Міністерство освіти і науки України
Сумський державний університет

В. В. Шендрик, О. В. Бондар, Ю. В. Парфененко

WEB-ПРОГРАМУВАННЯ

Конспект лекцій

Суми
Сумський державний університет
2013

Міністерство освіти і науки України
Сумський державний університет

WEB-ПРОГРАМУВАННЯ

Конспект лекцій
для студентів спеціальності
(7) 8.05010102 “Інформаційні технології проектування”
денної та заочної форм навчання

Затверджено на засіданні кафедри комп’ютерних наук як конспект лекцій із дисципліни “Web-програмування”.
Протокол № 1 від 28.08.2012р.

Суми
Сумський державний університет
2013

Web-програмування : конспект лекцій / укладачі : В. В. Шендрик, О. В. Бондар, Ю. В. Парфененко – Суми : 2013. – 124 с.

Сумський держаний університет,
секція інформаційних технологій проектування кафедри комп’ютерних наук

Зміст

Передмова

Цей конспект лекцій представляє собою керівництво зі спільного застосування PHP та MySQL для розробки високоефективних та інтерактивних Web-сайтів з динамічним вмістом.

Перевагою цього конспекту лекцій є орієнтація на вирішення реальних задач, що втілено у великій кількості типових прикладів, які часто виникають при розробці.

У конспекті лекцій широко розглядається формальний синтаксис та семантика мови PHP, основи побудови додатків баз данних та особливості застосування об’єктно-орієнтованної методології при розробці додатків для Web.

1 Вступ

У лекції розглядаються такі питання:

1.1 Поняття Web-сервісу;
1.2 Архітектура Web-серверів;
1.3 Мова PHP;
1.4 Основи HTML;
1.5 Поняття URL- адреси;
1.6 MySQL - СУБД для Інтернету;
1.7 Інтеграція сценаріїв з базами даних.

1.1 Поняття Web-сервісу

Уперше термін "Web Services" з'явився в 2000 році при оголошенні інформації про нову технологію Microsoft.NET. У цей час Web-сервіси перебувають у стадії інтенсивного розвитку. Всі найбільші виробники, включаючи Sun, Microsoft, IBM й BEA, поєднуються для виробітку стандартів, проявляючи зацікавленість у їхньому швидкому прийнятті, розробляють і випускають інструментальне ПЗ, що дозволяє створювати Web-сервіси.

Web-сервіси базуються на застосуванні відкритим, затверджуваним консорціумом W3C, стандартах і протоколах, ключовими з яких є наступні:

- SOAP (Simple Object Access Protocol) — протокол доступу до простих об'єктів, тобто механізм для передачі інформації між вилученими об'єктами на базі протоколу HTTP і деяких інших Інтернет-протоколів;
- WSDL (Web Services Description Language) — мова опису Web-сервісів;

- UDDI (Universal Description, Discovery and Integration) — універсальний опис, виявлення й інтеграції (протокол пошуку сервісів в Інтернеті).

Ціль технології Web-сервісів полягає в тому, щоб перебороти обмеженість сучасних інформаційних технологій, розірвавши тверді фіксовані зв'язки між додатками й компонентами додатків, замінивши її більш гнучкими слабкими зв'язками. Модель зі слабкими зв'язками, що лежить в основі Web-сервісів, не залежить від того, чи застосовується вона для з'єднання компонентів у рамках організації або для організації комбінованого сервісу, що використовує функції від багатьох компаній.

Ухвалюючи рішення щодо вибору базової технології, необхідно враховувати цілий ряд факторів. Аргументами на користь технології Web-сервісів будуть вимоги масштабованості, відкритості, можливості доступу ззовні будь-якому клієнтові й ін. На ефективність розробки вплине й вибір інструментальних засобів і методології розробки.

Істотну роль у розробці програмних систем грає створення архітектури системи як проектування на найвищому рівні. На думку провідних IT-компаній й аналітиків (найбільш відомими з них є IBM, Microsoft, Sun Microsystems, BEA, SAP, Oracle, Gartner Group, Stencil Group, International Data Corp.), важливими й перспективними напрямками в розвитку інформаційних систем і ПЗ є архітектури, орієнтовані на сервіси (Service Oriented Architecture - **SOA**). При цьому основний акцент ставиться на SOA, що орієнтована на Internet й intranet, тобто на архітектуру веб-сервісів (Web Services Architecture - **WSA**).

Архітектура, орієнтована на сервіси (SOA), має наступні характерні риси:

1. Архітектура є розподіленою. Функціональні модулі (системи) можуть бути розподілені по безлічі обчислювальних систем і здатні до взаємодії з використанням локальних або глобальних мереж.
2. Інтерфейс функціональних модулів такий, що використання модулів не залежить від технології або платформи, у рамках якої вони реалізовані.
3. Можливий динамічний пошук і підключення потрібних функціональних модулів.
4. Архітектура базується на загальноприйнятих галузевих стандартах.

Основу архітектури, орієнтованої на сервіси, становить взаємодія трьох учасників:

- постачальника сервісу;
- споживача сервісу;
- реєстру сервісів.

Ця взаємодія представлена на рисунку 1.1

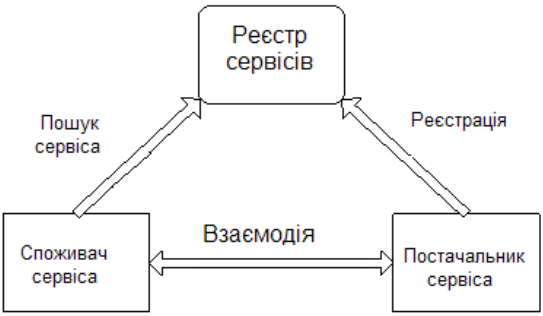

Рисунок 1.1 - Компоненти архітектури SOA

Взаємовідносини учасників включають наступні основні аспекти:

- публікація сервісу;
- пошук сервісу;
- підключення й використання.

Для реалізації SOA необхідні три типи угоди:

1. Транспортна угода: про формати й протоколи взаємодії.
2. Угода про опис функціональності сервісу, у вигляді, придатному для автоматичної генерації коду, що визначає процес взаємодії між клієнтом і постачальником сервісу.
3. Угода про спосіб виявлення сервісу.

Архітектура веб-сервісів є однією з реалізацій SOA. Поняття архітектури, орієнтованої на сервіси, склалося в ході розвитку концепції веб-сервісів. Однак існують й інші підходи до реалізації SOA: Java RMI (від Sun Microsystems), CORBA (від консорціуму OMG), DCOM (від Microsoft), DCE (запропонований асоціацією Open Group).

Стандартизацією архітектури веб-сервісів займаються робочі групи комітету W3C. Вони пропонують наступне визначення поняття ***"веб-сервіс"*****:** "веб-сервіс - це реалізована програмними засобами система для підтримки міжмашинної взаємодії через мережу. Інтерфейс сервісу описується мовою, що читається машиною, наприклад, **WSDL**. Інші системи взаємодіють із веб-сервісом способом, зазначеним у його описі, використовуючи повідомлення в стандарті **SOAP**, передані з використанням **HTTP** й **XML** й у сполученні з іншими стандартами, що відносяться до Web".

Фізично Web-сервіс являє собою фрагмент програмного забезпечення, що називається *"агентом"***.** Агент здатний передавати й приймати повідомлення, він реалізує функціональність сервісу. Існує різниця між **агентом** і **сервісом** - той самий сервіс може бути забезпечений різними агентами.

Механізм обміну повідомленнями визначається в описі сервісів (Web Services Description), що являє собою

специфікацію інтерфейсу сервісу й охоплює формати повідомлень, типи даних, транспортні протоколи, способи серріалізації, що використовуються при обміні між агентами замовника й постачальника послуг. Крім того, опис сервісу містить вказівку на одну або кілька точок у мережі (endpoint), звідки доступний постачальник.

Технологія **Universal Description, Discovery and Integration** (**UDDI**) припускає ведення реєстру веб-сервісів. Підключившись до цього реєстру, споживач зможе знайти веб-сервіси, які задовольняють його потребам. Технологія UDDI дає можливість пошуку й публікації потрібного сервісу, як людиною, так і програмою-клієнтом.

Стек технологій, що реалізує архітектуру веб-сервісів, представлений на рисунку 1.2.

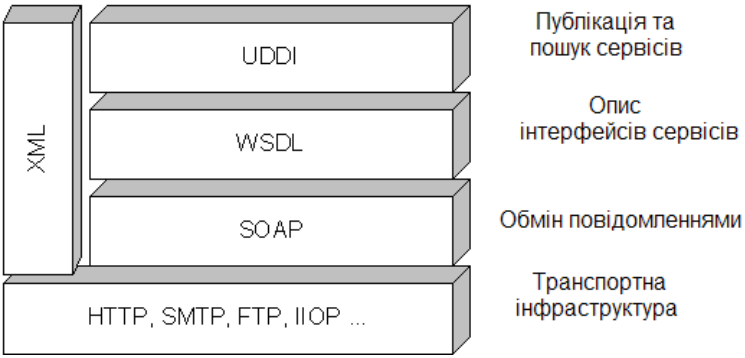

Рисунок 1.2 - Стек технологій архітектури веб-сервісів

Концепція веб-сервісів має на увазі, що окремі веб-сервіси мають певну обмежену функціональність. А для вирішення складних завдань потрібно використовувати функціональність декількох сервісів. Тому в ході розвитку архітектури веб-сервісів виникли поняття "композиція Web-сервісів" (Web-services composition) і "потік Web-

сервісів" (Web-services flow), останнім часом замість цих понять використовують відповідно "оркестровка" (Web Service Choreography) і "хореографія" (Web Service Choreography) веб-сервісів. Ці поняття відбивають взаємодію сервісів і послідовність їхнього виконання. Тобто, додатки, побудовані з використанням веб-сервісів, розглядають як додатки, засновані на потоках робіт (Workflow-based applications).

При розгляді додатків, заснованих на потоках робіт, виділяють три рівні як це показано на рисунку 1.3.

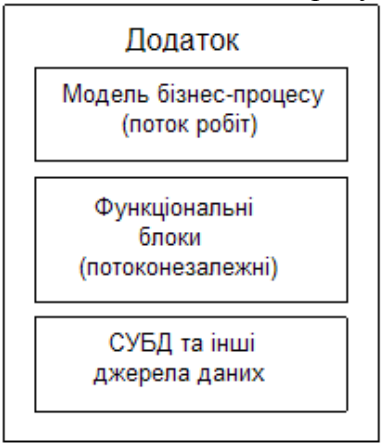

Рисунок 1.3 - Додаток, заснований на потоках робіт

Веб-сервіси є функціональними блоками й відповідають одиницям робіт у термінах потоків робіт. Способи знаходження, підключення, використання одиниць роботи (тобто, веб-сервісів) забезпечуються стеком технологій, що був описаний вище й представлений на малюнку 2.

Веб-сервіси широко використовуються для інтеграції, у тому числі для взаємодії бізнес-процесів компаній-партнерів. Це підвищує важливість механізму організації веб-сервісів.

1.2 Архітектура Web-серверів

Архітектура веб-сервісів (і більш загальна концепція - архітектура, орієнтована на сервіси) є напрямком, що динамічно розвивається, в інформаційних системах і ПЗ. Наявний набір галузевих стандартів уже сьогодні дозволяє вирішувати завдання інтеграції (у тому числі, навіть на рівні бізнес-процесів).

1.3 Мова PHP

PHP - це мова обробки гіпертексту (HTML), використовується на стороні сервера (server side scripting language), конструкції якого вставляються в HTML-текст. В 2003 році вийшла версія **PHP 5.0** на базі машини Zend Engine 2 (фірма Zend Technologies Ltd), що дає можливості для створення повномасштабних додатків, починаючи з п'ятої версії її можна називати об’єктно-орієнтованою. **PHP** є відкритим програмним продуктом, що означає його безкоштовність і можливість створювати свої власні розширення мови.

Мова **PHP** використовується приблизно на 52% з 14,5 мільйонів сайтів, що працюють під Apache, що у свою чергу є найпоширенішою в Інтернеті (за різними оцінками близько 70%), популярність якої швидко росте.

PHP є наймолодшою, перспективною і швидкою у розвитку з мов програмування для Інтернет, частка її використання в порівнянні з іншими мовами швидко росте. Її основні переваги: широка підтримка різних технологій, сумісність із серверами, базами даних, простота й безкоштовність.

PHP дозволяє відокремити HTML-текст від виконуваної частини, за рахунок чого можна домогтися значного зниження витрат часу на розробку проекту. У

багатьох випадках вдається відокремити програмну частину проекту від розробки сторінок на HTML, що полегшує завдання й дизайнерові, і програмістові. Майже завжди виявляється, що швидкість просування проектів, створюваних на **PHP** буде вище, ніж при використанні інших мов програмування.

Можливості PHP

- підтримувані технології: платформи Win32 (9x/NT/2000/XP), Windows 7, Windows 8, UNIX, OS/2, QNX, MacOS, BeOS, OCX;
- сумісність із серверами: Apache (Win32, UNIX), phttpd, fhttpd, thttpd, ISAPI(Zeus, IIS), NSAPI, Roxen/Caudium, AOLServer;
- підтримка технологій COM, XML, Java, CORBA, WDDX, Macromedia Flash;
- розвинена функціональність для роботи з мережними з'єднаннями;
- підтримує понад 20 баз даних і має розвинену функціональність для роботи з ними;
- можливість створення повноцінних об’єктно-орієнтованих додатків;
- порівняно простий синтаксис і зручність у практичному використанні;
- безкоштовність;
- відкритість коду, завдяки якій можна створювати власні розширення мови

1.4 Основи HTML

HTML (HyperText Markup Language) - мова розмітки гіпертексту - призначена для створення Web-Сторінок.

Під гіпертекстом у цьому випадку розуміється текст, пов'язаний з іншими текстами покажчиками-посиланнями.

HTML являє собою досить простий набір кодів, які описують структуру документа. HTML дозволяє виділити в тексті окремі логічні частини (заголовки, абзаци, списки й т.д.), помістити на Web-сторінку підготовлену фотографію або малюнок, організувати на сторінці посилання для зв'язку з іншими документами. HTML не задає конкретні й точні атрибути форматування документа. Конкретний вид документа остаточно визначає тільки програма-браузер на комп'ютері користувача Інтернету.

HTML також не є мовою програмування, але web-сторінки можуть містити в собі убудовані програми-скрипти на мовах Javascript і Visual Basic Script і програми-апплети мовою Java.

Основними компонентами HTML є:

- Тег (tag). Тег HTML це компонент, що командує Web-браузеру виконати певне завдання типу створення абзацу або вставки зображення.
- Атрибут (або аргумент). Атрибут HTML змінює тег. Наприклад, можна вирівняти абзац або зображення усередині тегу.
- Значення. Значення привласнюються атрибутам і визначають внесені зміни. Наприклад, якщо для тегу використовується атрибут вирівнювання, то можна вказати значення цього атрибута. Значення можуть бути текстовими, типу left або right, а також числовими, як наприклад ширина й висота зображення, де значення визначають розмір зображення в пікселях.

Теги являють собою зарезервовані послідовності символів, що починаються з < (знака менше) і що закінчуються > (знаком більше). Закриття тегу відрізняється від відкриття тільки наявністю символу '/'.

Припустимо, у нас є гіпотетичний атрибут форматування тексту, керований кодом <X>, і ми хочемо застосувати його до слів "Це мій текст". HTML-послідовність кодів і власне тексту буде виглядати так:

<X>Це мій текст</X>

Теги можуть вкладатися один в інший ієрархічно, але без перетинань, тобто припустиме вкладення виду <teg1><teg2></teg2> </teg1>, але не <teg1><teg2> </teg1></teg2>. Дія вкладених тегів поєднується. Наприклад, якщо усередину тегу, що створює жирне накреслення шрифту, вкладений тег курсиву, то в результаті вийде жирний курсив.

1.5 Поняття URL-адреси

URL використовується в гіпертекстових посиланнях і забезпечує доступ до розподілених ресурсів мережі. В URL можна адресувати як інші гіпертекстові документи формату HTML, так і ресурси e-mail, telnet, ftp, Gopher, WAI, наприклад. Різні інтерфейсні програми по різному здійснюють доступ до цих ресурсів.

Одні, як наприклад Netscape, самі здатні підтримувати взаємодію по протоколах, відмінним від протоколу HTTP, базового для WWW, інші, як наприклад Chimera, викликають для цієї мети зовнішні програми. Однак, навіть у першому випадку, базовою формою подання відображуваної інформації є HTML, а посилання на інші ресурси мають форму URL. Слід зазначити, що програми обробки електронної пошти у форматі MIME також мають можливість відображати документи, представлені у форматі HTML. Для цієї мети в MIME зарезервований тип "text/html".

1.6 MySQL - СУБД для Інтернету

MySQL – це одна із най популярніших і найпоширеніших СУБД (система управління базами даних) в Інтернеті. Вона не призначена для роботи з великими обсягами інформації, але її застосування ідеальне для Інтернет сайтів, як невеликих, так і досить великих.

