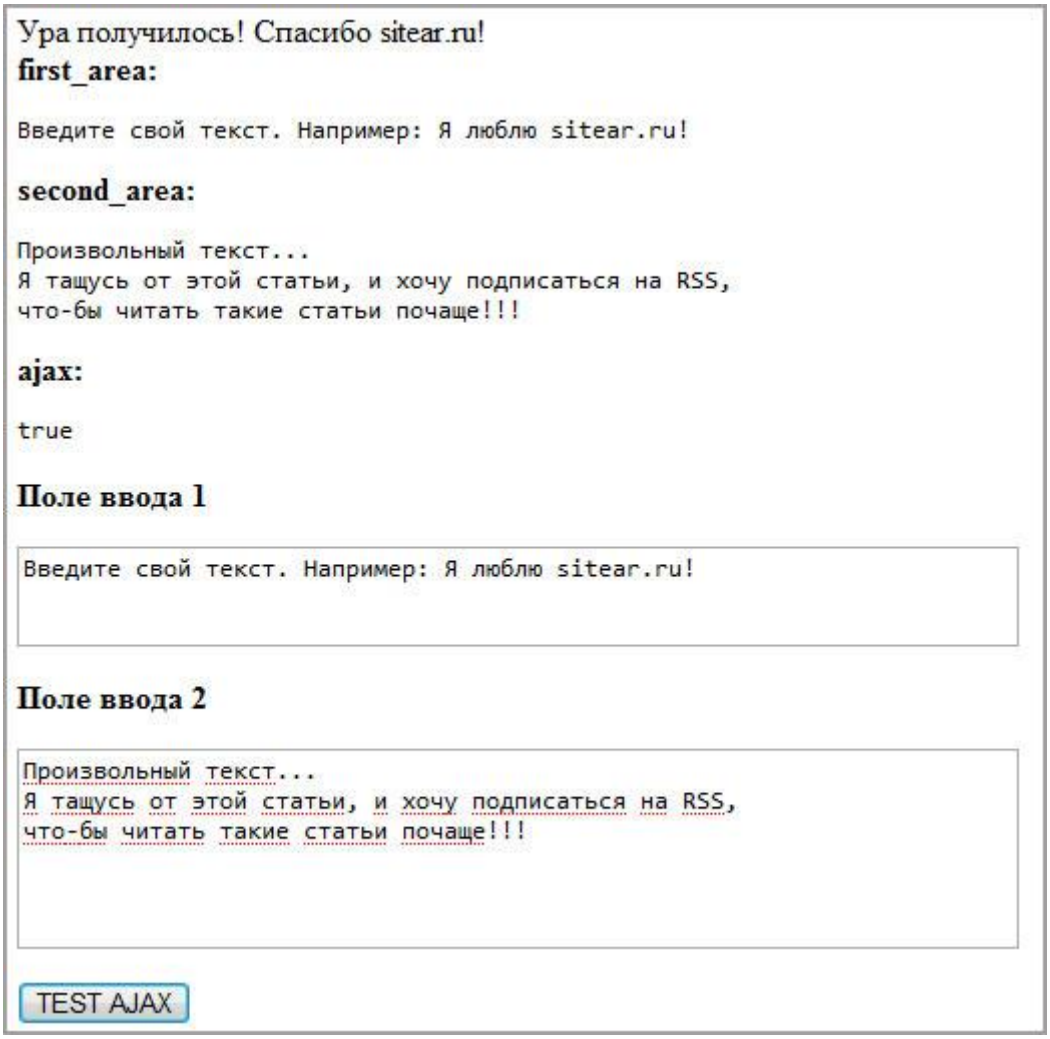
Виклик функції AJAX зроблений подією onclick=ajax().

Тепер розберемо серверну сторону-ajax+php додатки, тобто файл get_ajax.php.

```
<?php
header("Content-type: text/plain; charset=windows-1251");
header("Cache-Control: no-store, no-cache, must-revalidate");
header("Cache-Control: post-check=0, pre-check=0", false);
sleep(2);
echo "Ура вийшло! Я люблю sitear.ru!<br>";
while(list ($key, $val) = each ($_POST))
{
        $val = iconv("UTF-8","CP1251", $_POST[$key]);
        echo "<b>".$key.":</b> "."<pre>".stripslashes($val)."</pre>";
}
?>
```

Тут все набагато простіше. Спочатку встановлюємо кодування вихідних даних, за допомогою header. Встановлюємо заборону на кешування даних. sleep(2) – припиняє роботу скрипта на 2 секунди, це для того, щоб побачити.

Це відповідь, що отримано від файла get_ajax.php.



Коли запит посилається з JavaScript до файла PHP, наступні події трапляється.

PHP відкриває під'єднування до сервера MYSQL.

Правильна особа знайдена.

Сторінка HTML створена, наповнена даними, і послана назад до замінника "TxtHint".

У результаті, під час оновлення даних веб-сторінка не перезавантажується повністю, і WEB-додатки стають швидше і зручніше.

4.4. Використання фреймворків

Середовища розробки. Інтегроване середовище розробки, ICP (англ. Integrated development environment – IDE), також єдине середовище розробки, ECP – комплекс програмних засобів, який використовується програмістами для розроблення програмного забезпечення (ПЗ). Середовища розробки IDE більшою мірою визначають продуктивність праці членів проєктних команд.

Краща безкоштовна IDE-*NetBeans*. Відкриває рейтинг IDE NetBeans. По-перше, вона підтримує російську мову, що для багатьох PHP-розробників життєво важливо. По-друге, NetBeans підтримує всі відомі фреймворки, у тому числі Zend, Laravel, Doctrine, CakePHP, Smarty, Yii і Symfony2. Крім того, природно підтримуються javascript, HTML, CSS і PHP в найсвіжіших версіях.