MySQL відрізняться гарною швидкістю роботи, надійністю, гнучкістю. Робота з нею, як правило, не викликає великих труднощів. Підтримка сервера MySQL автоматично включається в поставку PHP.

Немаловажним фактором є її безкоштовність. MySQL поширюється на умовах загальної ліцензії GNU (GPL, GNU Public License).

Раніше для довгострокового зберігання інформації працювали з файлами: поміщали в них деяку кількість рядків, а потім витягали їх для наступної роботи. Завдання тривалого зберігання інформації дуже часто зустрічається в програмуванні Web-додатків: підрахунок відвідувачів у лічильнику, зберігання повідомлень у форумі, вилучене керування змістом інформації на сайті й т.д.

Тим часом, професійні прийоми роботи з файлами дуже трудомісткі: необхідно піклуватися про розміщення в них інформації, про її сортування, добування, при цьому не потрібно забувати, що всі ці дії будуть відбуватися на сервері хост-провайдера. При цьому, обсяг коду значно зростає, і зробити помилку в програмі дуже просто.

Всі ці проблеми вирішує використання бази даних. Бази даних самі піклуються про безпеку інформації і її сортування й дозволяють витягати й розміщати інформацію за допомогою одного рядка. Код з

використанням бази даних виходить більше компактним, і налагоджувати його набагато легше.

1.7 Інтеграція сценаріїв з базами даних

Однією із причин популярності сервера баз даних MySQL, поряд з його доступністю й продуктивністю, можна вважати його інтеграцію з PHP. При цьому продуктивність зв'язки PHP, Apache й MySQL у більшості випадків можна вважати однією з найвищих у порівнянні з іншими рішеннями. Стандартна зборка PHP майже завжди містить у собі бібліотеки для роботи з MySQL, надаючи розроблювачам всі необхідні інструменти для взаємодії із сервером баз даних. Функції для роботи з MySQL, які доступні при написанні проектів з використанням PHP, вирішують головні завдання, що виникають при необхідності звертання до бази даних зі сценарію PHP - з'єднання із сервером баз даних, передача йому запиту й добування результатів, які повернув сервер.

Перед тим як ми одержимо можливість працювати з інформацією, збереженою в базі даних, необхідно встановити з'єднання із сервером баз даних. Для цього призначені 2 функції PHP, що майже нічим не відрізняються за результатами дії: mysql_pconnect() і mysql_connect(). Єдина відмінність цих функцій полягає в тім, що перша з них встановлює постійне з'єднання із сервером баз даних, що залишається відкритим навіть після того, як ваш сценарій буде виконаний, і його неможливо буде закрити навіть функцією mysql_close(). Коли у вашому PHP-сценарії запускається функція mysql_pconnect(), вона попередньо перевірить, немає чи відкритого раніше постійного з'єднання, і якщо воно є, то відкривати нове вона вже не буде. Подібний підхід заощаджує час і знімає навантаження із сервера баз даних.

Для від'єднання від сервера баз даних існує функція mysql_close(), але, як правило, у її використанні немає необхідності. Якщо було встановлено постійне з'єднання, то команда на закриття його буде зігнорована, а у випадку використання функції mysql_connect() з'єднання буде розірвано по закінченні роботи PHP-сценарієм.

2 Інтеграція PHP з Web-сторінками

У лекції розглядаються такі питання:

2.1 Використання PHP.
2.2 Вбудова PHP в HTML.
2.3 Додавання динамічного вмісту.
2.4 Доступ до змінних форми.

2.1 Використання PHP

PHP (англ. PHP: HypertextPreprocessor— «PHP: препроцесоргіпертексту», англ. PersonalHomePageTools — «Інструменти для створення персональних веб-сторінок») — мова програмування, створена для генерування HTML-сторінок на веб-сервері і роботи з базами даних. У даний час підтримується переважною більшістю хостинг-провайдерів. Входить в LAMP — «стандартний» набір для створення веб-сайтів (Linux, Apache, MySQL, PHP (Python або Perl)).

Група розроблювачів PHP складається з великої кількості людей, що добровільно працюють над ядром і розширеннями PHP і суміжними проектами, такими як PEAR або документація мови.

В області програмування для Мережі, PHP — одна з най популярніших скриптових мов (поряд з JSP, Perl і мовами, використовуваними в ASP.NET) завдяки своїй

простоті, швидкості виконання, багатій функціональності й поширенню вихідних кодів на основі ліцензії PHP. PHP відрізняється наявністю ядра й модулів, що підключають, «розширень»: для роботи з базами даних, сокетами, динамічною графікою, криптографічними бібліотеками, документами формату PDF і т.п. Будь-хто бажаючий може розробити своє власне розширення й підключити його. Існують сотні розширень, однак у стандартну поставку входить лише кілька десятків, що добре зарекомендували себе. Інтерпретатор PHP підключається до веб-серверу або через модуль, створений спеціально для цього сервера (наприклад, для Apache або IIS), або як CGI-додаток.

Крім цього, він може використовуватися для рішення адміністративних завдань в операційних системах UNIX, GNU/Linux, Microsoft Windows, Mac OS X і AmigaOS. Однак у такій якості він не одержав поширення, віддаючи перевагу Perl, Python і VBScript.

У цей час PHP використовується сотнями тисяч розроблювачів. Порядку 20 мільйонів сайтів повідомляють про роботу з PHP, що становить більше п'ятої частки доменів Інтернету.

Один з найпоширеніших додатків будь-якої мови створення серверних сценаріїв - обробка HTML-форм. Вивчення PHP почнемо з реалізації форми замовлення для вигаданої компанії із продажу компакт дисків. Всі вихідні коди наведені нижче.

Приклад 1: Форма замовлення.

У цей час програміст якоїсь компанії із продажу CD займається створенням форми для їхнього продажу. Вона показана на рисунку 2.1. Це проста форма, на відміну від багатьох інших, котрими часто користуються власники віртуальних магазинів і т.п. Насамперед, директор нашої вигаданої фірми, хотів би знати, що замовили його клієнти,

обчислити загальну суму замовлення й можливо податок із продажів, що буде потрібно сплатити по виконанню замовлення.

Вихідний html-код нашого замовлення:

```
<html>
<head>
  <title>Форма замовлення</title>
</head>
<body>
<h1>Продаж CD</h1>
<h2>Форма замовлення:</h2>

<form action="cd.php" method=post>
<table border=0>
<tr bgcolor=#cccccc>
  <td width=150>Товар</td>
  <td width=15>Кількість</td>
</tr>
<tr>
  <td>CD-R</td>
  <td align=center><input type="text"
name="cdr" size=3 maxlength=3></td>
</tr>
<tr>
  <td>CD-RW</td>
  <td align=center><input type="text"
name="cdrw" size=3 maxlength=3></td>
</tr>
<tr>
  <td>Футляри для CD</td>
  <td align=center><input type="text"
name="fut" size=3 maxlength=3></td>
</tr>
<tr>
  <td colspan=2 align=center><input
type=submit value="Зробити замовлення"></td>
```

```
</tr>
</table>
</form>

</body>
</html>
```

Результат:

Рисунок 2.1 - Форма замовлення

По-перше, дії, що виконуються формою, привласнене ім'я PHP-сценарію, що буде обробляти замовлення клієнта. У загальному випадку значенням атрибута ACTION є URL-адреса, що буде завантажуватися при натисканні користувачем у форму й буде відправлятися по цій URL-адресі з використанням методу, зазначеного в атрибуті METHOD: або GET (дані приєднуються в кінець URL-адреси), або POST (дані відправляються у вигляді окремого пакета).

По-друге, варто звернути увагу на імена полів форми - cdr, cdrw й fut. Ці імена будуть знову використовуватися в PHP-сценарії. Тому полям форми важливо привласнювати осмислені імена, які легко запам'ятати при написанні PHP-сценарію.

Приклад 2: Обробка форми

Для обробки форми буде потрібно створити сценарій, згаданий в атрибуті ACTION дескриптора FORM і названий cd.php. У текстовому редакторі створюємо цей файл. Для цього вводимо наступний html-код:

```
<html>
<head>
  <title>Замовлення CD. Результати замовлення</title>
</head>
<body>
<h1>Продаж CD</h1>
<h2>Результати замовлення:</h2>
</body>
</html>
```

Це все є звичайним HTML-текстом. Додамо в сценарій PHP-код.

2.2 Вбудовування PHP в HTML

Під заголовком <h2> файлу вводимо наступні рядки:

```
<?
echo "Замовлення оброблене.";
?>
```

Збережемо файл і завантажимо його у свій браузер, потім заповнимо форму й натиснемо кнопку "Відправити". На екрані повинне відобразитися щось схоже на зображення, показане на рис. 2.2.

Рисунок 2.2. Результати замовлення

Зверніть увагу, як написаний PHP-код вбудовується у звичайний HTML-файл. У браузері ви повинні побачити наступні рядки коду:

```
<html>
<head>
  <title> Замовлення CD. Результати замовлення </title>
</head>
<body>
<h1>Продаж CD</h1>
<h2>Результати замовлення:</h2>
<p> Замовлення оброблене </p>
</body>
</html>
```

Рядків PHP-коду не видно. Це відбувається тому, що інтерпретатор PHP переглянув сценарій і замінив його рядками виводу. Отже, із середовища PHP можна створити чистий код HTML, придатний для перегляду в будь-якому браузері - інакше кажучи, браузер, що застосовується користувачем не обов'язково повинен розуміти PHP.

Тепер код у розглянутому файлі складається із чотирьох частин:

- HTML;

- Дескриптори PHP;
- Оператори PHP;
- Пробіли.

У нього можна додати також ще одну частину:

- Коментарі.

Більшість рядків у наведеному прикладі - усього лише простий HTML-код.

2.3 Додавання динамічного вмісту

Дотепер ми не використовували PHP для виконання яких-небудь дій, які не можна було б реалізувати за допомогою звичайного html.

Розглянемо простий приклад. Замінимо PHP-код у файлі cd.php на наступний код:

```
<?
echo "Ваше замовлення було прийнято в ";
echo date("H:i, j F");
echo "<br>";
?>
```

У цьому коді убудована PHP-функція date() використовується для повідомлення клієнтові дати й часу обробки замовлення. Це значення буде змінюватися при кожнім виконанні сценарію. Вивід, отриманий у результаті одного такого виконання сценарію показаний на рисунку 2.3

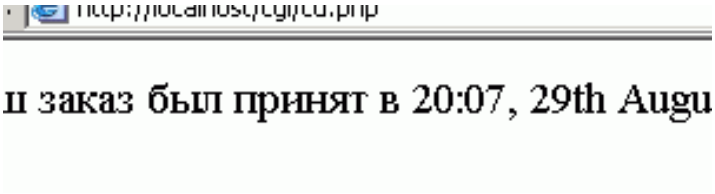

Рисунок 2.3. Виведення дати й часу

Виклик функції

Поглянемо на виклик функції date(). Це загальна форма виклику функції. PHP має велику бібліотеку функцій, які можна використовувати при розробці web-додатків. Виклик функції: **date("H:i, j F").**

Зверніть увагу, що переданий функції рядок укладений у круглі дужки. Це значення називається аргументом, або параметром функції.

Функція date()

Аргумент, переданий у функцію date(), повинен бути рядком формату, що задає необхідний стиль виводу. Кожна буква в рядку представляє частину рядка дати й часу. H представляє години в 12-годинному форматі, i - хвилини із провідним нулем, коли потрібно, j - день місяця без провідного нуля, S представляє звичайний суфікс, а F - рік, представлений чотирма цифрами.

2.4 Доступ до змінних форми

Весь зміст використання форми замовлення полягає в одержанні інформації про замовлення клієнта. Одержання докладної інформації про те, що ввів клієнт, реалізується в PHP дуже просто.

Усередині PHP-сценарію до кожного з полів форми можна одержати доступ як до змінної, що має те ж ім'я, що й у поля форми. Розглянемо приклад.

Додамо наступні рядки в нижню частину PHP-сценарію:

```
echo "<p>Ваше замовлення було таким:";
echo "<br>";
echo $cdr." CDR-диск(ов)<br>";
echo $cdrw." CDRW-диск(ов)<br>";
echo $fut." футляр(ов)<br>";
```

Після відновлення вікна браузера вивід сценарію повинен виглядати подібно показаному рисунку 2.4. Звичайно ж, фактичні значення будуть залежати від того, що уведено у форму.

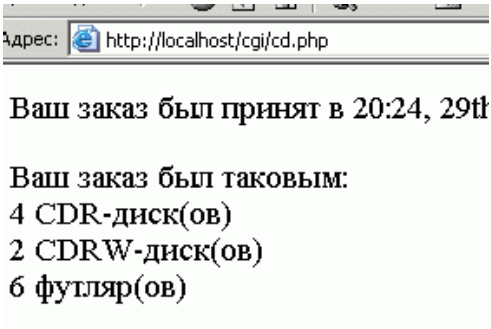

Рисунок 2.4 - Вивід замовлення

Змінні форми

В остаточному підсумку, дані зі сценарію попадають в PHP-змінні. Імена змінних в PHP легко розпізнати, оскільки всі вони починаються із символу долара ($). (Пропуск символу долара - ще одна розповсюджена помилка програмування). Існують два способи доступу до даних форми через змінні.

У цьому прикладі для посилання на змінні форми використовується скорочений стиль. При використанні

цього стилю можна, наприклад, просто почати роботу зі змінної, скажемо, $CDR.

Другий стиль полягає в одержанні змінних форми через один з 2-х масивів, що зберігаються в змінних $HTTP_POST_VARS й $HTTP_GET_VARS. Один із цих масивів буде містити докладну інформацію про всі змінні форми. Вибір використовуваного масиву залежить від методу відправлення форми: POST або GET.

Скорочений стиль можна застосовувати тільки при установці в значенні "On" директиви register_globals у файлі **php.ini**. Це - настроювання за замовчуванням у стандартному файлі php.ini.

Обоє ці методи аналогічні методам, використовуваних в інших мовах створення сценаріїв, наприклад, Perl, і можуть виглядати знайомо.

Конкатенація рядків

У сценарії оператор echo застосовувався для виводу значень, уведених користувачем у кожному з полів форми, за якими ішов деякий пояснювальний текст. Якщо уважно придивитися до операторів echo, можна помітити, що між ім'ям змінної й наступним за ним текстом міститься крапка(.), наприклад:

```
echo $CDR. " CDR-диск(ов)<br>";
```

Це операція конкатенації рядків, що використовується для об'єднання рядків (фрагментів тексту). Вона буде часто застосовуватися при пересиланні виводу в браузер за допомогою оператора echo. Ця операція дозволяє уникати запису декількох команд echo. Інакше можна було б записати так:

```
echo "$CDR CDR-диск(ов)<br>";
```

Цей оператор еквівалентний першому. Обидва формати припустимі й використання кожного з них - особиста справа кожного.

Змінні та літерали

Змінні й рядки, поєднувані в кожному з операторів echo, - різні поняття. Змінні - це символи (позначення) для даних. Рядки ж - це і є дані. Фрагмент неструктурованих даних у програмі подібній розглянутій називається літералом, на відміну від змінної. $CDR - це змінна, тобто символ, що представляє уведені клієнтом дані. З іншого боку, "CDR" - це літерал. Він приймається так, як виглядає.

Фактично, в PHP існують два види рядків - з подвійними лапками й одинарними лапками. PHP буде намагатися оцінити рядки, укладені в подвійні лапки. Рядки, укладені в одинарні лапки, будуть оброблятися, як справжні літерали.

3 Основний синтаксис PHP

У лекції розглядаються такі питання:

3.1 Базовий синтаксис.
3.2 Типи даних.
3.3 Константи.
3.4 Операції.
3.5 Керуючі структури (альтернатива, варіант, цикли).

3.1 Базовий синтаксис

Вихід з HTML

Коли PHP розбирає файл, він просто передає текст файлу, поки не виявить один зі спеціальних тегів, що говорить про необхідність почати інтерпретацію тексту як коду PHP. Розбирач виконує весь знайдений код до закриваючого тегу PHP, що говорить розбирачеві, що потрібно знову почати просто передавати текст. Цей механізм дозволяє впроваджувати PHP-код в HTML: усе за межами тегів PHP залишається без змін, а усередині тегів - розбирається як код.

Є чотири набори тегів, які використовуються для позначення блоків коду PHP. Тільки два з них (<?php. . .?> і <script language="php">. . .</script>) завжди доступні; інші можна включати й відключати з файлу конфігурації php.ini.

Теги, підтримувані PHP (способи виходу з HTML):

1. <?php echo("якщо ви хочете працювати з документами XHTML або XML, робіть так");
   ?>

2. <? echo ("це найпростіша SGML- Інструкція процесинга ");
?>
<?= вираз ?> Це абревіатура для "<? echo вираз ?>"

3. <script language="php">
echo ("деякі редактори (ніби FrontPage) не люблять інструкції процесинга");
</script>

4. < % echo ("Ви можете на вибір використовувати теги в стилі ASP"); %>
< %= $variable; # Це абревіатура для "<% echo . . ." %>

Перший спосіб, <?php. . .?>, це кращий метод, тому що він дозволяє використовувати PHP у коді, що відповідає правилам XML, такому як XHTML.

Другий спосіб взагалі неможливий. Скорочені теги доступні тільки тоді, коли вони підключені. Це можна зробити функцією short_tags() (тільки в PHP 3), включивши установку конфігурації short_open_tag в PHP config-файлі, або скомпілювавши PHP з опцією -- -i enable-short-tags в configure. Навіть якщо ви за замовчуванням включили в php.ini-dist, використання скорочених тегів не рекомендується.

Четвертий спосіб доступний, тільки якщо теги в стилі ASP включені з використанням установки конфігурації asp_tags.

Поділ інструкцій

Інструкції розділяються так само, як і в C - кожний оператор закінчується символом "крапка з комою".

Закриваючий тег (?>) також має на увазі кінець оператора, тому наступні варіанти еквівалентні.

3.2 Типи даних

PHP підтримує 8 примітивних типів.
4 скалярних типи:

- Boolean
- Integer
- число із плаваючою крапкою (float)
- string

Два складових типи:

- array
- object

І, нарешті два спеціальних типи:

- resource
- NULL

Тип змінної звичайно програмістом не встановлюється; навпроти, він визначається PHP на етапі прогону, залежно від контексту, у якому ця змінна використовується.

Булев - це найпростіший тип. **boolean** виражає правильність значення. Значення може бути TRUE або FALSE.

Цілі числа (integer) - це число з набору Z = {..., -2, -1, 0, 1, 2, ...}

Цілі числа можуть специфікуватися в десятеричній (база 10), 16- річній (база 16) або 8- річній (база 8) нотації з необов'язковим знаком (- або +).

Якщо ви використовуєте 8-річну нотацію, першим символом числа повинен бути 0 (нуль), для 16- річної нотації першими символами числа будуть 0x.

Цілочисленні літерали

```
$a = 1234; # 10- річне число
```

```
$a = -123; # негативне число
$a = 0123; # 8- річне число (еквівалентне 10- річному 83)
$a = 0x1A; # шістнадцятирічне число (еквівалентне 10-річному 26)
```

Розмір integer залежить від платформи, хоча максимальне значення близько 2 мільйонів є звичайним (тобто 32- бітне знакове). PHP не підтримує беззнакові integer.

Числа із плаваючою крапкою ("float", "double" або "real") можуть специфікуватись з використанням наступних видів синтаксису:

```
$a = 1.234; $a = 1.2e3; $a = 7 E-10;
```

Розмір float залежить від платформи, хоча максимально можлива величина ~1.8e308 з точністю, грубо, 14 десяткових цифр (тобто 64 бітний IEEE-Формат).

Рядки (string) - це серія символів. В PHP символ це те ж саме, що й байт, тобто є точно 256 різних можливих символів. Це також припускає, що в PHP немає убудованої підтримки Unicode.

Рядковий літерал може специфікуватись трьома способами.

- одинарними лапками
- подвійними лапками
- heredoc-синтаксисом

Масив в PHP це впорядкована карта. Карта/map це тип, що відображає значення в ключі. Цей тип оптимізується різними способами, тому ви можете

використовувати його як реальний масив або список (вектор), кеш-таблицю (яка є реалізацією карти), словник/dictionary, колекцію/collection, стек/stack, чергу/queue і, можливо, щось ще.

Ресурс це спеціальна змінна, що містить посилання на зовнішній ресурс. Ресурси створюються й використовуються за допомогою спеціальних функцій.

Спеціальне **значення NULL** представляє змінну, що не має значення. NULL це єдино можливе значення типу NULL.

Змінна вважається NULL, якщо

- їй привласнена константа NULL
- їй взагалі не привласнене яке-небудь значення
- вона була unset()

3.3 Константи

Константа - це ідентифікатор (ім'я) для простого значення. Як слідує з назви, це значення не може змінюватися в процесі виконання скрипта. ("Магічні константи" __FILE__ і __LINE__ є виключенням із цього правила, але вони в дійсності не є константами.) За замовчуванням константа чутлива до регістра символів. За згодою ідентифікатори констант уводяться у верхньому регістрі.

Ім'я константи дотримує ті ж правила, що й будь-яка мітка PHP. Правильне ім'я константи починається з букви (латинської) або символу підкреслення, за якої може випливати будь-яка кількість букв, цифр або символів підкреслення.

Область видимості константи є глобальною - можна одержувати до неї доступ у будь-якій частині скрипта, незалежно від області видимості.

Синтаксис

Визначають константу функцією define(). Після того як константа визначена, вона не може бути змінена або розвизначена/undefined.

Тільки скалярні дані (boolean, integer, float і string) можуть утримуватися в константах.

Можна одержати значення константи, просто специфікувавши її ім'я. На відміну від змінних, не потрібно вводити перед константою символ $. Ви можете також використовувати функцію constant() для читання значення константи, якщо ви одержуєте ім'я константи динамічно. Використовуйте get_defined_constants() для одержання списку всіх певних констант.

Між константами й змінними є відмінності:

- Перед ім'ям константи немає знака dollar ($);
- Константи можуть бути визначені тільки через використання функції define(), але не простим присвоєнням;
- Константи можуть бути визначені, і доступ до них може бути отриманий, у будь-якому місці, поза залежністю від правил області видимості змінних;
- Константи не можуть бути перевизначені або розвизначені після свого визначення;
- Константи можуть обчислюватися тільки в скалярні значення.

Визначення констант

```
<?php
define("CONSTANT", "Hello world.");
echo CONSTANT; // виводить "Hello world."
```

```
echo Constant; // виводить "Constant" і повідомлення.
?>
```

3.4 Операції

Оператори виконують безпосередню дію над аргументами й, можливо, повертають результат. Мова PHP надає досить багато операторів.

Арифметичні операції

Арифметичні оператори служать для виконання арифметичних дій над аргументами. Всі арифметичні оператори, підтримувані PHP: + (сума), - (різниця), * (добуток), / (частка від ділення), % (цілочислений залишок від ділення).

Операція ділення ("/") завжди повертає число із плаваючою крапкою, навіть якщо операнди є цілими числами (або рядками, які конвертуються в цілі числа).

Операції присвоєння

Базова операція присвоєння це "=". Вона означає, що лівий операнд одержує значення правого виразу (тобто це значення в нього "установлюється").

Значенням виразу присвоєння є привласнене значення. Тобто значення "$a = 3" дорівнює 3. Це дозволяє виконувати деякі трюки:

```
$a = ($b = 4) + 5; // $a зараз дорівнює 9, а $b має значення
4.
```

Бітові операції

Бітові операції дають можливість установлювати значення специфікованих бітів цілочисленого значення. Якщо лівий і правий параметри є рядками, бітова операція виконується над символами рядка.

Таблиця 3.1 - Бітові операції

Приклад	Ім'я	**Результат**
$a & $b	And	Установлюються біти, які встановлені й в $a, і в $b.
$a | $b	Or	Установлюються біти, які встановлені в $a або в $b.
$a ^ $b	Xor	Установлюються біти, які встановлені в $a або $b, але не в обох.
~ $a	Not	Установлюються біти, які в $a не встановлені, і навпаки.
$a << $b	Зсув вліво	Зрушує біти змінної $a на $b кроків уліво (кожний крок/зсув означає "помножити на 2").
$a >> $b	Зсув вправо	Зрушує біти змінної $a на $b кроків вправо (кожний крок/зсув означає "розділити на 2").

Операції порівняння

Операції порівняння, як видно з назви, порівнюють два значення.

Таблиця 3.2 - Операції порівняння

Приклад	**Назва**	**Результат**
$a == $b	дорівнює	**TRUE**, якщо $a дорівнює $b.
$a === $b	ідентично	**TRUE**, якщо $a дорівнює $b і вони одного типу. (тільки в PHP 4)
$a != $b	не дорівнює	**TRUE**, якщо $a не дорівнює $b.
$a <> $b	не дорівнює	**TRUE**, якщо $a не дорівнює $b.

Продовження таблиці 3.2 - Операції порівняння

Приклад	Назва	Результат
$a !== $b	не ідентично	**TRUE**, якщо $a не дорівнює $b або вони різних типів. (тільки в PHP 4)
$a < $b	меньше	**TRUE**, якщо $a строго менше $b.
$a > $b	більше	**TRUE**, якщо $a строго більше $b.
$a <= $b	меньше або дорівнює	**TRUE**, якщо $a менше або дорівнює $b.
$a >= $b	більше або дорівнює	**TRUE**, якщо $a більше або дорівнює $b..

Умовною операцією є операція "?:" (тернарна), що оперує так само, як в C і багатьох інших мовах.

(expr1) ? (expr2) : (expr3);

Цей вираз обчислюється в expr2, якщо expr1 обчислюється в TRUE, і в expr3, якщо expr1 обчислюється в FALSE.

Операції керування помилками

PHP підтримує одну операцію керування помилками: знак (@). Якщо він вставлений як префікс виразу PHP, будь-які помилки, які можуть генеруватися цим виразом, придушуються.

Якщо увімкнено track_errors, будь-які повідомлення про помилки, що генеруються цим виразом, будуть

зберігатися в глобальній змінній $php_errormsg. Ця змінна буде перезаписуватися при виникненні кожної нової помилки, тому перевіряйте її відразу, якщо необхідно.

Операції виконання

PHP підтримує одну операцію виконання: зворотні апострофи (''). Зверніть увагу, що це не одинарні лапки! PHP намагається виконати вміст усередині зворотних апострофів як команду оболонки; вертається вивід (тобто він не просто видається на виході; він може бути привласнений змінній).

```
$output = 'ls -al';
echo "<pre>$output</pre>";
```

Операції інкремента/декремента

PHP підтримує операції pre- і post- інкремента й декремента в стилі C.

Таблиця 3.3 - Операції інкремента/декремента

Прикладр	Назва	Ефект
++$a	Pre-increment	Збільшує $a на 1, потім повертає $a.
$a++	Post-increment	Повертає $a, потім збільшує $a на 1.
--$a	Pre-decrement	Зменшує $a на 1, потім повертає $a.
$a--	Post-decrement	Повертає $a, потім зменшує $a на 1.

Логічні операції

Перелік основних логічних операцій наведено у таблиці 3.4

Таблиця 3.4 - Логічні операції

Приклад	Ім'я	Результат
$a and $b	And	**TRUE**, якщо и $a, і $b **TRUE**.
$a or $b	Or	**TRUE**, якщо $a або $b **TRUE**.
$a xor $b	Xor	**TRUE**, якщо $a або $b **TRUE**, но не оба.
! $a	Not	**TRUE**, якщо $a не **TRUE**.
$a && $b	And	**TRUE**, якщо і $a, і $b **TRUE**.
$a || $b	Or	**TRUE**, якщо $a або $b **TRUE**.

Строкові операції

Є дві строкові операції. Перша - операція конкатенації ("."), що повертає об'єднання із правого й лівого аргументів. Друга - операція присвоєння (".="), що приєднує правий аргумент до лівого аргументу.

```
$a = "Hello ";
$b = $a . "World!"; // тепер $b містить "Hello World!"

$a = "Hello ";
$a .= "World!";     // тепер $a містить "Hello World!"
```

3.5 Керуючі структури

Будь-який PHP-скрипт складається із серії операторів. Це може бути присвоєння, виклик функції,

цикл, умовний оператор або навіть оператор, що нічого не робить (порожній оператор).

Оператор звичайно завершується крапкою з комою. Крім того, оператори можна групувати за допомогою фігурних дужок {}. Група операторів сама також є оператором.

if

Конструкція if є однією із ключових у багатьох мовах, у тому числі й в PHP. Вона дозволяє виконувати фрагменти коду при виконанні умови. PHP пропонує структуру if, що аналогічна такій же структурі мови C:

```
if (expr) statement
```

Оператори if можуть вкладатися один в одного, що дає вам повну волю при умовному виконанні різних частин програми.

else

Часто потрібно виконати оператор, якщо дотримано яку-небудь умову, і інший оператор - якщо умова не дотримана. Для цього призначений оператор else.

else розширює оператор if і виконує свої оператори, якщо перевіряється вираз, що, в операторі if обчислюється в FALSE. Наприклад, наступний код виведе a is bigger than b, якщо $a виявиться більше $b, і a is NOT bigger than b - у противному випадку:

```
if ($a > $b) {
   print "a is bigger than b";
} else {
   print "a is NOT bigger than b";
}
```

Оператор else виконується тільки в тому випадку, якщо вираження if обчислюється в FALSE.

Elseif

Elseif – це комбінація if і else. Подібно else, вона розширює оператор if для виконання інших операторів у тому випадку, якщо оригінальний вираз if обчислюється в FALSE. Однак, на відміну від else, elseif буде виконувати альтернативний вираз тільки тоді, коли умовне вираження в elseif буде обчислене в TRUE.

Може бути кілька elseif усередині одного оператора if. Перший вираз elseif (якщо є), що обчислюється в TRUE, буде виконуватися. В PHP ви можете також записати 'else if' (двома словами), і поводження буде ідентично 'elseif' (в одне слово). Синтаксично значення злегка розрізняються, але в результаті поведінка буде зовсім аналогічною.

Оператор elseif виконується тільки в тому випадку, якщо попереднє вираження if і будь-яке попереднє вираження elseif обчислені в FALSE, а поточне вираження elseif обчислене в TRUE.

Альтернативний синтаксис структур керування

PHP пропонує альтернативний синтаксис для деяких структур керування: if, while, for, foreach і switch. У кожному випадку базова форма синтаксису змінюється - відкриваюча фігурна дужка заміняється на двокрапку (:), а закриваюча - на endif;, endwhile;, endfor;, endforeach; або endswitch;, відповідно.

```
<?php if ($a == 5): ?>
A is equal to 5
<?php endif; ?>
```

У цьому прикладі HTML-блок "A is equal to 5" вкладений в оператор if, записаний в альтернативному синтаксисі. Це HTML-блок буде відображений, тільки якщо значення $a дорівнює 5.

switch

Оператор switch нагадує виконання серії операторів IF над одним виразом. У багатьох випадках вам потрібно буде порівняти одну змінну (або вираз) з різними значеннями й виконати різні ділянки коду залежно від того, яке значення. Саме це допоможе зробити оператор switch. Приклад:

```
switch ($i) {
    case 0:
        print "i equals 0";
        break;
    case 1:
        print "i equals 1";
        break;
    case 2:
        print "i equals 2";
        break;
}
```

Оператор switch виконується порядково (фактично - оператор за оператором). На початку ніякий код не виконується. Тільки коли буде знайдений оператор case зі значенням, що збігається зі значенням виразу switch, PHP почне виконувати оператори. PHP продовжує виконання операторів до кінця блоку switch або до першого оператора break. Якщо ви не запишете оператори break наприкінці оператора case у списку, PHP буде виконувати оператори наступного case.

Цикли

Дуже часто необхідно виконати неодноразово яку-небудь дію в програмі. Якщо ці частини в програмі повторюються, то використовують, так званий, цикл. Таким чином, цикл - це повторювана дія, як у комп'ютерному світі, так і в житті.

Цикли while

Найпростіший вид циклу в php - цикл while. Цей оператор заснований на перевірці умови. Він використовується, коли не відомо, скільки ітерацій буде потрібно для виконання умови. Основна структура циклу while має вигляд:

while (умова) вираз;

Наприклад:

```
$num = 1;
while ($num <= 3 )
 {
       echo $num."<br>";
       $num++;
 }
```

Умова перевіряється на початку кожної ітерації. Якщо вона приймає значення false, блок не буде виконуватися й цикл завершується. Після цього виконується оператор, що випливає за циклом.

Цикли for

За допомогою циклу for такого роду цикл можна записати в більш компактній формі. Основна структура циклу for має вигляд:

```
for( вираз1; умова; вираз2)
вираз3;
```

Наприклад:

```
for ($i=1; $i <= $numnames; $i++)
{
        $temp= "name$i";
        echo $$temp. "
"; //   будь-яка необхідна обробка
}
```

У функціональному змісті цикли while і for ідентичні. Однак цикл for іноді компактніше.

Цикли do...while

Загальна структура оператора do...while має вигляд:

```
Do
вираз;
while ( умова );
```

Цикл do...while відрізняється від циклу while тим, що в ньому умова перевіряється наприкінці. Це означає, що в циклі do...while оператор або блок усередині циклу виконується завжди, не менш одного разу.

```
$num = 100;
do
{
acho $num."
";
}
while ($num < 1);
```

4 Зберігання та відновлення даних

У лекції розглядаються такі питання:

4.1 Огляд обробки файлів.
4.2 Відкривання файлів.
4.3 Читання та запис у файл.
4.4 Інші файлові функції.
4.5 Блокування файлів.