Краща платна IDE-*PHPStorm*. Дана IDE ідеально підійде для повноцінної full-stack WEB-розробки. Також підтримує фреймворки Symfony, Zend, Yii, CakePHP, основні CMS системи Magento, Drupal, Wordpress, налагоджувальники Zend Debugger і Xdebug, а для фронтенда вам доступні CSS3, HTML5, Sass, CoffeeScript, TypeScript, Stylus, Less і ще кілька. Програмне середовище поставляється безкоштовно для вчителів і студентів профільних ЗВО.

Найперспективніша IDE – *Zend Studio*. Потужний продукт від Zend Technologies.

Налагодження за допомогою Zend Debugger, Xdebug, інтеграція з Z-Ray.

Підтримка SVN, CVS, Docker, Git Flow.

Можливість перегляду MySQL, SQLite, PostgreSQL та інших СУБД.

Підтримка бібліотеки Dojo.

Якісний редактор коду, з функціями аналізу, виправлення, рефакторінгу, індикації, генерації.

Підтримка хмарних сервісів Microsoft Azure і Amazon AWS.

Вільне поводження з JavaScript, CSS, HTML і звичайно PHP (4 і 5).

Панель розробника Zend Studio Toolbar для браузерів IE і Mozilla Firefox.2.1.5.1. Бібліотеки, фреймворки та систем керування контентом, використання фреймворків.

Істотно скорочує терміни розробки.

Дозволяє писати добре структурований, добре документований і повторно використовуваний код.

Дозволяє створювати масштабовані, легко розширювані програми.

Приховує від розробника необхідність піклуватися про низькорівневу безпеку сайту.

Стимулює слідувати шаблону проєктування MVC (Модель – Представлення – Контролер), що дозволяє розділити логіку програми і подання даних.

Сприяє застосуванню сучасних методів програмування, насамперед об'єктно-орієнтованого.

Embarcadero *HTML5 Builder*. Дає можливість створювати традиційні WEB або мобільні додатки на основі єдиного вихідного коду з використанням стандартних WEB-технологій, а потім запускати їх на різних мобільних платформах (WEB, iOS, Android, Blackberry, Windows Phone), тим самим різко скорочуючи час розроблення.

Бібліотеки, фреймворки та систем керування контентом.

Використання фреймворків:

істотно скорочує терміни розроблення;

дозволяє писати добре структурований, добре документований і повторно використовуваний код;

дозволяє створювати масштабовані, легко розширювані програми;

приховує від розробника необхідність піклуватися про низькорівневу безпеку сайта;

стимулює використання шаблона проєктування MVC (Модель – Представлення – Контролер), що дозволяє розділити логіку програми і подання даних;

сприяє застосуванню сучасних методів програмування, насамперед об'єктно-орієнтованого.

Наведемо характеристики найбільш популярних фреймворків для PHP.

Laravel. Незважаючи на свою молодість (перший реліз вийшов у 2011 році), це вже зовсім зрілий продукт, і, згідно з опитуванням, проведеним порталом SitePoint, він займає перше місце за популярністю серед розробників PHP.

Зараз Laravel – це величезна екосистема, що вміщує хостинг і платформу для розгортання додатків. Він має власний обробник шаблонів "Blade", елегантний синтаксис, що спрощує виконання рутинних операцій, таких, як авторизація, управління сесіями, чергами, кешуванням і маршрутизацією. Крім того, Laravel містить локальне середовище розробки Homestead, що є частиною пакета Vagrant.

Величезним плюсом є те, що існує російськомовне співтовариство, де перевели майже всю технічну документацію.

Symfony. Компоненти фреймворку Symfony 2 використовують такі відомі проєкти, як Drupal і phpBB, і розглянутий вище Laravel. Над Symfony працює велика спільнота розробників і він має величезну армію прихильників.

Symfony Components – це набір PHP бібліотек, здатних задовольнити найрізноманітніші потреби розробника, будь то створення форм, маршрутизація, авторизація, розробка шаблонів і багато іншого. На сайті розробників є портфоліо проєктів, виконаних за допомогою цього фреймворку.

CodeIgniter. Легковагий фреймворк із давньою історією (перший реліз вийшов у 2006 році). Традиційною особливістю є виключно легкий швидкий процес установки, і практично повна відсутність потреби в конфігурації. Це ідеальний вибір, якщо потрібно уникнути конфліктів з версіями, оскільки працює практично на всіх наявних платформах (зараз вимагає тільки PHP 5.2.4).

CodeIgniter не зовсім підтримує парадигми MVC – якщо рівень контролер є обов'язковим, то рівні моделі і уявлення опціональні. Розробник може використовувати власні правила кодування та угоди про імена, що, безсумнівно, надає йому велику свободу. Ядро фреймворку має незначний обсяг (близько 2 Мб), але функціональність можна розширити за рахунок плагінів від інших розробників.

Yii 2 активно використовує концепцію "ледачого" (або "відкладеного") завантаження, що робить його одним із найбільш швидких PHP фреймворків. Використовує об'єктно-орієнтований підхід та концепцію DRY (Don't

Repeat Yourself – Не повторюйте) і дозволяє створювати ясний і легко читається код.

Yii 2 тісно інтегрований з jQuery, містить набір AJAX-функції і вбудований механізм "шкурок" і тим, що ідеально підходить для програмістів і фронтенд розробників. Крім того, у складі Yii 2 є такий потужний засіб, як генератор коду Gii, що дозволяє полегшити рутинні операції під час розробки проєкту.

Phalcon фреймворк вперше з'явився в 2012 році і швидко набув популярності серед розробників. Він досягає високої швидкодії за рахунок того, що написаний на C/C++, що і знайшло відображення в його назві (Phalcon співзвучно англ. falcon – сокіл). Незважаючи на це, вся функціональність реалізована у вигляді PHP класів.

Phalcon досить добре оптимізований на рівні ядра, що значно підвищує продуктивність і знижує навантаження порівняно з типовими MVC додатками, а його базова функціональність доповнюється безліччю корисних компонентів, таких як універсальний автозавантажувач, менеджер ресурсів, механізм кешування та локалізації і багато інших. Крім того Phalcon забезпечений прекрасною документацією, так що він належить до тієї категорії продуктів, які, безсумнівно, варто спробувати.

CakePHP. Історія розвитку CakePHP налічує вже 10 років (перший реліз вийшов у 2005 році), але він досі залишається дуже популярним, оскільки активно розвивається і йде в ногу із часом. Остання версія цього фреймворку, CakePHP 3.0, містить перероблений менеджер сесій, поліпшену, за рахунок поділу деяких компонентів, модульність, і можливість створення самостійних бібліотек.

На домашній сторінці проєкту представлено значне портфоліо цього фреймворку – з його допомогою створені сайти таких великих корпорацій, як BMW, Hyundai і Express. Це відмінний інструмент для розроблення додатків, на перше місце яких ставиться безпека. Перевірка даних, що вводяться, захист від впровадження SQL коду, міжсайтового скриптинга (XSS), міжсайтової підробки запитів (CSRF) – все це присутнє в CakePHP.

Zend Framework – це потужний і стабільний PHP фреймворк, що володіє багатими можливостями настройки, тому він, як правило, не рекомендується для невеликих проєктів. Партнерами Zend є такі гранди комп'ютерної індустрії, як IBM, Microsoft, Google і Adobe. Прийдешній реліз Zend Framework під номером 3 буде оптимізований для PHP 7, проте збереже підтримку PHP 5.5.

Поточний реліз Zend Framework 2 має безліч функцій, таких, як інструменти для шифрування, зручний редактор, що підтримує drug and drop і front-end технології (HTML, CSS, JavaScript), повноцінний online дебагер, модулі для тестування і підключення до баз даних. Zend Framework створювався з урахуванням методології розроблення Agile і призначений для розроблення високоякісних додатків корпоративного рівня.

Тема 5. Динамічна графіка на WEB-сторінках

5.1. Графічні елементи HTML5

Canvas (холст) – елемент HTML5 для створення растрового двомірного зображення. Зазвичай використовується спільно з JavaScript.

Ширину і висоту Canvas можна змінювати.

Використовується в основному для відтворення графіків та ігрових елементів у браузерних іграх, вставки відео, створення повноцінного плеєра.

Canvas так само використовують у WebGL для апаратного прискорення 3D-графіки. У результаті можна створювати навіть 3D-ігри, що працюють у вікні браузера.

Створення Canvas.

```
<!doctype html>
<html>
  <head>
    <title>Canvas</title>
    <meta charset='utf-8' />
  </head>
  <body>
    <canvas id='test' height='320' width='480'>Текст показується, якщо елемент не підтримується</canvas>
    <script type="text/javascript">
      var canvas = document.getElementById("test");
      var ctx = canvas.getContext('2d');
      /* Малює контур прямокутника на всю ширину і висоту Canvas */
      ctx.strokeRect(0, 0, canvas.width, canvas.height);
    </script>
  </body>
</html>
```

Створюється Canvas шляхом звичайної вставки тега <canvas> у html-код. Далі через звичайний JavaScript ми отримуємо елемент і вміст Canvas (рядок 10, 11) і малюємо обведений прямокутник, який розтягується на всю ширину і висоту Canvas.

Формат SVG.

SVG (Scalable Vector Graphic) існує з 1999 р., а з 16 серпня 2011 р. включена в рекомендації W3C. SVG дуже недооцінена WEB-розробниками, хоча має кілька важливих переваг.

Масштабування: на відміну від растрової графіки, SVG не втрачає якості під час масштабування, тому її зручно використовувати для розробки під retina.

Зменшення HTTP-запитів: під час використання SVG скорочується кількість звернень до сервера, відповідно збільшується швидкість завантаження сайта.

Стайлинг і скриптинг: за допомогою CSS можна змінювати параметри графіки на сайті, наприклад фон, прозорість або межі.

Анімація та редагування: за допомогою JavaScript можна анімувати SVG, а також редагувати в текстовому або графічному редакторі (InkScape або Adobe Illustrator).

Малий розмір: об'єкти SVG важать набагато менше растрових зображень.

Використання SVG на WEB-сторінках.

Згідно з офіційною специфікацію можна малювати прості об'єкти з допомогою SVG: прямокутник, коло, лінію, еліпс, ламану лінію або багатокутник за допомогою тега svg.

Проста лінія за допомогою тега line зі всього двома параметрами – точками початку (x1 і x2) і кінця (y1 і y2):

```
<svg>
<line x1="0" y1="0" x2="200" y2="200" stroke-width="1" stroke=rgb(0,0,0)"/>
</svg>
```

Також можна додати атрибути або стилі stroke and stroke-width, щоб задати колір і ширину:

```
style="stroke-width:1; stroke:rgb(0,0,0);"
```

Ламана лінія. Синтаксис аналогічний попередньому, використовується тег <polyline>, атрибут points задає точки:

```
<svg>
<polyline points="0,0 50,0 150,100 250,100 300,150" fill=rgb(249,249,249)" stroke-width="1" stroke=rgb(0,0,0)"/>
</svg>
```

Прямокутник. Викликається тегом <rect>, можна додати деякі атрибути:

```
<svg>
<rect width="200" height="200" fill=rgb(234,234,234)" stroke-width="1" stroke=rgb(0,0,0)"/>
</svg>
```

Окружність. Викликається тегом <circle>, в прикладі за допомогою атрибута r задаємо радіус, cx і cy задають координати центру:

```
<svg>
<circle cx="102" cy="102" r="100" fill=rgb(234,234,234)" stroke-width="1" stroke=rgb(0,0,0)"/>
</svg>
```

Еліпс. Викликається тегом <ellipse>, працює аналогічно <circle>, але можна задати два радіуса – rx і ry:

```
<svg>
<ellipse cx="100" cy="50" rx="100" ry="50" fill=rgb(234,234,234)" stroke-width="1" stroke=rgb(0,0,0)"/>
</svg>
```

Багатокутник. Викликається тегом <polygon>, багатокутник може мати різну кількість сторін:

```
<svg>
<polygon points="70.444,218.89 15.444,118.89 70.444,18.89 180.444,18.89 235.444,118.89 180.444,218.89" fill=rgb(234,234,234)" stroke-width="1" stroke=rgb(0,0,0)"/>
</svg>
```

Анімація SVG здійснюється за допомогою мови SMIL. Ця мова була рекомендована WC3 консорціумом як основна для опису сценаріїв векторної XML графіки. SMIL не є мовою програмування, це всього лише мова розмітки, причому заснована на тому самому XML. Усього одним тегом <animate> можна описати складний сценарій для окремих фігур (переміщення, трансформація, візуалізація).