Збереження даних з метою подальшого використання

Існують два основних способи зберігання даних: у двовимірних (звичайних) файлах і в базах даних.

Двовимірний файл може мати безліч форматів, але в загальному випадку під двовимірним (flat) файлом будемо розуміти простий текстовий файл.

Запис у файли й зчитування з них у середовищі PHP зовсім ідентичні реалізації цих завдань у середовищі C. Якщо раніше доводилось програмувати мовою C, цей процес повинен здатися досить знайомим.

4.1 Огляд обробки файлів

Запис даних у файл реалізується в три кроки:

- Відкриття файлу. Якщо файл ще не існує, його буде потрібно створити.
- Запис даних у файл.
- Закриття файлу.

Аналогічно, зчитування даних з файлу також пов'язане з виконанням трьох кроків:

- Відкриття файлу. Якщо файл не може бути відкритий (наприклад, він не існує), ця ситуація повинна бути розпізнана й варто передбачити коректний вихід з її.
- Зчитування даних з файлу.

- Закриття файлу.

При необхідності зчитування даних з файлу можна вибирати, яка частина файлу повинна зчитуватися за один раз.

4.2 Відкриття файлів

Файл є послідовністю байтів, що зберігається на якому-небудь фізичному носії інформації. Кожний файл має абсолютний шлях, по якому визначається його місцезнаходження. Як роздільник шляхи в Windows може використовуватися як прямій (/), так і зворотний (\) слеш. В інших операційних системах використовується тільки прямій слеш.

Відкриття файлів у файловій системі сервера проводиться за допомогою функції fopen:

```
int fopen(string filename, string mode [, int use_include_path])
```

Перший аргумент filename - ім'я файлу або абсолютний шлях до нього. Якщо абсолютний шлях не вказується, то файл повинен перебувати в поточному каталозі.

Другий аргумент mode говорить про те, для яких дій відкривається файл і може приймати наступні значення:

- r (Відкрити файл тільки для читання; після відкриття покажчик файлу встановлюється в початок файлу);
- r+ (Відкрити файл для читання й запису; після відкриття покажчик з файлу встановлюється в початок файлу);
- w (Створити новий порожній файл тільки для запису; якщо файл із таким ім'ям уже є вся інформація в ньому знищується);

- w+ (Створити новий порожній файл для читання запису; якщо файл із таким ім'ям уже є вся інформація в ньому знищується);
- a (Відкрити файл для дозапису; дані будуть записуватися в кінець файлу);
- a+ (Відкрити файл для дозапису й читання даних; дані будуть записуватися в кінець файлу);
- b (Прапор, що вказує на роботу (читання й запис) із двійковим файлом; вказується тільки в Windows).

Третій необов'язковий аргумент use_include_path визначає чи повинні шукатися файли в каталозі include_path. (Параметр include_path установлюється у файлі php.ini).

У випадку вдалого відкриття файлу, функція fopen повертає дескриптор файлу, у випадку невдачі - false. Дескриптор файлу являє собою покажчик на відкритий файл, що використовується операційною системою для підтримки операцій із цим файлом. Повернутий функцією дескриптор файлу необхідно потім указувати у всіх функціях, які надалі будуть працювати із цим файлом.

Код, наведений нижче, відкриває файл C:/WWW/HTML/file.txt для читання:

```
<?
 $file = fopen("c:/www/html/file.txt","r");
 if(!file)
   {
    echo("Помилка відкриття файлу");
   }
?>
```

Відкриття двійкового файлу, приміром , малюнка відбувається в такий же спосіб, тільки із прапором rb.

4.3 Читання з файлів і запис у файли

Читання з файлів

Прочитати рядок з відкритого файлу можна за допомогою функції fread:

```
string fread ( int file, int length )
```

Ця функція повертає рядок довжиною length символів з файлу з дескриптором file.

Приклад (читання з файлу):

```
<?
 $file = fopen("c:/www/html/file.txt","r");
 if(!file)
   {
    echo("Помилка відкриття файлу");
   }
   else
   {
    $buff = fread ($file,100);
    print $buff;
   }
?>
```

Для читання з файлу можна також користуватися функцією fgets:

```
string fgets ( int file, int length)
```

Ця функція читає й повертає рядок довжиною length - 1 байт. Читання припиняється, коли досягнутий новий рядок або кінець файлу. При досягненні кінця файлу, функція повертає порожній рядок.

Для читання файлу з видаленням з нього тегів HTML застосовується функція fgetss:

```
string fgetss (int file, int length [, string allowable_tags])
```

Необов'язковий третій параметр allowable_tags може містити рядок зі списком тегів, які не повинні бути відкинуті, при цьому теги в рядку записуються через кому.

Якщо необхідно записати вміст файлу в масив, застосовується функція file:

```
array file (string filename [, int use_include_path])
```

Функція зчитує файл із ім'ям filename і повертає масив, кожний елемент якого відповідає рядку в прочитаному файлі.

Функцію file варто застосовувати лише для читання невеликих файлів.

Для читання файлів з розширенням *.csv застосовується функція fgetcsv:

```
array fgetcsv ( int file, int length, char delim)
```

Формат CSV є одним з форматів, у якому може зберігати файли MSExcel.

Запис у файли

Запис у файли здійснюється функціями fputs і fwrite, які абсолютно ідентичні:

```
int fputs ( int file, string string [, int length ])
int fwrite ( int file, string string [, int length ])
```

Перший аргумент - дескриптор файлу, у який здійснюється запис. Другий аргумент являє собою рядок, що повинен бути записаний у файл. Третій необов'язковий аргумент задає кількість символів у рядку, які повинні бути записані. Якщо третій аргумент не зазначений, записується весь рядок.

У цьому прикладі у файл "file.txt" записується рядок "Hello, world!"

```
<?
  $file = fopen ("file.txt","r+");
  $str = "Hello, world!";
  if ( !$file )
  {
    echo("Помилка відкриття файлу");
  }
  else
  {
    fputs ( $file, $str);
  }
  fclose ($file);
?>
```

4.4 Інші файлові функції

Закриття файлу

Після того, як Ви завершили використовувати файл, його необхідно закрити. Це здійснюється за допомогою простенької функції fclose():

```
fclose($fp);
```

При цьому, вона повертає значення true у випадку успішного закриття файлу й false, якщо файл не був закритий.

Визначення кінця файлу: feof()

У цьому прикладі використовується цикл while для зчитування з файлу доти, доки не буде досягнутий кінець файлу. Перевірка на наявність кінця файлу здійснюється за допомогою функції feof():

```
while (!feof($fp))
```

Зчитування всього файлу: readfile(), fpassthru(), file()

Замість зчитування по одному рядку з файлу за один прохід можна зчитувати весь файл. Існують три різних способи.

Перший полягає у використанні функції readffle(). Функція readfile() відкриває файл, повторює його вміст у стандартному виводі (вікні браузера), а потім закриває файл. Прототип цієї функції має вигляд:

```
int readfile (string ім'я_файлу, int [use_include_path]) ;
```

Необов'язковий другий параметр указує, чи належний PHP шукати файл у шляху use_include_path, і діє так само, як у функції fopen(). Функція повертає загальну кількість байтів, зчитаних з файлу.

По-друге, можна використовувати функцію fpassthru(). Спочатку необхідно відкрити файл за допомогою функції fopen(). Потім покажчик файлу можна передати у функцію fpassthru(), що завантажить уміст файлу, починаючи з позиції, заданої покажчиком, у стандартний висновок. По завершенні цього процесу функція закриває файл.

Функція fpassthru() повертає значення true, якщо зчитування було виконано успішно, і false - у противному випадку.

Третя можливість зчитування всього файлу - використання функції file(). Ця функція ідентична функції

readfile() за винятком того, що замість повторення файлу в стандартному висновку вона перетворить його в масив.

Зчитування символу: fgetc()

Ще одна можливість обробки файлів - зчитування з файлу по одному символу. Це виконується за допомогою функції fgetc(). Як свій єдиний параметр вона приймає покажчик файлу й повертає наступний символ файлу.
Приклад:

```
while (!feo?($fp))<
$char = ?getc($fp);
if <!feof<$fp))
echo ($char=="\n" ? "<br>": $char);
}
```

Використовуючи функцію fgetc(), цей код зчитує з файлу по одному символі за раз і зберігає його в змінній Schar, поки не буде досягнутий кінець файлу. Потім виконується невелика додаткова обробка з метою заміщення текстових символів кінця рядка \n HTML-роздільниками рядків. Це робиться лише для упорядкування форматування. Оскільки без цього коду браузери не розпізнають нові рядки, весь файл був би виведений у вигляді єдиного рядка.

Зчитування рядків довільної довжини: fread()

Останній спосіб зчитування з файлу, що ми розглянемо - використання функції fread() для зчитування з файлу довільної кількості байтів. Ця функція має наступний прототип:

```
string £read(int fp, int length);
```

Перевірка існування файлу: file_exists()

Якщо необхідно перевірити файл на предмет існування без його відкриття, можна скористатися функцією file_exists().

З'ясування розміру файлу: filesize()

Розмір файлу можна перевірити за допомогою функції filesize(). Вона повертає розмір файлу, виражений у байтах.

Копіювання, перейменування й видалення файлів

Копіювання файлів здійснюється функцією copy:

```
int copy ( string file1, string file2)
```

Перейменування файлу виконується за допомогою функції rename:

```
int rename ( string old, string new)
```

Ця функція перейменовує файл із ім'ям old у файл із ім'ям new.

Видалення файлу здійснюється за допомогою функції unlink:

```
int unlink ( string filename)
```

4.6 Блокування файлів

Уявіть собі ситуацію, коли два клієнти одночасно намагаються замовити товар. (Ця ситуація виникає не настільки вже рідко, особливо коли Web-сайт починає обробляти значні інформаційні потоки.) Що відбудеться, якщо один клієнт викличе функцію fopen() і почне запис, а потім другий клієнт також викличе функцію fopen() і теж

спробує виконати запис? Яким у результаті буде вміст файлу? Чи буде спочатку записане перше замовлення, а потім друге, або навпаки? Чи буде записане перше замовлення або друге? Або ж уміст буде являти собою щось менш корисне, на зразок двох довільно, що чергуються замовлень? Відповідь на ці питання залежить від конкретної використовуваної операційної системи, але часто точно відповісти на них неможливо.

Щоб уникнути подібних проблем використовується блокування файлів. У PHP блокування реалізується за допомогою функції flock(). Ця функція повинна викликатися після відкриття файлу, але перед зчитуванням даних з файлу або їхнім записом у файл.
Прототип функції flock() виглядає так:

```
bool flock (int fp , int operation);
```

У функцію необхідно передати покажчик на відкритий файл і число, що представляє вид необхідного блокування. Функція повертає значення true, якщо блокування було успішно виконане, і false - у противному випадку.
Можливі значення параметра operation:

1 Блокування читання. Це означає, що файл може використовуватися разом з іншими читаючими додатками.
2 Блокування запису. Це монопольний режим. Файл не доступний для спільного використання.
3 Зняття існуючого блокування.
4 Додавання до поточного значення параметра operation запобігає іншим спробам блокування під час виконання поточного блокування.

Якщо вирішено використовувати функцію flock(), її варто включити в усі сценарії, у яких використовується

даний файл; у противному випадку її застосування позбавлене змісту.
Приклад:

```
$fp = fopen("$DOCUMENT_ROOT/../orders/orders.txt", "a", 1);
flock($fp, 2); // блокування файлу для запису
fwrite($fp, $outputstring);
flock($fp, 3); // зняття блокування запису
fclose($fp);
```

Варто також додати блокування у файл vieworders.php:

```
$fp = fopen("$DOCUMENT_ROOT/../orders/orders.txt", "r");
flock($fp, 1); // блокування файлу для читання
// читання з файлу
flock($fp, 3); // зняття блокування запису
fclose($fp);
```

Що відбудеться, якщо два сценарії спробують одночасно виконати блокування? Це привело б до конфлікту, коли процеси суперничають за установку блокування, але не відомо, якому з них це вдасться, що, у свою чергу, могло б породити нові проблеми.

5 Використання масивів

У лекції розглядаються такі питання:

5.1 Числово-індексовані масиви.
5.2 Асоціативні масиви.
5.3 Багатовимірні масиви.
5.4 Сортування масивів.
5.5 Інші функції обробки масивів.

Використання масивів

Масив - це змінна, у якій зберігається набір, або послідовність, значень. Один масив може містити багато елементів. Кожний елемент може містити єдине значення, таке як текст або число, або інший масив.

Масив, що містить інші масиви, називається багатомірним масивом. PHP підтримує як чисельно індексовані, так і асоціативні масиви. Асоціативні масиви дозволяють використовувати значення, які знаходять більше широке застосування, чим індекси. Замість чисельних індексів кожний елемент такого масиву може мати слова або іншу корисну інформацію.

Скалярна змінна - це іменована комірка пам'яті, у якій зберігається значення; за аналогією, масив - це іменована комірка пам'яті, у якій зберігається набір значень, що дозволяє групувати звичайні скалярні значення.

Значення, що зберігаються в масиві, називаються елементами масиву. Кожний елемент масиву має пов'язаний з ним індекс (називаний також ключем), що використовується для доступу до елемента.

5.1 Чисельно індексовані масиви

Для створення масиву можна використовувати наступний рядок коду:

```
$products = array( "Tires", "Oil", "Spark Plugs" );
```

У результаті створюється масив products, що містить три заданих значення: "Tires", "Oil" і "Spark Plugs". Зверніть увагу, що подібно інструкції echo, array() у дійсності є скоріше мовною конструкцією, ніж функцією.

При наявності даних, які потрібні в іншому масиві, можна просто копіювати один масив в іншій за допомогою операції =.

Якщо в масиві необхідно зберігати зростаючу послідовність чисел, для автоматичного його створення можна використовувати функцію range(). Наступний рядок коду створює масив numbers, що містить елементи, які є числами від 1 до 10:

```
$numbers = range(1,10) ;
```

Якщо інформація зберігається у файлі на диску, уміст масиву можна завантажити безпосередньо з файлу.

Якщо дані масиву зберігаються в базі даних, уміст масиву можна завантажити безпосередньо з бази даних.

5.2 Асоціативні масиви

При створенні масиву ми надаємо PHP можливість привласнити кожному елементу індекс, обумовлений за замовчуванням. Це означає, що перший доданий елемент став 0 елементом, другий - 1 і т.д. PHP підтримує також асоціативні масиви. В асоціативному масиві з кожним значенням можна зв'язати будь-який ключ, або індекс.

5.3 Багатомірні масиви

Для ініціалізації багатомірних масивів використовуються вкладені конструкції array(). Обхід багатомірних масивів досягається за допомогою вкладених

циклів. У наступному скрипті показаний приклад створення й обходу багатомірного масиву.
Приклад:

ї

```
<?
  $ship = array(
    "Пасажирські кораблі" =>
array("Київ","Вітрило","Європа"),
    "Військові кораблі" =>
array("Адмірал","Капітан","Шторм"),
     );
  foreach($ship as $key => $type)
  {
    echo(
    "<h2>$key</h2>\n"."<ul>\n");
    foreach($type as $ship)
    {
      echo("\t<li>$ship</li>\n");
    }
  }
  echo("</ul>\n");
?>
```

Результат виконання цього скрипта::

Пасажирські кораблі

- Київ
- Вітрило
- Європа

Військові кораблі

- Адмірал
- Капітан
- Шторм

Ініціалізація масивів

В PHP існує 2 методи ініціалізації масивів. Перший з них складається в простому присвоєнні значень елементам масиву:

```
<?
  $car[] = "passenger car";
  $car[] = " land-rover";
  echo($car[1]); // виводить " land-rover"
?>
```

Індекс масиву можна вказати явно: [0], [1] і т.д.

Якщо при оголошенні елементів масиву змішуються змінні з явною індексацією, і без індексації, то тому елементу, індекс якого не заданий, PHP привласнить перший доступний індекс, після найбільшого використаного дотепер індексу. Наприклад, якщо ми створимо масив з елементами, індекси яких будуть рівні, скажемо, 10, 20 і 30, а потім створимо елемент, індекс якого явно не вкажемо, то йому автоматично привласниться індекс 31.

Альтернативний спосіб визначення масивів складається у використанні конструкції array():

```
<?
  $car = array("passenger car"," land-rover");
  echo($car[1]); // виводить " land-rover"
?>
```

Для явної вказівки індексів у цьому випадку застосовується оператор =>:

```
<?
  $car = array("passenger car", 5 => " land-rover",
    "station-wagon","victoria");
  echo($car[0]); echo("<br>"); // виводить "passenger car"
```

```
  echo($car[5]); echo("<br>"); // виводить " land-rover"
  echo($car[6]); echo("<br>"); // виводить "station-wagon"
  echo($car[7]); // виводить "victoria"
?>
```

Індексами масиву можуть бути й рядки.

5.4 Сортування масивів

sort() - функція сортування масиву по зростанню. Синтаксис:

```
void sort(array array [, int sort_flags])
```

Функція сортує масив array по зростанню. Необов'язковий аргумент sort_flags указує як саме повинні сортуватися елементи (задає прапори сортування). Припустимими значеннями цього аргументу є наступні:

- SORT_REGULAR - задає нормальне порівняння елементів (порівнює елементи "як є")
- SORT_NUMERI - порівнює елементи як числа
- SORT_STRING - порівнює елементи як рядка

Приклад:

```
<?
  $arr = array("2", "1", "4", "3","5");
  sort($arr);
  for($i=0; $i < count($arr); $i++)
  {
    echo ("$i:$arr[$i]  ");
  }
  // виводить "0:1 1:2 2:3 3:4 4:5"
?>
```

Результат:
0:1 1:2 2:3 3:4 4:5

rsort() - сортування масиву по убуванню.
Синтаксис:

```
void rsort(array arr [, int sort_flags])
```

Аналогічна функції sort(), тільки сортує по убуванню.

asort() - сортування асоціативного масиву по зростанню.
Синтаксис:

```
void asort(array arr [, int sort_flags])
```

Функція asort() сортує масив arr так, щоб його значення йшли в алфавітному (якщо це рядки) або зростаючому (для чисел) порядку. Важлива відмінність цієї функції від функції sort() полягає в тому, що при застосуванні функції asort() зберігаються зв'язки між ключами й відповідними їм значеннями, чого немає у функції sort() (там ці зв'язки попросту розриваються).
Приклад:

```
<?
  $arr = array("a" =>"one","b" => "two","c" => "three","d"
=> "four");
  asort($arr);
  foreach($arr as $key => $val)
  {
    echo (" $key => $val ");
  }
?>
```

Результат:
d => four a => one c => three b => two

Значення прапорів сортування sort_flags наведені в описі функції sort().

arsort() - сортування асоціативного масиву по убуванню.
Синтаксис:

```
void arsort(array arr [, int sort_flags])
```

Ця функція аналогічна функції asort(), тільки вона впорядковує масив по убуванню.

ksort() - сортування масиву по зростанню ключів.
Синтаксис:

```
int ksort(array arr [, int sort_flags])
```

У цій функції сортування здійснюється не за значеннями, а по ключах у порядку їхнього зростання.

Якщо у попередньому прикладі замінити asort на ksort, то результат буде:
a => one b => two c => three d => four

krsort() - сортування масиву по убуванню індексів.
Синтаксис:

```
int krsort(array arr [, int sort_flags])
```

Те ж саме, що й функція ksort(), тільки впорядковує масив по ключах у зворотному порядку (по убуванню).

array_reverse() - розміщення елементів масиву у зворотному порядку.
Синтаксис:

array array_reverse(array arr [, bool preserve_keys])

Функція array_reverse() повертає масив, елементи якого випливають у зворотному порядку щодо масиву arr, переданого в параметрі. При цьому зв'язки між ключами й значеннями зберігаються. Можна ще необов'язковий параметр preserve_keys зробити true, тоді у зворотному порядку переставляться ще й ключі.

shuffle() - перемішування елементів масиву випадковим чином.
Синтаксис:

void shuffle(array arr)

natsort() - виконує "природне" сортування масиву.
Синтаксис:

void natsort(array arr)

Під природним сортуванням розуміється сортування таким чином, коли елементи того, що сортується розташовуються в "зрозумілому" для людини порядку.
Приклад:

```
<?
  $array1 = $array2 = array("pict10.gif", "pict2.gif",
"pict20.gif", "pict1.gif");
  echo ("звичайне сортування:"); echo ("<br>");
  sort($array1);
```

```
  print_r($array1);
  echo ("<br>"); echo ("природне сортування:"); echo
("<br>");
  natsort($array2);
  print_r($array2);
?>
```

Результат:

звичайне сортування:
Array ([0] => pict1.gif [1] => pict10.gif [2] => pict2.gif [3] => pict20.gif)
природне сортування:
Array ([3] => pict1.gif [1] => pict2.gif [0] => pict10.gif [2] => pict20.gif)

5.5 Інші функції обробки масивів

Функція count()
Синтаксис:

```
int count(mixed var)
```

Ця функція приймає як аргумент масив і повертає кількість елементів у ньому.

Функція in_array()
Синтаксис:

```
boolean in_array(mixed needle, array haystack [, bool strict])
```

Ця функція шукає в масиві haystack значення needle і повертає true якщо воно знайдено й false у противному випадку.

Функція reset()
Синтаксис:

mixed reset(array array)

Функція reset() установлює покажчик масиву на перший елемент і повертає значення першого елемента масиву.

Робота з курсором (покажчик) масиву

reset() - ця функція робить скидання курсору масиву, тобто встановлює внутрішній курсор масиву на початок масиву й повертає значення першого елемента.

end() - виконує дію, зворотна функції reset(), переносить курсор у кінець масиву.
Синтаксис:

mixed end(array array arr)

next() - робить перенос курсору масиву вперед на одну позицію.
Синтаксис:

mixed next(array array arr)

Тобто ця функція переміщає курсор масиву на наступний елемент, при цьому зі значення елемента, на якому перебував курсор до переміщення

prev() - робить перенос курсору назад на одну позицію. Синтаксис і робота функції повністю аналогічні функції next().

Синтаксис:

```
mixed prev(array array arr)
```

current() - для визначення поточного елемента масиву, без зміни положення курсору, використовується функція.
Синтаксис:

```
mixed current(array array arr)
```

Функція current() повертає значення елемента, на якому в цей момент перебуває курсор масиву, при цьому не зрушуючи курсор. У тому випадку, якщо курсор виявився за межами масиву, або масив складається з порожніх елементів, функція поверне false.

Повним синонімом функції current() є функція pos().

key() - повертає індекс поточного елемента масиву.
Синтаксис:

```
mixed key(array array arr)
```

Функція each() повертає пари "індекс - значення" поточного елемента масиву й зрушує курсор масиву на наступний елемент. При цьому, як видно, функція повертає масив, причому він має чотири елементи:

1. [1] => "значення"
2. [value] => "значення"
3. [0] => індекс
4. [key] => індекс

Синтаксис:

```
array each(array array arr)
```

Якщо курсор досяг кінця масиву, функція повертає false.

array_walk() - досить важлива функція, що дозволяє застосовувати користувальницьку функцію до кожного елемента масиву.
Синтаксис:

bool array_walk(array arr, callback func [, mixed userdata])

Як видно із синтаксису цієї функції, вона застосовує користувальницьку функцію func до кожного елемента масиву arr. У користувальницьку функцію передаються два або три аргументи: значення поточного елемента, його індекс і аргумент userdata. Останній аргумент є необов'язковим. Помітимо, що у випадку, якщо func вимагає більше трьох аргументів, при кожному її виклику буде видаватися попередження, і, щоб вони не видавалися, потрібно поставити знак "@" перед функцією array_walk().

6 Багатократне використання коду та створення функцій

У лекції розглядаються такі питання:

6.1 Шаблони Web-сайту.

6.2 Завантаження шаблонів.

6.3 Використання функцій у PHP.

6.4 Область дії.

6.1 Шаблони Web- Сайту

Web - сайт - це набір Web-сторінок і файлів зв'язаних між собою гіперпосиланнями. Web-сторінки або гіпертекстові документи являють собою текст, у якому втримуються спеціальні команди, називані тегами (tags). Ці теги забезпечують форматування елементів сторінки й дозволяють розміщати на ній графічні об'єкти, малюнки, гіперпосилання й т.д.

Web-сторінки створюються за допомогою спеціальної мови HTML. Можливість роботи з Web-сторінками забезпечує один з видів сервісу Internet, що називається World Wide Web або скорочено WWW. В основу World Wide Web був покладений протокол прикладного рівня http, що забезпечує прийом і передачу Web- сторінок.

Шаблони можна розглядати як "розширення" програмного коду. Шаблони не тільки автоматизують стомлюючий процес кодування, але й забезпечують структурний розподіл проекту в робочих групах. Роль такого розподілу зростає зі збільшенням обсягів проекту й чисельності груп, а також з ускладненням архітектури проекту, причому не тільки на стадії програмування, але й при наступному супроводі програми.

Існує два різних підходи до створення шаблонів PHP:

- впровадження HTML у код PHP;
- включення файлів у сторінку.

Хоча перша схема більше зрозуміла й простіше реалізується, вона також більшою мірою обмежує свободу дій. Головна проблема полягає в тім, що код PHP змішується з компонентами HTML, що утворять макет сторінки.

Друга схема в багатьох ситуаціях виявляється набагато зручніше першою. Проте, хоча структура "заголовок - основна частина - колонтитул" добре підходить для структурування відносно малих сайтів із чітко визначеним форматом, зі збільшенням обсягів і складності проекту ці обмеження проявляються усе помітніше. У цій схемі розділяються два головних компоненти web-додатка: дизайн і програмування. Подібний розподіл забезпечує можливість паралельної розробки (web-дизайн і програмування) без необхідності постійної координації протягом усього робочого циклу. Більш того, воно дозволяє в майбутньому модифікувати один компонент, не впливаючи на роботу іншого.

Головною метою при розробці систем шаблонів є фактичне відділення дизайну від функціональних можливостей. Дійсно, ця система й створюється для того, щоб програмісти й дизайнери могли незалежно трудитися над своїми аспектами додатка, не заважаючи роботі іншої групи.

Приклад шаблона

```
<html>
<head>
<title>:::::{page_title}:::::</title>
</head>
```

```
<body bgcolor="{bg_color}">
Welcome to your default home page. {user_name}!<br>
You have 5 MB and 3 email addresses at your disposal.<br>
Have fun!
</body>
</html>
```

Зверніть увагу на три рядки (page_title, bg_color і userjiame), ув'язнені у фігурні дужки ({ }). Фігурні дужки мають особливий сенс при обробці шаблонів - укладений в них рядок інтерпретується як ім'я змінної, замість якого підставляється її значення. Дизайнер будує сторінку за своїм розсудом; усе, що від нього буде потрібно, - включати у відповідні місця документа ці ключові рядки. Звичайно, програмісти й дизайнери повинні заздалегідь погодити імена всіх змінних!

Як працює схема: насамперед, можливо, нам доведеться одночасно працювати з декількома шаблонами, що володіють тими самими загальними атрибутами. У таких ситуаціях застосування технології об'єктно-орієнтованого програмування (ООП) виявляється особливо ефективним. Із цієї причини всі функції побудови й виконання операцій із шаблонами будуть оформлені у вигляді методів класу. Визначення класу починається так:

```
class template {
VAR $files = array( );
VAR $variables = array( );
VAR $opening_escape = '{';
VAR $closing_escape = '}';
```

У масиві $files зберігаються ідентифікатори файлів і вміст кожного файлу. Атрибут $variables являє собою двомірний масив для зберігання файлового ідентифікатора (ключа) і всіх відповідних змінних, оброблюваних у схемі

шаблонів. Нарешті, атрибути $opening_escape і $closing_escape задають обмежники для частин шаблона, які повинні замінятися системою. Як було показано в прикладі, у наших прикладах як обмежники будуть використовуватися фігурні дужки ({ }).

Кожний метод класу вирішує конкретне завдання, що відповідає тій або інший операції в процесі обробки шаблона. На найпростішому рівні цей процес можна розділити на чотири стадії.

- Реєстрація файлів - реєстрація всіх файлів, оброблюваних сценаріями шаблонів.
- Реєстрація змінних - реєстрація всіх змінних, які повинні замінятися своїми значеннями в зареєстрованих файлах.
- Обробка файлів - заміна всіх змінних, що перебувають між обмежниками, у зареєстрованих файлах.
- Висновок файлу - висновок оброблених зареєстрованих файлів у браузері.

6.2 Завантаження шаблонів

Один з найважливіших етапів створення сайту є розміщення його на хостингу. Веб-хостинг - це місце для розміщення сайту на сервері в мережі Internet, що надає доступ до Web-сторінок відвідувачам сайту.

Для розміщення сайту на хостингу необхідно зареєструватися на одному із серверів, що надає послуги по розміщенню. Інтернет-адреса або доменна адреса сайту залежить від того, який Ви рівень домена придбали. При роботі в Internet використовуються не доменні імена, а універсальні покажчики ресурсів, називані URL (Universal Resource Locator).

Для завантаження файлів сайту на сервер можна використовувати файловий менеджер (команду завантажити) з розділу керування сайтом на сервері, на якому Ви розміщаєте сайт. Спочатку за допомогою файлового менеджера створюють на сервері директорію (папку), у яку будете поміщати файли або виберіть готову папку на сервері.

Завантажити файли на сервер можна й за допомогою браузера (Internet Explorer) по протоколу ftp, наприклад ftp://ftp.lessons-tva.info/. Далі з'явиться діалогове вікно із запитом увести пароль, після уведення пароля й натискання кнопки "ОК" у вікні перегляду будуть відображені всі Ваші директорії на сервері.

Далі відкрийте папку, де будуть розміщені файли сайту й скопіюйте їх туди одним з методів. Адреса FTP-сервера вказується в персональних даних, які ви одержите після реєстрації на сервері. Щоб з'єднання по FTP-протоколу відбувалося швидше необхідно попередньо ввійти у свій аккаунт по протоколу http.

Але найкраще для завантаження файлів використовувати WC або Total Commander для цього в меню "МЕРЕЖА" вибрати команду "Нове FTP - з'єднання" і в діалоговому вікні, що з'явилося, "З'єднатися" увести FTP-Адресу (наприклад, ftp://ftp.lessons-tva.info/). Потім по запиту ввести пароль і на одній з панелей WC з папки, розташованій на сервері, в одну з них необхідно помістити (скопіювати) файли.

Далі Ви привласніть ім'я цьому з'єднанню й збережіть його. Це ім'я буде поміщено в опцію "З'єднатися з FTP- Сервером" у меню "МЕРЕЖА". При повторному з'єднанні (наприклад, при відновленні сторінок) Ви вибираєте це ім'я, далі на запит уводите пароль і з'єднуєтеся з директорією, у якій перебувають сторінки й файли Вашого сайту.

6.3 Використання функцій у PHP

Під функцією розуміють незалежний модуль коду, що встановлює інтерфейс виклики, виконує певне завдання й, необов'язково, повертає результат.

Функції користувача в PHP

Підпрограма - це спеціальним образом оформлений фрагмент програми, до якого можна звернутися з будь-якого місця усередині програми. Підпрограми істотно спрощують життя програмістам, поліпшуючи читабельність вихідного коду, а також скорочуючи його, оскільки окремі фрагменти коду не потрібно писати кілька разів.

В PHP такими підпрограмами є користувальницькі функції.

Особливості користувальницьких функцій в PHP:

- Доступні параметри за замовчуванням. Є можливість викликати ту саму функцію зі змінним числом параметрів;
- Користувальницькі функції можуть повертати будь-який тип;
- Область видимості змінних усередині функції є ієрархічною (деревоподібною);
- Є можливість змінювати змінні, передані як аргумент.

При використанні користувальницьких функцій встає питання про область видимості змінних.

Змінні по області видимості підрозділяються на глобальні й локальні.

Глобальні змінні - це змінні, які доступні всій програмі, включаючи підпрограми (функції).

Локальні змінні - змінні, визначені усередині підпрограми (функції). Вони доступні тільки усередині функції, у якій вони визначені.

Для PHP всі оголошені й використовувані у функції змінні за замовчуванням локальні для функції.

Якщо ви в тілі користувальницької функції будете використовувати змінну з ім'ям, ідентичним ім'я глобальної змінної (що перебуває поза користувальницькою функцією), то ніякого відношення до глобальної змінної ця локальний змінна мати не буде. У даній ситуації в користувальницькій функції буде створена локальна змінна з ім'ям, ідентичним імені глобальної змінної, але доступна дана локальна змінна буде тільки усередині цієї користувальницької функції.

Крім локальних і глобальних змінних, в PHP існує ще один тип змінних - статичні змінні.

Якщо в тілі користувальницької функції оголошена статична змінна, то компілятор не буде її видаляти після завершення роботи функції. Приклад роботи користувальницької функції, що містить статичні змінні:

```
<?php
  function funct()
  {
    static $a;
    $a++;
    echo "$a";
  }
  for ($i = 0; $i++<10;) funct();
?>
```

Даний сценарій виводить рядок:

1 2 3 4 5 6 7 8 9 10

Якщо ми видалимо інструкцію static, буде виведений рядок:

1 1 1 1 1 1 1 1 1 1

Це пов'язане з тим, що змінна $a буде віддалятися при завершенні роботи функції й обнулятись при кожному її виклику. Змінна $a інкрементується відразу після обнуління, а тільки потім виводиться.

Створення функцій користувача

Функція користувача може бути оголошена в будь-якій частині програми (скрипта), до місця її першого використання. І не потрібно ніякого попереднього оголошення.