Найпростіший приклад SVG анімації – жовтий прямокутник, який повільно переходить з прозорого стану в непрозорий.

```
<?xml version="1.0" standalone="no"?>
<svg width="12cm" height="4cm" viewBox="0 0 1200 400" xmlns="http://www.w3.org/2000/svg" version="1.1">
<rect x="400" y="100" width="400" height="200" fill="yellow">
  <animate attributeName="opacity" attributeType="CSS" from="0" to="1" begin="0s" dur="10s" repeatCount="1" />
</rect>
/>
```

Керування елементами SVG за допомогою JavaScript.

Малювання SVG не таке вже й складне завдання, бо в мережі є безліч JavaScript-бібліотек для роботи з SVG-зображеннями. Ці JS-бібліотеки допомагають дизайнерам і розробникам створювати для своїх проєктів і WEB-додатків інноваційну і візуально красиву графіку.

Опишемо бібліотеки JavaScript для роботи з SVG, які дозволяють створювати привабливий графічний і анімаційний контент.

Textures.js дозволяє легко додавати шаблони SVG для поліпшення візуалізації даних. У бібліотеці є величезна кількість різноманітних текстур, в тому числі ліній, кіл, контурів і навіть власні візерунки.

Circulus.svg – це генератор кругового меню, який дозволяє створювати SVG-меню. Він пропонує різни стилі: повне коло або півколо, а також інші.

deSVG може легко витягти вбудований SVG з HTML. Це дозволяє задавати стилі SVG за допомогою CSS, і забезпечити доступ до SVG навіть без JavaScript.

SVG Morpheus – це бібліотека JavaScript, яка дозволяє створювати SVG-іконки, які трансформуються один в одного. Бібліотека проста у використанні і працює з використанням Delightful Details переходів від Material Design.

Vivus – це JavaScript-клас для анімування SVG без необхідності введення залежностей. Він дозволяє створити асинхронну анімацію, встановити затримку і покрокове виконання анімації.

Walkway.js – це простий спосіб анімувати нескладні елементи SVG. Бібліотека складається з простих вбудованих функції, а також містить опції для вибору й установки тривалості анімації.

ZorroSVG дозволяє легко додавати маски для SVG-зображень. Завдяки йому можна створювати зображення з прозорістю як при використанні формату PNG, але водночас зберігаючи невеликі розміри файлів.

Raphael – це JavaScript бібліотека, яка полегшує створення і використання на сайті векторної графіки. Для створення графіки вона використовує SVG і VML. Графіка генерується в якості DOM-об'єкта, і нею можна керувати через JavaScript.

Snap.SVG – це повнофункціональна безкоштовна JavaScript-бібліотека, яка полегшує роботу з SVG в сучасних браузерах. Snap.SVG створена розробниками найпопулярнішої бібліотеки SVG Raphael. Вона підтримує такі функції, як маски, візерунки, повні градієнти, групи і багато іншого.

D3.JS – це бібліотека JavaScript для роботи з документами, що базуються на даних. D3 допомагає "оживити" дані, використовуючи HTML, SVG і CSS. А також дає можливість повністю задіяти сучасні браузери без обмежень, пов'язаних із використанням фреймворків. Бібліотека поєднує в собі потужні компоненти візуалізації й орієнтований на даних підхід у роботі з DOM.

Paths.js – це JavaScript бібліотека для створення SVG-контурів, які потім можуть бути використані разом із движками шаблонів, такими, як Mustache або Handlebars, для відображення цих SVG фігур у браузері. Paths пропонує для роботи три різних API.

SVG.JS – це JavaScript бібліотека для простої роботи (управління та анімування) з SVG. Бібліотека є автономною, дуже компактною (розмір пакета всього 5 кілобайт) і має безліч корисних функцій. Вона містить вбудовані методи для створення форм (прямокутник, коло, багатокутник тощо) або визначення зображень. Для всіх елементів може бути задана анімація (зміна розміру, положення, кольору або будь-яких інших властивостей) і взаємодія зі стандартними подіями JavaScript.

Jim Knopf – це бібліотека JavaScript для створення перемикачів на основі SVG. Бібліотека не вимагає встановлення будь-яких JS-фреймворків і поставляється з вбудованими типами перемикачів. Створені елементи

управління можна повністю масштабувати (завдяки SVG), їхній дизайн можна налаштовувати за допомогою CSS.

Крім того, вони можуть використовувати курсор миші, колесо прокрутки, клавіатуру або тачпад. Існують опції для встановлення мінімальних / максимальних значень, початкового положення і допустимого кута повороту перемикача.

Seen.js виводить 3D-сцени на SVG або HTML5-полотнах. Seen.js містить прості абстракції для використання SVG і HTML5 на межі їх графічних можливостей. Усі інші компоненти цієї бібліотеки незалежні від типу вмісту, що виводиться на полотні.

BonsaiJS – це компактна бібліотека з інтуїтивним графічним API і візуалізацією SVG. Її основні функції містять архітектурно розділені "майстри" запуску і візуалізації; Iframe, контексти запуску Worker та Node, форми, контури, елементи (відео, зображення, шрифти), ключові кадри і стандартну анімацію, морфінг фігур / контурів і багато іншого.