Синтаксис оголошення функцій наступний:

```
function Ім'я
(аргумент1[=значення1],...,аргумент1[=значення1])
{
тіло_функції
}
```

Оголошення функції починається службовим словом function, потім йде ім'я функції, після імені функції - список аргументів у дужках. Тіло функції заключене у фігурні дужки й може містити будь-яку кількість операторів.

Вимоги, пропоновані до імен функцій:

- Імена функцій можуть містити російські букви, але давати функціям імена, що складаються з російських букв не рекомендується;
- Імена функцій не повинні містити пробілів;
- кожної користувальницької функції повинне бути унікальним. При цьому, необхідно пам'ятати, що регістр при оголошенні функцій і звертанні до них не враховується. Тобто , наприклад, функції funct() і FUNCT() мають однакові імена;

- Функціям можна давати такі ж імена, як і змінним, тільки без знака $ на початку імен.

Типи значень, що повертаються користувальницькими функціями, можуть бути будь-якими. Для передачі результату роботи користувальницьких функцій в основну програму (скрипт) використовується конструкція return. Якщо функція нічого не повертає, конструкцію return не вказують. Конструкція return може повертати все, що завгодно, у тому числі й масиви.

Приведемо приклади використання функцій користувача:

```
<?php
function funct() {
$number = 777;
 return $number;
}
$a = funct();
echo $a;
?>
```

У розглянутому прикладі функція funct повертає за допомогою інструкції return число 777. Повернуте функцією значення привласнюється глобальній змінній $a, а потім оператор echo виводить значення змінної $a у браузер. У результаті ми побачимо в браузері число 777.

Передача аргументів функціям користувача

При оголошенні функції можна вказати список параметрів, які можуть передаватися функції, наприклад:

```
<?php
function funct($a, $b, /* ..., */ $z) { ... };
?>
```

При виклику функції funct() потрібно вказати всі передані параметри, оскільки вони є обов'язковими. В PHP користувальницькі функції можуть мати необов'язкові параметри або параметрами за замовчуванням, але про це пізніше.

Передача аргументів по посиланню

Якщо ви хочете, щоб аргумент передавався по посиланню, ви повинні вказати амперсанд (&) перед ім'ям аргументу в описі функції:

```
<?php
function funct(&$string)
{
    $string .= 'а цей всередині.';
}
$str = 'Цей рядок за межами функції, ';
funct($str);
echo $str;      // Виведе 'Цей рядок за межами функції, а
цей всередині.'
?>
```

Параметри за замовчуванням

В PHP функції можуть повертати будь-які значення залежно від переданих їм параметрів.

```
<?php
function makecup($type = "Чаю")
{
    return "Зробіть чашечку $type.\n";
}
echo makecup();
echo makecup("Кави");
?>
```

Результат роботи наведеного скрипта буде таким:

Зробіть чашечку Чаю
Зробіть чашечку Кави

Значення за замовчуванням повинне бути константним виразом.

6.4 Область дії

Діапазон дії змінних управляє тим, де змінна видима й застосовна. У різних мовах програмування діють різні правила, що встановлюють діапазон дії змінних. У PHP діють дуже прості правила:

- Змінні, які оголошені усередині функції, діють в області від оператора, у якому вони оголошені до закриваючої дужки наприкінці функції. Ця область називається областю функції, а такі змінні - локальними змінними.
- Змінні, які оголошені поза функцією, діють в області від оператора, у якому вони оголошені до кінця файлу, але не усередині функцій. Ця область називається глобальною областю, а такі змінні - глобальними змінними.
- Використання операторів require() і include() не впливає на область дії змінних. Якщо оператор використовується усередині функції, застосовується область функції. Якщо він використовується не усередині функції, застосовується глобальна область.
- Ключове слово global може використовуватися для вказівки вручну того, що змінна, котра визначена або використовується усередині функції, буде мати глобальну область дії.

- Змінні можуть бути вручну вилучені за допомогою функції unset($variable_name). Якщо змінна вилучена, вона більше не перебуває в області дії.

Наступні приклади допоможуть розібратися з описаними концепціями. Наступний код не створює ніякого виводу. У ньому оголошується змінна $var усередині функції fh(). Оскільки ця функція оголошується усередині функції, вона має область дії функції й існує від місця її оголошення до кінця функції. При новому звертанні до $var поза функцією, створюється нова змінна $var. Ця нова змінна має глобальну область дії й буде видима до кінця файлу.

```
function fn()
{
$var = "contents";
}
echo $var ;
```

Наступний приклад протилежний попередньому. Ми повідомляємо змінну зовні функції, а потім намагаємося її використовувати усередині функції.

```
function fn()
{
echo "inside the function, \$var = ".$var."<br>";
$var = "contents2";
echo "inside the function, \$var = ".$var."<br>";
}
$var = "contents 1";
fn();
echo "outside the function, \$var = ".$var."<br>";
```

Цей код створить наступний висновок:

```
inside the function, $var =
inside the function, $var = contents 2
outside the function, $var = contents 1
```

Функції не виконуються доти, поки вони не будуть викликані, тому першим виконуваним оператором є $var = "contents 1"; Він створює змінну $var, що має глобальну область дії й вміст "contents 1". Наступний виконуваний оператор - звертання до функції fh(). Рядки усередині оператора виконуються по черзі. Перший рядок у функції звертається до змінного $var. Коли цей рядок виконується, вона не може бачити попередню створену нами змінну $var, тому вона створює нову змінну, що має область функції, і повторює її у висновку. У результаті створюється перший рядок висновку.

Наступний рядок усередині функції встановлює вміст змінної $var рівним "contents 2". Оскільки дії виконуються усередині функції, цей рядок змінює значення локальної змінної Svar, а не глобальної. Другий рядок висновку підтверджує виконання цієї зміни.

На цьому виконання функції завершується, тому виконується заключний рядок сценарію. Цей оператор echo демонструє, що значення глобальної змінної не змінилося.

7 Об'єктно-орієнтоване програмування на PHP

У лекції розглядаються такі питання:

7.1 Принципи ООП.
7.2 Класи та об'єкти.
7.3 Створення класів та екземплярів класів
7.4 Написання коду класу.
7.5 Поліморфізм.
7.6 Спадкування.

Об'єктно-орієнтоване програмування на PHP

ООП (Об’єктно-орієнтоване програмування) - це підхід до розробки щодо більших проектів з довгим часом життя. Мова PHP стає усе більш популярною і використовується в багатьох професійних проектах. Через це, техніка ООП все частіше використовується в PHP-проектах.

Немає особливого сенсу використовувати ООП для дрібних проектів, які будуть існувати короткий проміжок часу. Однак якщо ви плануєте в майбутньому додати нові функції й т.п., то ООП заощадить ваш час.

7.1 Принципи ООП

Об’єктно-орієнтоване програмування засноване на трьох принципах:

- Інкапсуляції;
- Поліморфізмі;
- Спадкуванні.

Інкапсуляція - це механізм, що поєднує дані й обробляє їхній код як єдине ціле.

Інкапсуляцією називається включення різних дрібних елементів у більший об'єкт, у результаті чого

програміст працює безпосередньо із цим об'єктом. Це приводить до спрощення програми, оскільки з неї виключаються другорядні деталі.

Поліморфізм дозволяє використовувати ті самі імена для схожих, але технічно різних завдань. Головним у поліморфізмі є те, що він дозволяє маніпулювати об'єктами шляхом створення стандартних інтерфейсів для схожих дій. Поліморфізм значно полегшує написання складних програм.

Спадкування дозволяє одному об'єкту здобувати властивості іншого об'єкта, не плутайте з копіюванням об'єктів. При копіюванні створюється точна копія об'єкта, а при спадкуванні точна копія доповнюється унікальними властивостями, які характерні тільки для похідного об'єкта.

7.2 Класи й об'єкти

Клас - це базове поняття в ООП. Класи утворюють синтаксичну базу ООП. Їх можна розглядати як свого роду "контейнери" для логічно зв'язаних даних і функцій (звичайно називаних методами). Якщо сказати простіше, то клас - це своєрідний тип даних.

Екземпляр класу - це об'єкт. Об'єкт - це сукупність даних (властивостей) і функцій (методів) для їхньої обробки. Властивості й методи називаються членами класу. Взагалі, об'єктом є все те, що підтримує інкапсуляцію.

Якщо клас можна розглядати як тип даних, то об'єкт - як змінну (за аналогією). Скрипт може одночасно працювати з декількома об'єктами одного класу, як з декількома змінними.

Усередині об'єкта дані й код (члени класу) можуть бути або відкриті, або ні. Відкриті дані й члени класу є доступними для інших частин програми, які не є частиною

об'єкта. А от закриті дані й члени класу доступні тільки усередині цього об'єкта.

7.3 Створення класів та екземплярів класів

Опис класів в PHP починаються службовим словом class:

```
class Ім'я_класу {
// опис членів класу - властивостей і методів для їхньої обробки
}
```

Для оголошення об'єкта необхідно використовувати оператор new:

```
Об'єкт = new Ім'я_класу;
```

Дані описуються за допомогою службового слова var. Метод описується так само, як і звичайна користувальницька функція. Методу також можна передавати параметри.

За загальноприйнятими правилами імена класів ООП починаються із прописної букви, а всі слова в іменах методів, крім першого, починаються із прописних букв (перше слово починається з малої літери).

Приклад класу на PHP:

```
<?php
// Створюємо новий клас Coor:
class Coor {
// дані (властивості):
var $name;
var $addr;

// методи:
```

```
 function Name() {
 echo "<h3>John</h3>";
 }
}
// Створюємо об'єкт класу Coor:
$object = new Coor;
?>
```

Доступ до класів і об'єктів в PHP

Ми розглянули, яким чином описуються класи й створюються об'єкти. Тепер нам необхідно одержати доступ до членів класу, для цього в PHP призначений оператор ->. Приведемо приклад:

```
<?php
…….
// Одержуємо доступ до членів класу:
$object->name = "Alex";
echo $object->name;
// Виводить 'Alex'
// А тепер одержимо доступ до методу класу (фактично,
до функції усередині класу):
$object->Getname();
// Виводить 'John' заголовними буквами
?>
```

Щоб одержати доступ до членів класу усередині класу, необхідно використовувати покажчик $this, котрий завжди ставиться до поточного об'єкта. Модифікований метод Getname():

```
function Getname() {
echo $this->name;
}
```

У такий же спосіб, можна написати метод Setname():

```
function Setname($name) {
$this->name = $name;
}
```

Тепер для зміни ім'я можна використовувати метод Setname():

```
$object->Setname("Peter");
$object->Getname();
```

А от і повний лістинг коду:

```
<?php
// Створюємо новий клас Coor:
class Coor {
// дані (властивості):
var $name;

// методи:
 function Getname() {
 echo $this->name;
 }

 function Setname($name) {
 $this->name = $name;
 }
}
// Створюємо об'єкт класу Coor:
$object = new Coor;
// Тепер для зміни ім'я використовуємо метод Setname():
$object->Setname("Nick");
// А для доступу, як і колись, Getname():
$object->Getname();
// Сценарій виводить 'Nick'
?>
```

Покажчик $this можна також використовувати для доступу до методів, а не тільки для доступу до даних.

7.4 Написання коду класу

Конструктори

Конструктор являє собою метод, що задає значення деяких властивостей (а також може викликати інші методи). Конструктори викликаються автоматично при створенні нових об'єктів. Щоб це стало можливим, ім'я методу-конструктора повинне збігатися з ім'ям класу, у якому він утримується. Приклад конструктора:

```
<?
class Webpage {
var $bgcolor;
 function Webpage($color) {
 $this->bgcolor = $color;
 }
}

// Викликати конструктор класу Webpage
$page = new Webpage("brown");
?>
```

Можна викликати конструктор, що просто створює об'єкт, але не ініціалізує його властивості:

```
$page = new Webpage;
```

Об'єкт можна створити за допомогою конструктора, визначеного в класі:

```
$page = new Webpage("brown");
```

Деструктори

У PHP відсутня безпосередня підтримка деструкторів. Проте, ви можете легко імітувати роботу деструктора, викликаючи функцію PHP unset(). Ця функція знищує вміст змінної й повертає займані нею ресурси системі. З об'єктами unset() працює так само, як і зі змінними. Допустимо, ви працюєте з об'єктом $Webpage. Після завершення роботи із цим конкретним об'єктом викликається функція:

```
unset($Webpage);
```

Ця команда видаляє з пам'яті весь уміст $Webpage. Необхідність у виклику деструкторів виникає лише при роботі з об'єктами, що використовують великий обсяг ресурсів, оскільки всі змінні й об'єкти автоматично знищуються по завершенні сценарію.

Звертання до елементів класів

Звертання до елементів класів здійснюється за допомогою оператора :: "подвійна двокрапка". Використовуючи "подвійну двокрапку", можна звертатися до методів класів.

При звертанні до методів класів, програміст повинен використовувати імена цих класів.

```
<?php
class A {
    function example() {
        echo "Це первісна функція A::example().<br>";
    }
}

class B extends A {
```

```
    function example() {
        echo "Це перепевна функція B::example().<br>";
        A::example();
    }
}

// Не потрібно створювати об'єкт класу A.
// Виводить наступне:
// Це первісна функція A::example().
A::example();

// Створюємо об'єкт класу B.
$b = new B;

// Виводить наступне:
//   Це перепевна функція B::example().
//   Це первісна функція A::example().
$ b-b->example();
?>
```

7.5 Поліморфізм

Поліморфізм є наслідком ідеї спадкування. Поліморфність класу - це властивість базового класу використовувати функції похідних класів, навіть якщо на момент визначення ще невідомо, який саме клас буде включати його в якості базового й, тим самим, ставати від нього похідним.

Розглянемо властивість поліморфності класів на основі наступного приклада:

```
<?php
class A {
// Виводить, функція якого класу була викликана
function Test() { echo "Test from A\n"; }
// Тестова функція - просто переадресує на Test()
```

```
function Call() { Test(); }
}
class B extends A {
// Функція Test() для класу B
function Test() { echo "Test from B\n"; }
}
$a=new A();
$b=new B();
?>
```

Використовуємо наступні команди:

```
$ a-a->Call(); // виводить "Test from A"
$ b-b->Test(); // виводить "Test from B"
$ b-b->Call(); // Увага! Виводить "Test from B"!
```

Зверніть увагу на останній рядок: всупереч очікуванням, викликається не функція Test() із класу A, а функція із класу B! Складається враження, що Test() з B просто перевизначила функцію Test() з A. Так воно насправді і є. Функція, переобумовлена в похідному класі, називається віртуальної.

7.6 Спадкування

Спадкування - це не просте створення точної копії класу, а розширення вже існуючого класу, щоб нащадок міг виконувати які-небудь нові, характерні тільки йому функції.

Отже, нехай у нас є деякий клас A з певними властивостями й методами. Але те, що цей клас робить, нас не зовсім улаштовує - наприклад, нехай він виконує більшість функцій, що нам необхідні, але не реалізує деяких інших. Задамося метою створити новий клас B, як

би "розширював" можливості класу А, що додає йому кілька нових властивостей і методів.

Тепер на практиці розглянемо, що ж являє собою спадкування (або розширення можливостей) класів:

```
class B extends A {
function B(параметри_для_A, інші_параметри)
{ $this->A(параметри_для_A);
инициализируем інші поля B
}
function Test() { ... }
function Test() { ... }
}
```

Ключове слово extends говорить про те, що створюваний клас є лише "розширенням" класу А, і не більше того. Тобто B містить ті ж самі властивості й методи, що й А, але, крім них і ще деякі додаткові, "свої". Тепер "частина А" перебуває прямо усередині класу В і може бути легко доступна, нарівні з методами й властивостями самого класу В. Наприклад, для об'єкта $obj класу В припустимі вираження $obj->Test() і $obj->Test().

Отже, ми бачимо, що, дійсно, клас В є втіленням ідеї "розширення функціональності класу А". Зверніть також увагу: ми можемо тепер забути, що В успадкував від А деякі властивості або методи - зовні все виглядає так, начебто клас В реалізує їх самостійно.

8 Проектування баз даних для використання в Web

У лекції розглядаються такі питання:

8.1 Концепції реляційних баз даних.
8.2 Способи проектування бази даних для Web.
8.3 Архітектура баз даних для Web.
8.4 Створення бази даних користувачів.
8.5 Система повноважень MySQL.

8.1 Концепції реляційних баз даних

База даних – це сукупність зв'язаних даних, організованих за певними правилами, що передбачає загальні принципи опису, зберігання й маніпулювання, незалежна від прикладних програм. База даних є інформаційною моделлю предметної області. Звертання до баз даних здійснюється за допомогою системи керування базами даних (**СУБД**). СУБД забезпечує підтримку створення баз даних, централізованого керування й організації доступу до них різних користувачів.

Реляційна база даних - база даних, заснована на реляційній моделі. Слово "реляційний" походить від англійського "relation" (відношення). Для роботи з реляційними БД застосовують Реляційні СУБД.

Теорія реляційних баз даних була розроблена доктором Коддом з компанії IBM в 1970 році. У реляційних базах даних всі дані представлені у вигляді простих таблиць, розбитих на рядки й стовпці, на перетинанні яких розташовані дані. Запити до таких таблиць повертають таблиці, які самі можуть ставати предметом подальших запитів. Кожна база даних може

включати кілька таблиць. Коротко особливості реляційної бази даних можна сформулювати в такий спосіб:

- Дані зберігаються в таблицях, що складаються зі стовпців ("атрибутів") і рядків ("записів", "кортежів");
- На перетинанні кожного стовпця й рядка стоїть в точності одне значення;
- У кожного стовпця є своє ім'я, що служить його назвою, і всі значення в одному стовпці мають один тип.
- Запити до бази даних повертають результат у вигляді таблиць, які теж можуть виступати як об'єкт запитів.

Рядки в реляційній базі даних невпорядковані - упорядкування проходить в момент формування відповіді на запит.

Загальноприйнятим стандартом мови роботи з реляційними базами даних є мова SQL.

Завдання тривалого зберігання й обробки інформації з'явилося практично відразу з появою перших комп'ютерів. Для рішення цього завдання наприкінці 60-х років були розроблені спеціалізовані програми, що одержали назву систем керування базами даних (СУБД). СУБД проробили тривалий шлях еволюції від системи керування файлами, через ієрархічні й мережні бази даних. Наприкінці 80-х років домінуючою стала система керування реляційними базами даних (СУРБД). Із цього часу такі СУБД стали стандартом де-факто, і для того, щоб уніфікувати роботу з ними, була розроблена структурована мова запитів (SQL), що являє собою мову керування саме реляційними базами даних.

Модель реляційної бази даних представляє дані у вигляді таблиць, розбитих на рядки й стовпці, на

перетинанні яких перебувають дані. Приклад такої таблиці показаний у таблиці:

Таблиця 8.1- Структура реляционной бази даних.

	стовпець		
рядок	**id_forum**	**name**	**Description**
	1	дизайн	Обговорюються питання дизайну
	2	MySQL	Обговорюються питання, пов'язані з MySQL
	3	PHP	Обговорюються питання, пов'язані з PHP
	4	різне	Інші питання

У табл. 8.1 наведений приклад таблиці forums бази даних великого форуму, у якому є кілька розділів, присвячених різним етапам побудови Web-додатка. Кожний рядок цієї таблиці являє собою один розділ форуму. Чотири рядки таблиці являють собою весь форум.

Кожний стовпець таблиці forums представляє один елемент даних для кожного з форумів. Стовпець id_forum містить унікальний ідентифікатор форуму, стовпець name містить назву форуму й стовпець description містить короткий опис проблеми, обговорюваної на форумі.

8.2 Проектування баз даних для Web

Переваги використання реляційних баз даних у порівнянні із двовимірними файлами. Серед них:

- СУРБД забезпечують більш швидкий доступ до даних, чим двовимірні файли.

- СУРБД можна просто відправити запит на пошук наборів даних, відібраних за певним критерієм.
- СУРБД мають убудований механізм для роботи з паралельним доступом, так що вам, як програмістові, турбуватися про цьому не потрібно.
- СУРБД забезпечують довільний доступ до даних.
- СУРБД мають убудовані системи підтримки привілеїв.

Якщо говорити більш предметно, то використання реляційної бази даних дає можливість швидко й без особливих зусиль відповісти на такі питання, як: "звідки ваші покупці?", "який тип продукції продається найбільше ефективно?" або "які покупці витрачають більше грошей?"

База даних складається з таблиць. Кожна таблиця являє собою набір упорядкованих стовпців, називаних полями (атрибутами). Кожний стовпець має унікальне ім'я усередині таблиці й має певний тип даних. Таблицю можна також представляти як набір рядків (або записів, кортежів), що складаються зі стовпців. Кожний стовпець відповідає одному запису в таблиці. У кожного рядка ті самі стовпці (атрибути). Деякі стовпці мають певну функцію - вони є ключами. За значенням у таких стовпцях можна однозначно визначити унікальність запису в таблицях. Ключ також часто має сполучну функцію, тобто забезпечує зв'язок даних із двох різних таблиць, що дозволяє співвідносити запису у двох таблицях за спеціальними правилами. Сам зв'язок, утворений ключами, називається відношенням між таблицями. Відношення може бути трьох типів: "один до одного", "один до багатьох" і "багато до багатьох". Всі разом: таблиці, правила їхніх відносин і зв'язків, - називається схемою бази даних і являє собою структуроване креслення, що візуально відображає структуру бази даних.

Перед тим, як почати проектування бази даних, подумайте про реальні об'єкти. Адже, як правило, вам треба буде моделювати предмети й відносини між ними виходячи з їхнього положення в реальному світі. Опираючись на це, на кожний "предмет" реального світу вам буде потрібна своя таблиця.

Як приклад візьмемо такий варіант: одержання замовлень на поставку запчастин із заводу-виробника дилерам. Нам будуть потрібні дані про запчастини, клієнтів, замовлення і їхні особливості. Запчастини мають відповідні атрибути - каталожним кодом (номером), назвою й ціною. Виходячи із цього, у базі даних повинні бути як мінімум три таблиці: запчастини (Materials), інформація про клієнтів (Clients) і замовлення (Orders).

8.3 Архітектура баз даних для Web

Система Web-сервера складається із двох об'єктів: Web-браузера й Web-сервера. Між ними повинен існувати канал зв'язку. Web-браузер надсилає запит на сервер, сервер відсилає назад відповідь. Для сервера, що відсилає звичайні статичні сторінки, така архітектура підходить.

Архітектура ж сайту, що містить у собі базу даних, трохи складніше.

Типова транзакція Web-бази даних складається з етапів. Ми розглянемо їх на прикладі магазина " Book-O-Rama".

- Web-браузер користувача відправляє HTTP-запит певної Web-сторінки. Наприклад, пошук у магазині "Book-O-Rama" всіх книг, написаних Лорой Томсон (Laura Thomson), використовуючи HTML-форму. Сторінка з результатами пошуку називається results.
- Web-сервер приймає запит на results.php, одержує файл і передає його механізму PHP на обробку.

- Механізм PHP починає синтаксичний аналіз сценарію. У сценарії присутній команда підключення до бази даних і виконання запиту в ній (пошук книг). PHP відкриває з'єднання із сервером MySQL і відправляє необхідний запит.
- Сервер MySQL приймає запит у базу даних, обробляє його, а потім відправляє результати - у цьому випадку, список книг - назад у механізм PHP.
- Механізм PHP завершує виконання сценарію, форматувати результати запиту у вигляді HTML, після чого відправляє результати в HTML-форматі Web-серверу.
- Web-сервер пересилає HTML у браузер, за допомогою якого користувач переглядає список необхідних книг.

Процес цей, як правило, протікає незалежно від того, який сценарний механізм і який сервер баз даних використовується. Найчастіше програмне забезпечення Web-сервера, механізм PHP і сервер баз даних перебувають на одній машині. Правда, не менш часто сервер бази даних працює на іншій машині. Це робиться з міркувань безпеки, збільшення обсягу або поділу потоку. З погляду перспектив розвитку, у роботі обидва варіанти однакові, однак у плані продуктивності другий варіант може виявитися більш кращим.

8.4 Створення баз даних користувачів

Система баз даних MySQL може підтримувати безліч різних баз даних. Звичайно на один додаток буде існувати одна база даних. У нашім прикладі з " Book-O-Rama" база даних буде називатися books.

Створення бази даних

Це найпростіша частина. Уведіть у командному рядку MySQL:

```
mysql> create database dbname;
```

Замість dbname варто ввести ім'я бази, що потрібно створити. Для цілей приклада "Book-0-Rama" необхідно створити базу даних з ім'ям books.
Відповідь повинна виглядати приблизно так:

```
Query OK, I row affected (0.06 sec)
```

Це значить, що все спрацювало як треба. Якщо подібної відповіді не було, подивіться, є наприкінці рядка крапка з комою. Крапка з комою повідомляє MySQL, що уведення команди завершене і її настав час виконувати.

8.5 Система повноважень MySQL

Користувачі й привілеї

Система MySQL може містити багато користувачів. Користувач root з міркувань безпеки повинен використовуватися тільки для адміністративних цілей. Кожний користувач, якому необхідно працювати в системі, повинен одержати обліковий запис і пароль. Вони не повинні бути точно такими ж, як поза MySQL (наприклад, ім'я й пароль, які використовуються для входу в UNI або NT). Теж саме ставиться й до користувача root. Взагалі, розумно мати різні паролі для входу в систему й для MySQL, особливо, якщо мова йде про пароль користувача root.

Паролі для звичайних користувачів установлювати не обов'язково, але все-таки настійно рекомендується зробити це.

При створенні Web-бази даних варто завести хоча б одного користувача в кожному Web-додатку.

Ви запитаєте: "Чому так варто робити?" - і відповідь підкажуть привілеї.

Установка користувачів: команда GRANT

Команди GRANT і REVOKE використовуються для того, щоб надавати й віднімати права в користувачів MySQL на чотирьох рівнях привілеїв. От ці рівні:

- Глобальний
- Бази даних
- Таблиці
- Стовпця

За допомогою команди GRANT можна заводити користувачів і надавати їм привілеї. Загальний вид команди GRANT виглядає в такий спосіб:

```
GRANT privileges [columns]
ON item
TO user_name [IDENTIFIED BY 'password' ]
[WITH GRANT OPTION]
```

Конструкції, укладені у квадратні дужки, є необов'язковими. У даній синтаксичній структурі присутній ряд заповнювачів.

Перший, privileges (привілеї), повинен заповнюватися розділеним комами списком привілеїв, чітко визначених в MySQL.

Заповнювач columns (стовпці) необов'язковий. Ним можна скористатися для того, щоб призначати привілеї по конкретних стовпцях. Можна визначати або ім'я одного стовпця, або розділений комами список імен стовпців.

Заповнювач item (елемент) може бути базою даних або таблицею, до якої застосовуються нові привілеї.

Указавши на його місці *.*, можна встановити привілеї для всіх баз даних. Така дія називається призначенням глобальних привілеїв. Той же ефект досягається й вказівкою лише *.

Найчастіше задаватися будуть всі таблиці в певній базі даних - dbname. * (ім'я_бази_даних.*), конкретна таблиця - dbname.tablename (ім'я_бази_даних ім'я_-таблиці) або певні стовпці - dbname.tablename і список необхідних стовпців у заповнювачі columns. Все перераховане представляє три інших доступних рівні привілеїв: бази даних, таблиці й стовпця, відповідно. Якщо при видачі цієї команди використовується якась конкретна база даних, параметр tablename сам по собі буде витлумачений як "таблиця в поточній базі даних".

Як значення user name повинне стояти ім'я користувача, під яким користувач повинен входити в MySQL. Пам’ятайте, що воно не повинне збігатися з реєстраційним ім'ям, під яким входите в систему ви. В MySQL user name може містити в собі й ім'я хоста, що досить зручно для того, щоб розрізняти користувачів, скажемо, laura (є через laura@localhost) і laura@somewhere.com. Це дуже корисна річ, оскільки часто користувачі на різних доменах мають ті самі імена. Крім того, підвищується ступінь захищеності, оскільки є можливість указати, звідки користувачі можуть заходити й до яких баз даних або таблицям можуть мати доступ.

Параметр password варто замінити на пароль, необхідний для входу. Керуйтеся загальними правилами використання паролів. Пароль не повинен бути легко вгадуваним. Не варто вживати слово зі словника або співпадаюче з ім'ям користувача. В ідеалі пароль повинен містити в собі букви верхнього й нижнього регістрів і небуквені символи.

Опція WITH GRANT OPTION, якщо зазначено, надає право користувачеві передавати свої привілеї іншим.

Привілеї зберігаються в чотирьох системних таблицях, що належать базі даних з ім'ям mysql. Ці чотири таблиці називаються так: mysql.user, mysql.db,

mysql.tables_priv і mysql.columns_priv; вони прямо відносяться до згаданих раніше чотирьох рівнів привілеїв. Як альтернатива команді GRANT, можна безпосередньо правити ці таблиці.

9 Доступ до бази даних MySQL з Web за допомогою PHP

У лекції розглядаються такі питання:

9.1 Встановлення з'єднання.

9.2 Вибір бази.

9.3 Отримання результату запита.

9.4 Інші PHP-інтерфейси роботи з базами даних.

9.1 Встановлення з'єднання

Для підключення до сервера MySQL у сценарії повинен бути такий рядок:

```
@ $db = mysql_pconnect("localhost", "bookorama", "bookorama");
```

Для підключення до бази даних використовується функція mysql_pconnect() з наступним прототипом:

```
int mysql_pconnect ( [string host [.-port] [:/socketpath] ] ,
[string user] , [string password] ) ;
```

Буде потрібно вказати ім'я вузла (host), на якому розміщений сервер MySQL, ім'я користувача (user), щоб увійти в нього, і пароль (password). Все це в принципі необов'язково і якщо не вказати все перераховане вище, функція скористається значеннями за замовчуванням - локальна машина замість вузла, ім'я користувача, під яким запущений PHP, і порожній пароль.

У випадку успіху функція поверне ідентифікатор зв'язку з базою даних (який варто зберегти для подальшого використання), а у випадку невдачі - значення false.

Результат не варто ігнорувати, оскільки без з'єднання з базою даних робота неможлива. Це робить наступний код:

```
if (!$db)
{
echo "Error: Could not connect to database. Please try again later.";
exit;
}
```

Як альтернативу, можна використовувати іншу функцію, що робить практично те ж саме - mysql_connect(). Єдина відмінність полягає в тому, що mysql_connect() установлює постійне з'єднання з базою даних.

Звичайне з'єднання з базою даних закривається, коли сценарій завершує своє виконання або коли звертається до функції mysql_close(). Постійне з'єднання залишається відкритим і після того, як сценарій виконаний, а функцією mysql_close() його закрити не можна.

Може виникнути питання, для чого це потрібно. Відповідь така: з'єднання з базою даних припускає деякі непродуктивні витрати, що вимагає часу. Коли викликається mysql_pconnect(), перш ніж вона спробує підключитися до бази даних, вона автоматично перевірить, чи немає вже відкритого постійного з'єднання. Якщо є, вона не стане відкривати нове. Це й час заощаджує, і запобігає перевантаженню сервера.

Однак якщо PHP виконується як CGI, то постійне з'єднання виявиться не таким вже і постійним. (Кожний виклик сценарію PHP запускає нову копію механізму PHP і закриває її, коли сценарій завершує свою роботу. Це, у свою чергу, також закриває будь-яке постійне з'єднання.)

Кількість з'єднань в MySQL, які існують одночасно, обмежено. Границю встановлює параметр max_connections. Його завдання (як і родинного йому параметра Apache MaxClients) - змусити сервер відкидати нові запити на з'єднання, коли ресурси вузла зайняті або коли програмне забезпечення не функціонує.

Значення цих параметрів можна змінювати, редагуючи файл конфігурації. Щоб настроїти MaxClients в Apache, варто правити файл httpd.conf. Настроювання max_connections в MySQL здійснюється за рахунок редагування файлу my.conf. Якщо ви користуєтеся постійними з'єднаннями, і практично кожній сторінці на вашім сайті потрібен доступ до бази даних, вам, мабуть, знадобиться постійне з'єднання для кожного процесу Apache. Якщо ж використовуються значення параметрів, прийняті за замовчуванням, можуть виникнути певні складності. По-змовчанню Apache допускає до 150 з'єднань, а MySQL - тільки 100. В особливо напружений час з'єднань може не вистачити. Тому найкраще настроїти параметри так, щоб у кожного процесу Web-сервера було своє з'єднання, звичайно, з оглядкою на технічні можливості застосовуваних апаратних засобів.

9.2 Вибір бази даних

Працюючи з MySQL з командного рядка, необхідно вказувати, яка база даних потрібна:

```
use books ;
```

Те ж саме необхідно й при підключенні з Web. Це може зробити PHP-функція

```
mysql_select_db():
```

```
mysql_select_db ("books") ;
```

Прототип цієї функції виглядає так:

```
int mysql_select_db (string database, [int database_connection] ) ;
```

У результаті буде використовуватися база даних з ім'ям database. Можна також використовувати з'єднання з базою даних, для якого потрібно виконати цю операцію (у нашім випадку $db), однак, якщо його не вказати, буде використовуватися останнє відкрите з'єднання. Якщо відкрите з'єднання не існує, воно відкривається за замовчуванням, як якби викликалася mysql_connect().

9.3 Отримання результату запита

Основні кроки виконання запитів до бази даних через Web

У будь-якому сценарії, що забезпечує доступ до бази даних з Web, є кілька базових кроків:

- Перевірка й фільтрація даних, що виходять від користувача.
- Установка з'єднання з необхідною базою даних.
- Передача запиту в базу даних.
- Одержання результатів.
- Подання результатів користувачеві.