5.2. Створення зображень на сервері

Бібліотеки графічних функцій.

Одна з найпопулярніших – це *GD library.* GD library дозволяє створювати зображення, видозмінювати і управляти ними без особливих зусиль.

Використання в PHP графічної бібліотеки GD library.

Потрібно переконатися в тому, що GD – бібліотека встановлена і активована на вашому сервері. Потрібно створити PHP-файл на сервері:

```
<?php echo phpinfo(); ?>
```

Тепер на сторінці браузера можливо побачити величезний список можливостей встановленої версії PHP. Якщо знайти розділ 'GD' і переконатися, що бібліотека підключена. На більшості серверів, GD-бібліотека встановлена та підключена.

Спочатку, потрібно зробити так, щоб всі звіти про помилки видавалися браузером. Далі потрібно ввести php-код:

```
<?php
 ini_set("display_errors", "1");
 error_reporting (E_ALL);
 ?>
```

Найпростіше намалювати за допомогою PHP синій квадрат, щоб розібратися з основами. Щоб отримати бажаний квадрат, необхідно виконати наступні дії:

встановити тип вмісту як "зображення", щоб браузер зміг інтерпретувати зображення належним чином;

створити нове порожнє зображення, встановивши потрібну ширину і висоту;

зробити колір фону синім;

зберегти остаточний варіант зображення і передати його в браузер;

очистити пам'ять, яка використовувалася для створення і зберігання зображення;

організувати відображення малюнка з файла index.php.

Кодування потрібно зберегти під ім'ям 'basic_square.php'.

```
//Встановлюємо відображення повідомлень про помилки
 ini_set ("display_errors", "1");
 error_reporting (E_ALL);

 //Встановлюємо тип вмісту
 header('content-type: image/png');

 //Визначаємо розміри зображення
 // 125px width, 125px height
 $image = imagecreate(125, 125);

 //Обираємо колір фону
 $blue = imagecolorallocate($image, 0, 0, 255);

 //Встановлюємо ще один колір – просто щоб переконатися, що під час відображення малюнку фоновим буде саме колір, встановлений першим ()
 //Зверніть увагу – квадрат буде синього, а не червоного кольору.
 $red = imagecolorallocate($image, 255, 0, 0);

 //Зберігаємо файл у форматі png і виводимо його
 imagepng($image);

 //Чистимо використану пам'ять
 imagedestroy($image);
 ?>
```

Додаємо опцію повідомлення про помилки, щоб за потреби швидко їх виправити.

Використовуємо функцію header(), щоб встановити тип вмісту, png.

Зберігаємо зображення і встановлюємо необхідну ширину і висоту.

Використовуємо функцію imagecolorallocate, щоб вибрати синій колір для фону нашого малюнка. Видно колір, встановлений в першу чергу, тобто квадрат.

Для остаточного збереження малюнка використовується imagepng(). Якщо потрібно зберегти малюнок в будь-якому каталозі, слід ввести додаткові параметри.

Почистимо пам'ять за допомогою функції **imagedestroy().**

Подивиться на намальований квадрат, відкривши файл blue_square.php, чи встановіть посилання на нього на головній сторінці.

```
<img src='http://themeforest.s3.amazonaws.com/65_gd/basic_square.php' alt=" />
```

Якщо все було зроблено правильно, можна побачити синій квадрат.

Щоб зробити зображення з текстом, написаним обраним нами шрифтом, можливо скористатися будь-яким шрифтом із каталогу TrueType Font (.tff). У наступних прикладах потрібно використовувати шрифт 'Advent', щоб написати "Hello, World" на темно-сірому квадраті.

```
<?php
 //Встановлюємо повідомлення про помилки
 ini_set("display_errors", "1");
 error_reporting (E_ALL);


 //встановлюємо тип вмісту
 header('content-type: image/png');

 //Визначаємо розмір зображення – 300x300 пікселів
 $image = imagecreate(300, 300);

 //Встановлюємо фоновий темно-сірий колір
 $dark_grey = imagecolorallocate($image, 102, 102, 102);
 $white = imagecolorallocate($image, 255, 255, 255);
```

```
//Вказуємо шлях до шрифту
$ font_path = 'advent_light';

//Пишемо текст
$string = 'Hello, World!';

//З'єднуємо текст і картинку
imagettftext($image, 50, 0, 10, 160, $white, $font_path, $string);

//Зберігаємо зображення
imagepng($image);

//Чистимо пам'ять
imagedestroy($image);
?>
```

Зберегти код у файлі 'hello_world.php' і відкривати його таким самим чином, який було описано вище.

Встановити розмір зображення 300 х 300 пікселів і використовували темно-сірий фон.

Вказати шлях до шрифту, використовуючи ім'я шрифту: справа в тому, щоб накласти текст на зображення.

Ім'я шрифту пишеться без розширення. Це залежить від того, яку версію GD-бібліотеки використовує PHP: якщо назвою шрифту не починається з '/', то буде потрібно розширення ttf – тобто, якщо перед назвою файла шрифту не варто, то в кінці буде автоматично додано ttf.

Пишемо "Hello, World!" і зберігаємо текст для подальшого використання.

Найважливіша функція, використана в цьому коді – це **imagettftext (),** що вимагає зазначення 8 параметрів: шлях до зображення, розмір зображення, кут нахилу, вісь х, вісь у, колір, шрифт, текст (в такому порядку).

Зберегти код, відкрити його будь-яким способом, і у браузері з'явиться картинка.

Функція imagettftext, що з'єднує текст і картинку, має параметр, що дозволяє розташувати текст під кутом.

Перепишемо, щоб повернути рядок на квадраті.

```
<?php
 //Встановлюємо повідомлення про помилки
 ini_set("display_errors", "1");
 error_reporting (E_ALL);

 //Встановлюємо тип вмісту
 header('content-type: image/png');

 //Визначаємо розмір зображення: 300x300 пікселів
 $image = imagecreate(300, 300);

 //Визначаємо колір фону – темно-сірий
 $dark_grey = imagecolorallocate($image, 102, 102, 102);
 $white = imagecolorallocate($image, 255, 255, 255);

 //Вказуємо шлях до шрифту
 $ font_path = 'advent_light';

 //Пишемо текст
 $string = 'Hello, World!';

 //З'єднуємо текст і картинку
 imagettftext($image, 50, -45, 30, 70, $white, $font_path, $string);

 //Зберігаємо зображення
 imagepng($image);

 //Чистимо пам'ять
 imagedestroy($image);
 ?>
```

Встановлено значення −45 градусів замість 0 градусів (як у попередньому прикладі), щоб повернути текст.

Можливо не тільки малювати картинки, але і використовувати готові. Зробимо календар, що відображає поточну дату (місяць, день і рік). Потрібно пошукати в Google заготовку для календаря, зображення у форматі .svg – потрібно змінити його розмір і перевести у формат png.

Обробити заготовку за допомогою функцій GD і PHP.код:

```
<?php
  //Встановлюємо повідомлення про помилки
  ini_set("display_errors", "1");
  error_reporting (E_ALL);

  //Встановлюємо тип вмісту
  header('content-type: image/jpeg');

  //Зберігаємо дату
  list($month, $day, $year) = explode('/', date('F/jS/Y'));

  //Завантажуємо фонове зображення
  $image = imagecreatefrompng('calendar_blank.png');
  $image_width = imagesx($image);

  //Визначаємо колір фону і шрифт
  $white = imagecolorallocate($image, 255, 255, 255);
  $black = imagecolorallocate($image, 0, 0, 0);
  $ font_path = 'advent_light';

  //Вказуємо позиції тексту
  $pos_month = imagettfbbox(13, 0, $font_path, $month);
  $pos_day = imagettfbbox(25, 0, $font_path, $day);
  $pos_year = imagettfbbox(8, 0, $font_path, $year);

  // Пишемо місяць
  imagettftext($image, 13, 0, ($image_width - $pos_month[2]) / 2, 40, $white,
$font_path, $month);

  //Пишемо день
  imagettftext($image, 25, 0, ($image_width - $pos_day[2]) / 2, 80, $black,
$font_path, $day);

  // Пишемо рік
  imagettftext($image, 8, 0, ($image_width - $pos_year[2]) / 2, 100, $black,
$font_path, $year);

  //Зберігаємо зображення
  imagejpeg($image, '', 100);

  //Чистимо пам'ять
  imagedestroy($image);
  ?>
```

Слід встановити тип вмісту, щоб правильно відобразити картинку.

Потрібно визначити формат дати за допомогою функції explode () і зберегти його. Таким чином можна впорядковувати будь-яку інформацію. Так відкрито доступ до відомостей про поточний місяць, день і рік.

Щоб організувати формат дати, слід скористатися функцією imagettfbbox(). Ця функція забезпечує відображення дати в заданому порядку. Для доступу до правого нижнього кута написати $pos_month. Це потрібно для того, щоб текст автоматично розташовувався правильно, не залежно від його розміру.

Відображення 3D в браузері, використовуючи технологію WebGL.

Для досягнення мети розглянемо завдання побудови декількох ліній у тривимірному просторі.

Хід роботи:

1. Отримати WebGL контекст із Canvas:

2. Завантажити програму шейдерів. А саме:

створити програму шейдерів;

отримати вихідний код окремо для вершинного і фрагментного шейдерів;

компілювати коди шейдерів;

приєднати до програми;

активувати програму.

3. Встановити дві матриці: model-view і projection.

4. Розмістити, заповнити, активувати буфери даних вершин.

5. Малювати.

WebGL контекст можливо отримати з Dom-елемента Canvas, викликавши метод getContext ("webgl"). Слід зауважити, що Khronos Group рекомендує (https://www.khronos.org/webgl/wiki/FAQ) для отримання контексту WebGL використовувати такий спосіб:

```
var names = ["webgl", "webgl", "webkit-3d", "moz-webgl"];
   gl = null;
   for (var ii = 0; ii < names.length; ++ii) {
      try {
         gl = canvas.getContext(names[ii]);
      } catch(e) {}
      if (gl) {
         break;
      }
   }
```

У разі успішного отримання контексту об'єкт gl має методи, назви яких дуже схожі на функції OpenGL ES. Наприклад, функція clear

(COLOR_BUFFER_BIT) для WebGL буде gl.clear(gl.COLOR_BUFFER_BIT),

що дуже зручно. Але слід пам'ятати, що не всі функції WebGL мають такий самий синтаксис, як і функції OpenGL ES 2.0.

Шейдерна програма є невід'ємною частиною побудови зображень за допомогою WebGL. Саме через неї задається положення і колір кожної вершини наших ліній. У завданні ми використовуємо два шейдери: вершинний і фрагментний. Під час побудови ліній у тривимірному просторі вершинний шейдер відповідає за положення вершин у просторі, ґрунтуючись на значеннях видової матриці й матриці перспективної проєкції. Фрагментний шейдер використовують для обчислення кольору ліній.

Вершинний рейдер.

```
attribute vec3 aVertexPosition;
attribute vec4 aVertexColor;
uniform mat4 mvMatrix;
uniform mat4 prMatrix;
varying vec4 color;

void main(void)
{
        gl_Position = prMatrix * mvMatrix * vec4 ( aVertexPosition, 1.0 );
        color = aVertexColor;
}
```

Фрагментний рейдер.

```
 #ifdef GL_ES
   precision highp float;
#endif
varying vec4 color;
void main(void)
{
    gl_FragColor = color;
}
```

Те, що визначено після "uniform", є загальним для всіх вершин. У цьому випадку це матриці перетворення: видова і перспективна. Те, що визначено після "attribute", використовується під час обчислення кожної вершини. Наразі це положення вершини та її колір. Після "varying" визначаємо змінну, яка буде передана з вершинного у фрагментний шейдер. Результат обчислення положення привласнюємо змінної gl_Position, кольору-gl_FragColor.

Модельно-видова матриця і матриця перспективної проєкції матриці мають розмір 4 × 4 і використовуються для розрахунку відображення тривимірних об'єктів на двовимірну площину. Їх відмінність полягає в тому, що видова матриця визначає, як об'єкти будуть виглядати для спостерігача, наприклад, у разі зміни його положення, а матриця проєкції спочатку задає спосіб проєктування.

У програмі значення матриці проєкції задаються в разі виклику функції gluPerspective на етапі ініціалізації, надалі ця матриця не змінює своїх значень. Функція gluPerspective не є стандартною, її ми визначили самі. Її аргументи: **fovy, aspect, zNear, zFar.** Аргумент **fovy** – область кута перегляду по вертикалі в градусах; **aspect** – відношення ширини області перегляду до висоти; **zNear** – відстань до ближньої площини відсікання (все що ближче – не малюється); **zFar** – відстань до далекої площини відсікання (все що далі – не малюється).

Для завдання значень модельно-видової матриці можна використовувати кілька підходів. Наприклад, створити і використовувати функцію **gluLookAt (camX, camY, camZ, tarX, tarY, tarZ, upX, upY, upZ)** – аналог функції OpenGL, яка приймає в якості аргументів координати положення камери, координати цілі камери та up-вектор камери. Інший спосіб – це створення і використання функцій glTranslate, glRotate, glScale, які виробляють зрушення, обертання, масштабування відносно спостерігача (камери). Для первинного визначення положення камери можна використовувати **gluLookAt,** а для наступних перетворень – **glTranslate, glRotate, glScale.** Так чи інакше, ці функції лише змінюють значення однієї і тієї самої модельно-видової матриці. Для зручності обчислення матриць можна використовувати бібліотеку **sylvester.js.**

Далі розглянемо спосіб змінювати значення обох матриць. А саме: їх передачу програмі шейдерів. У вершинному шейдері для модельно-видової матриці використовують змінну "mvMatrix". Щоб передати цю змінну

значення матриці, потрібно спочатку отримати її індекс у програмі. Для цього використовуємо функцію

loc=gl.getUniformLocation (shaderProgram, name),

яка є стандартною.

Перший аргумент – змінна, яка вказує на програму шейдерів, що отримана на другому етапі, а аргумент "name" – ім'я змінної, якою ми хочемо передати значення, тут **name= "mvMatrix".** Тепер, отримавши індекс, використовуємо функцію **gl.uniformMatrix4fv (loc, false, new Float32Array(mat.flatten()))** для передачі значення матриці mat. Аналогічно, отримуємо індекс і встановлюємо значення для матриці проєкції. Слід пам'ятати, що видову матрицю в шейдерної програмі потрібно оновлювати всякий раз під час зміни її значень, щоб вони вступили в силу.

Використання буферів у WebGL обов'язково. Положення кожної точки і її колір будемо зберігати в двох буферах. Розглянемо фрагмент коду, що виконує всю роботу для буфера, що зберігає координати точок, між якими будемо малювати прямі.

```
/* Створюємо буфер  */
vPosBuffer = gl.createBuffer();
   /* Активуємо буфер*/
gl.bindBuffer(gl.ARRAY_BUFFER, vPosBuffer);
/* Копіюємо в буфер координати вершин */
gl.bufferData(gl.ARRAY_BUFFER, new Float32Array(verticies), gl.DYNAMIC_DRAW);
  /* Визначаємо, що координати вершин мають певний індекс атрибута і містять 3 floats на вершину */
gl.vertexAttribPointer(vPosLoc, 3, gl.FLOAT, false, 0, 0);
/ * Задіємо індекс атрибута */
gl.enableVertexAttribArray(vPosLoc);
```

Тут **verticies** – масив координат точок ліній. Координати йдуть по шість штук, перші 3 з яких – x-, y-, z-координата початку лінії, такі, відповідно, кінця. vPosLoc- це Індекс атрибута **"aVertexPosition"** у шейдерній програмі. Оскільки в програмі були явно задані індекси за допомогою **gl.bindAttribLocation (shaderProgram, loc, shadersAttrib)** на етапі складання шейдерної програми, то отримувати їх ще раз не потрібно. Якби такого не було, то слід отримати індекс, використовуючи команду **"vPosLoc = getAttribLocation(shaderProgram, "aVertexPosition")".** Аналогічні дії проводяться і з другим буфером, відрізнятися буде даними

(замість verticies масив квітів) і індексом у шейдерній програмі (замість **vPosLoc**).

Очищення буфера кольору або, простіше кажучи, завдання фону робиться стандартними командами.

```
gl.clearColor(0, 0, 0, 1);
gl.clear(gl.COLOR_BUFFER_BIT);
/*Виконується малювання.*/
gl.drawArrays(gl.LINES, drawArrayIndexStart, drawArrayLength);
```

Перший аргумент цієї функції каже, що малюється саме лінії, другий індекс в буфері, з якого почнемо малювання, **drawArrayLength** – кількість елементів для малювання. У буфер передаються координати вершин із масиву **verticies.**

```
drawArrayLength = verticies.length / 3;
```

Якщо змінилися прямі, то потрібно виконати завантаження прямих. Якщо змінилося положення камери, то виконуємо зміну точки зору.

Висновки

У ході розроблення інтернет-додатків обрання архітектури клієнт-сервер визначає якісні характеристики продукту та є відповідальним етапом у розробленні WEB-ресурсів. Під час виконання цього етапу потрібно визначити які саме архітектури, фреймворки, методи та протоколи зв'язку, бібліотеки функцій, реалізації серверів баз даних мають використовуватися. Значну роль в якості кінцевого продукту має вибір інструментів побудови графічного інтерфейса. Обрання інтегрованої оболонки розробки визначає трудомісткість процесу розроблення, вимоги до проєктної команди, зокрема налагодження WEB-додатків.

Для якісного розроблення члени потрібно кваліфіковано оцінювати вади та переваги сучасних архітектур, фреймворків, методів та протоколів зв'язку, бібліотек функцій, реалізацій серверів баз даних.

Теоретичні запитання

1. Перерахуйте послідовність дій для отримання доступу до даних у БД.
2. Назвіть функції для підтримки дій з базою даних.

3. Дайте характеристику середовищ для розробки додатків на PHP

4. Сформулюйте послідовність дій для отримання даних із текстового файлу.

5. Сформулюйте послідовність дій для запису даних у текстовий файл.

6. Назвіть засоби, необхідні для оброблення даних із XML-файла.

7. У чому полягають переваги асинхронного взаємодії із сервером?

8. Дайте характеристику технології AJAX.

9. Що таке фреймворк?

10. Які переваги пов'язані з використанням фреймворка при розробленні?

11. Які можливості з організації асинхронного взаємодії надає JQuery?

12. Порівняйте відомі вам серверні технології створення WEB-ресурсів.

13. Назвіть відомі вам засоби створення динамічних зображень на WEB-сторінках.

14. У чому полягають переваги створення зображень на сервері?

15. Чи можна зберігати зображення в базі даних?

16. Що таке контекст тега Canvas?

17. З яких елементів може складатися зображення в Canvas?

18. Як вивести текст у Canvas?

19. Як задати колір тексту?

20. Які атрибути можуть бути присутніми в тегу Canvas?

21. Якими засобами можна створити зображення 3D на сторінці?

22. Сформулюйте послідовність дій по створенню зображення із скрипта на PHP.

Контрольні завдання

1. Побудуйте клієнт-серверний додаток, який би був спроможним обробляти контент із таблиць сервера MySQL з одного із варіантів бізнес-процесу (відділ кадрів, інтернет-магазин, бібліотека, авторемонтна майстерня).

2. Побудуйте клієнт-серверний додаток, якій би був спроможним відображати у графічній формі контент з одного з варіантів бізнес процесу (відділ кадрів, інтернет-магазин, бібліотека, авторемонтна майстерня).

3. Побудуйте клієнт-серверний додаток із використанням технології AJAX, який би був спроможним відображати контент з одного з варіантів бізнес-процесу (відділ кадрів, інтернет-магазин, бібліотека, авторемонтна майстерня).

Рекомендована література

Основна

1. Лабор В. В. Создание приложений для Windows / В. В. Лабор, Си Шарп. – Минск : Харвест, 2003. – 384 с.

2. Молчанов В. П. Основи проектування WEB-видань: навч. посіб. / В. П. Молчанов. – Харків : ХНЕУ ім. С. Кузнеця, 2017. – 150 с.

3. Томсон Л. Разработка Web-приложений на PHP и MySQL / Л. Томсон, Л. Веллингтон ; пер. с англ. – 2-е изд., испр. – Санкт-Петербург : ООО "ДиаСофтЮП", 2003. – 672 с.

4. Холзнер С. XML. Энциклопедия / С. Холзнер. – 2-е изд. – Санкт-Петербург : Питер, 2004. – 1101 с.

Інформаційні ресурси

5. 10 Лучших PHP-фреймворков [Электронный ресурс]. – Режим доступа : https://geekbrains.ru/posts/php_ides.

6. 15 JavaScript-библиотек для анимации SVG [Электронный ресурс]. – Режим доступа : https://www.internet-technologies.ru/articles/15-javascript-bibliotek-dlya-animacii-svg.html.

7. Знакомство с SVG-графикой [Электронный ресурс]. – Режим доступа : https://habr.com/ru/post/157087.

8. Знакомство с WebGL [Электронный ресурс]. – Режим доступа : https://habr.com/ru/post/112430.

9. Грималовский О. Функции регулярных выражений / О. Грималовский [Электронный ресурс]. – Режим доступа : http://code.mu/books/advanced/php/parsing/rabota-s-xml-v-php.html.

10. Справочник по HTML [Электронный ресурс]. – Режим доступа : http://htmlbook.ru/html.

Глосарій

WEB-сайт – кілька WEB-сторінок, пов'язаних інформаційно, єдиним оформленням, розміщенням і URL.

WEB-сторінка – гіпертекстовий документ, розміщений у мережі Інтернет, доступ до якого здійснюється по URL.

WEB-технологія – сукупність засобів і методів передавання та використання даних у комп'ютерній мережі Інтернет.

Динамічна графіка – створення зображень на екрані комп'ютера в ході виконання програм.

DOM (Document Object Model) – уявлення вмісту документа у вигляді дерева об'єктів, що відповідають елементам сторінки.

Клієнтський сценарій – програма, яка включена в текст сторінки у вигляді вихідного тексту та виконується браузером.

Мова розмітки (англ. markup language) – набір символьних міток (тегів), що вставляються в текст для передавання інформації про його виведення або будову.

Фреймворк (англ. Framework) – програмне забезпечення, що полегшує розроблення й об'єднання різних компонентів великого програмного проєкту.

Хостинг (англ. hosting) – послуга з надання ресурсів для розміщення даних на сервері, що підключений до мережі Інтернет.

Предметний покажчик

Зміст

НАВЧАЛЬНЕ ВИДАННЯ

Молчанов Віктор Петрович
Пандорін Олександр Костянтинович

ТЕХНОЛОГІЇ РОЗРОБКИ WEB-РЕСУРСІВ

Навчальний посібник

Самостійне електронне текстове мережеве видання

Відповідальний за видання *О. І. Пушкар*

Відповідальний редактор *М. М. Оленич*

Редактор *О. В. Анацька*

Коректор *О. В. Анацька*

План 2019 р. Поз. № 15-ЕНП. Обсяг 130 с.

Видавець і виготовлювач – ХНЕУ ім. С. Кузнеця, 61166, м. Харків, просп. Науки, 9-А

Свідоцтво про внесення суб'єкта видавничої справи до Державного реєстру
ДК № 4853 від 20.02.2015 р.