Те ж саме робить і сценарій results.php, і зараз ми досліджуємо кожний із цих етапів.

Виконання запиту до бази даних

Щоб здійснити запит, можна скористатися функцією mysql_query(). Однак запит необхідно настроїти:

```
$query = "select * from books where ".$searchtype." Like
'%".$searchterm." %'";
```

У цьому випадку буде відшукуватися значення, уведене користувачем ($searchterm), у поле, що вказав користувач ($searchtype).

Одержання результатів запиту

Розмаїтість функцій дає можливість одержати результат різними способами. Ідентифікатор результату - це ключ доступу до рядків, повернутим запитом, яких може бути нуль, один й більше.

У нашім прикладі використовувалися дві функції: mysql_numrows() і mysql_fetch_array(). Функція mysql_numrows() повідомляє кількість рядків, які повертає запит. У неї варто передати ідентифікатор результату:

```
$num_results = mysql_num_rows($result) ;
```

Це корисно знати, якщо планується обробляти або відображати результати. Знаючи їхню кількість, можна організувати цикл:

```
for ($i=0; $i )
{
// обробка результатів
}
```

На кожній ітерації циклу відбувається виклик mysql_fetch_array(). Цикл не буде виконуватися, якщо немає рядків. Ця функція бере кожний рядок зі списку результату й повертає її у вигляді асоціативного масиву, із ключем як ім'ям атрибутива й значенням як відповідним значенням масиву:

```
$row = mysql_fetch_array($result) ;
```

Маючи $row в асоціативному масиві, можна пройти кожне поле й належним чином його відобразити:

```
echo " ISBN: ";
echo stripslashes($row["isbn"] ) ;
```

Як уже згадувалося, stripslashes() викликають для того, щоб "підчистити" значення, перш ніж відображати його користувачеві.

Існують кілька варіантів одержання результату з ідентифікатора результату. Замість асоціативного масиву можна скористатися нумерованим масивом, застосувавши mysql_fetch_row():

```
$row = mysql_fetch_row($result);
```

Значення атрибутів будуть зберігатися в кожному порядковому значенні $row[0], $row[l] і т.д.

За допомогою функції mysql_fetch_object() можна вибрати рядок усередину об'єкта:

```
$row = mysql_fetch_object($result);
```

Після цього до кожного атрибута можна одержати доступ через $row->title, $row->author і т.д.

Кожний із цих варіантів має на увазі вибірку рядка за раз. Інший варіант - одержати доступ, використовуючи mysql_result(). Для цього буде потрібно вказати номер рядка (від 0 до кількості рядків мінус 1) і назва поля, наприклад:

```
$row = mysql_result($result, $i, "title");
```

Назву поля можна задати у вигляді рядка (або у формі "title" або у формі "books.title") або номером (як в mysql_fetch_row()). Не варто змішувати mysql_result() з іншими функціями вибірки.

Рядково-орієнтовані функції вибірки набагато більш ефективні, ніж mysql_result(), так що краще використовувати одну з них.

Від'єднання від бази даних

Для закриття непостійного з'єднання застосовується функція:

```
mysql_close( database_connection) ;
```

Однак у цьому немає особою необхідності, оскільки із завершенням виконання сценарію з'єднання закриється автоматично.

Створення й видалення баз даних

Для створення нової бази даних MySQL з PHP-сценарію застосовується функція mysql_create_db(), а для видалення бази даних - mysql_drop_db().

Розглянемо прототипи цих функцій:

```
int mysql_create_db(string database, [int database_connection] );
int mysql_drop_db(string database, [int database_connection] ) ;
```

Обидві функції використовують ім'я бази даних і з'єднання. Якщо з'єднання немає, буде використовуватися останнє відкрите. Функції створюють або видаляють зазначену базу даних. У випадку успіху функції повертають значення true, а у випадку невдачі - false.

9.4 Інші PHP-інтерфейси роботи з базами даних

PHP підтримує різні бібліотеки, що дає можливість підключатися до величезної кількості баз даних, включаючи Oracle, Microsoft SQL Server, mSQL і PostgreSQL.

У цілому принципи підключення й подачі запитів кожної із цих баз даних ті самі. Назви функцій можуть бути різними, різні бази даних можуть мати різну функціональність, але якщо ви можете підключитися до MySQL, то інші бази навряд чи поставлять вас у безвихідне положення.

Якщо необхідно використовувати базу даних, що не має специфічної бібліотеки, доступної в PHP, можна вдатися до узагальнених функцій ODBC.

ODBC - це відкритий інтерфейс доступу до баз даних і є стандартом підключення до баз даних. Функціональність ODBC не можна назвати надто широкою, однак на те є цілком очевидні причини: або універсальна сумісність, або залучення специфічних можливостей кожної системи.

До того ж до PHP-бібліотек, доступні такі класи абстракції баз даних, як Metabase, що дозволяє використовувати ті самі назви функцій у кожному типі бази даних.

На сьогоднішній день розроблені тисячі бібліотек, орієнтованих на різні області застосування. Крім того, цілком можливо, що існуючі рішення з якихось причин вам не підійдуть, і ви захочете створити свою власну бібліотеку. У кожному разі, вам знадобитися підключити цю бібліотеку до свого проекту.

Існують три основних способи підключення додаткових бібліотек.

Перший спосіб заснований на тім, що будь-яка PHP бібліотека - це набір PHP скриптів, які можна використовувати точно так як і будь-які інші скрипти. Тобто скопіювати в папку із проектом і в міру необхідності підключати за допомогою операторів include або require. При цьому потрібно пам'ятати, що структура каталогів самої бібліотеки повинна залишатися постійною. Цей спосіб зручний, якщо ви хочете поширювати файли бібліотеки разом з вашим проектом.

Другий спосіб припускає використання однієї й тієї ж копії бібліотеки декількома проектами. У цьому випадку взаємне розміщення бібліотеки й вашого проекту заздалегідь невідома. Для того щоб зробити бібліотеку доступною, необхідно вказати її розміщення у файлі php.ini за допомогою параметра include_path.

Розглянемо приклад. Ми хочемо зробити доступною бібліотеку simpletest.

Архів з бібліотекою ми розпакували в папку C:\simpletest_php. Тобто файли бібліотеки розміщені в такий спосіб:

```
C:\simpletest_php\simpletest\файли бібліотеки
```

Після цього у файлі php.ini (він розташований у папці з PHP) шукаємо параметр include_path, і вказуємо розміщення бібліотеки.

```
include_path="...;C:\simpletest_php;..."
```

Замість трикрапки у вас будуть зазначені папки з іншими бібліотеками. Якщо перед параметром стоїть крапка з комою (перетворює рядок у коментар) заберіть її. Врахуйте, що параметр include_path може бути створений автоматично, наприклад, менеджером пакетів PEAR. У

цьому випадку просто додавайте через крапку з комою розміщення ваших бібліотек.

Використовувати підключену бібліотеку можна в такий спосіб:

```
require_once "simpletest/unit_tester.php";
```

Третій спосіб стосується пакетів PEAR. По суті це дуже великий набір бібліотек для PHP, багато з яких тісно один з одним зв'язані. Для роботи із цією бібліотекою існує спеціальна програма - менеджер пакетів. Він дозволяє встановлювати пакети як із сайту проекту, за допомогою команди pear install <ім'я_пакета>, так і з локального комп'ютера (попередньо необхідно завантажити архів з потрібним пакетом) pear install <ім'я_пакета>.tgz.

Після установки в папці з PHP з'явиться папка PEAR, у якій будуть розміщені файли бібліотек.

10 Додаткові можливості MySQL

У лекції розглядаються такі питання:

10.1 Забезпечення безпеки баз даних MySQL.

10.2 Одержання додаткової інформації про бази даних.

10.3 Оптимізація проектування.

10.4 Резервне копіювання.

10.1 Забезпечення безпеки баз даних MySQL

Безпека дуже важлива, особливо коли ви починаєте підключати базу даних MySQL до Web-сайту.

MySQL з точки зору операційної системи

Якщо ви працюєте в UNIX-подібній операційній системі, то запускати MySQL- сервер (mysqld) як привілейованому користувачеві не рекомендується, оскільки крім повного набору привілеїв це дає користувачеві MySQL право читати й записувати файли в будь-якому місці операційної системи.

Для цієї мети найкраще створити окремого користувача. До того ж, можна зробити каталоги (у які зберігаються фізичні дані) доступними тільки конкретному користувачеві MySQL. У багатьох установках сервер настроєний на запуск під ім'ям користувача mysql у групі mysql.

В ідеалі, краще встановлювати сервер MySQL за брандмауером. Це дасть можливість запобігти несанкціонованому доступу - перевірте, чи зможете ви підключитися до сервера ззовні через порт 3306. Це порт, на якому MySQL запускається за замовчуванням і в брандмауері він повинен бути закритий.

Паролі

Стежите за тим, щоб у всіх користувачів (особливо, у привілейованих) були визначені паролі, причому добре обрані й регулярно змінювані, як в операційній системі. Головне, що варто пам'ятати в цьому випадку - паролі, складені зі словникових слів, досить слабкі. Краще скористатися комбінаціями з букв і цифр.

Якщо ви збираєтеся зберігати паролі у файлах сценаріїв, стежте за тим, щоб тільки той користувач, чий пароль перебуває в цьому файлі, міг його відкрити. Мова йде про два типові випадки:

- У сценарії mysql.server може знадобитися пароль привілейованого користувача UNIX. Якщо це так, то нехай доступу до цього файлу не має ніхто, крім нього.
- Пароль такого користувача повинен бути збережений у PHP сценаріях, використовуваних для підключення до баз даних. Це можна зробити безпечно, якщо помістити ім'я користувача й пароль у файл із назвою, скажемо, dbconnect.php, що буде включатися в міру необхідності. Сценарій може зберігатися поза деревом Web-документів і бути доступним тільки певному користувачеві. Пам’ятайте, що якщо помістити ці деталі в a.inc або у файл із іншим розширенням у рамках Web-дерева, варто проявляти граничну обережність і перевіряти, чи знає Web-сервер, що ці файли потрібно інтерпретувати як PHP-код, щоб деталі не можна було переглянути в Web-браузері.

Не зберігайте паролі своєї бази даних у текстових файлах. Паролі MySQL звичайно так не зберігають, однак найчастіше в Web-додатках зберігають імена користувачів Web-сервера і їхні паролі. Шифрувати паролі (односторонньо) можна, використовуючи MySQL-функції PASSWORDQ і MD5(). Пам’ятайте, що коли ви вставляєте пароль (INSERT) в одному із цих форматів, виконуючи

SELECT (щоб забезпечити вхід користувача в систему), доведеться використовувати цю функцію ще раз, щоб перевірити, який пароль був уведений користувачем.

10.2 Одержання додаткової інформації про бази даних

Детальний опис системи привілеїв

Раніше ми вже досліджували питання підключення користувачів і присвоєння їм привілеїв за допомогою команди GRANT.

Оператор GRANT впливає на таблиці в спеціальній базі даних, що називається mysql. Інформація про привілеї зберігається в п'яти таблицях цієї бази даних. Тому, привласнюючи користувачам привілеї в базах даних, будьте обережні з видачею доступу до бази даних mysql.

Команда GRANT доступна в MySQL тільки починаючи з версії 3.22.11.

Подивитися вміст бази даних mysql можна, зареєструвавшись у системі як адміністратор і набравши:

```
use mysql;
```

Після можна переглянути таблиці цієї бази, набравши:

```
show tables;
```

У результаті відображається щось подібне:

```
Tables in mysql
columns_priv
db
host
tables_priv
user
```

У кожній із цих таблиць утримується інформація про привілеї. Їх ще називають таблицями прав. Їхні функції можуть бути трохи різними, але завдання у всіх одне - визначати, що дозволено, а що не дозволено робити користувачам. У кожній таблиці втримується два типи поля: контекстні поля, які ідентифікують користувача, хост і частина бази даних, і поля привілеїв, що визначають, які дії може виконувати користувач, виходячи зі свого контексту.

Таблиця user призначена для визначення, чи може користувач підключатися до сервера MySQL і чи володіє він привілеями адміністратора.

Таблиці db і host визначають, до яких баз даних користувач може мати доступ. Таблиця tablesjriv - які таблиці в базі даних дозволяється використовувати, а columns_priv - до яких стовпців у таблицях є доступ.

Таблиця user

У цій таблиці втримуються дані про глобальні привілеї користувача, як то: чи має право користувач підключатися до сервера MySQL взагалі і є чи в нього які-небудь привілеї глобального рівня, тобто привілеї, які поширюються на всі бази даних у системі.

Оператор describe user - відображає структуру цієї таблиці. Схема таблиці user наведена в табл.10.1

Таблиця 10.1 - Схема таблиці user у базі даних mysql

Поле	Тип
Host	char(60)
User	char(16)
Password	char(16)
Select_priv	enumCN'/Y1)

Продовження таблиця 10.1 - Схема таблиці user у базі даних mysql

Поле	Тип
lnsert_priv	enum('NYY')
Update_priv	enum('N','Y')
Delete_priv	enum('NYY')
Create_priv	enum('NYY')
Drop_priv	enum('NYY')
Reload_priv	enum('N','Y')
Shutdown_priv	enum('N','Y')
Process_priv	enum('N','Y')
File_priv	enum('N','Y')
Grant_priv	enum('N','Y')
References_priv	enum('N','Y')
lndex_priv	enum('N','Y')
Alter_priv	enum('N','Y')

Кожний рядок у цій таблиці відповідає набору привілеїв користувача, що реєструється з певного хоста із застосуванням ім'я й пароля. У таблиці є поля з діапазоном, які описують діапазон для інших полів, називаних полями привілеїв.

Наведені в таблиці привілеї повністю погоджуються із привілеями, надаваними з використанням GRANT. Наприклад, select_priv прямо відповідає привілеям по виконанню оператора SELECT.

Якщо користувач має певний привілей, значенням у стовпці типу буде Y, якщо даного привілею в користувача немає, значенням стовпця буде N.

Всі привілеї, перераховані в таблиці user, є глобальними, тобто вони застосовуються до всіх баз даних у системі (включаючи й базу mysql). Більшість значень у стовпці, рівних Y, будуть мати адміністратори, тоді як для користувачів буде характерно багато значень N.

Таблиці db і host

Більшість привілеїв для рядових користувачів системи розміщаються в таблицях db і host.

Таблиця db визначає, які користувачі до яких таблиць і з яких хостів можуть одержати доступ. Перераховані в таблиці привілеї застосовуються до будь-якої бази даних.

Таблиця host доповнює таблицю db. Якщо користувач повинен з'єднуватися з деякою базою даних, використовуючи при цьому безліч хостів, у таблиці db для нього не буде зазначено жодного хоста. Навпаки, з таким користувачем буде зв'язаний набір записів у таблиці host, по одній для кожного можливого хоста.

Таблиця 10.2 - Схема таблиці db у базі даних mysql

Поле	Тип
Hosf	char(60)
Db	char(64)
User	char(16)
Select_priv	enum('N','Y')
lnsert_priv	enum('N','Y')
Update_priv	enum('N','Y')
Delete_priv	enum('N','Y')
Create_priv	enum('N','Y')
Drop_priv	enum('N','Y')

Таблиця 10.2 - Схема таблиці db у базі даних mysql

Поле	Тип
Grant_priv	enum('N','Y')
References_priv	enum('N','Y')
lndex_priv	enum('N','Y')
Alter_priv	enum('N','Y')

Таблиця 10.3 - Схема таблиці host у базі даних mysql

Поле	Тип
Host	char(60)
Db	char(64)
Select_priv	enum('N','Y')
lnsert_priv	enum('N','Y')
Update_priv	enum('N','Y')
Delete_priv	enum('N','Y')
Create_priv	enum('N','Y')
Drop_priv	enum('N','Y')
Gran+_priv	enum('N','Y')
References_priv	enum('N','Y')
lndex_priv	enum('N','Y')
Alter_priv	enum('N','Y')

Таблиці tables_priv і columns_priv

Ці таблиці призначені для зберігання привілеїв, відповідно, на рівні таблиці й на рівні стовпців. Вони працюють подібно таблиці db за винятком того, що

забезпечують привілеї для таблиць у певній базі даних і для стовпців у певній таблиці.

Структура розглянутих таблиць трохи відрізняється від структури таблиць user, db і host.

Таблиця 10.4 - Схема таблиці tables_priv у базі даних mysql

Поле	Тип
Host	char(60)
Db	char(64)
User	char(16)
Table_name	char(64)
Grantor	char(77)
Timestamp	timestamp(14)
Table_priv	set('Select', 'Insert', 'Update', 'Delete', 'Create', 'Drop', 'Grant','References', 'Index', 'Alter'))
Column_priv	set('Select', 'Insert', 'Update', 'References')

Схема таблиці columns_priv у базі даних mysql

Стовпець Grantor таблиці tables_priv зберігає інформацію про користувача, що видав привілей даному користувачеві. Стовпець Timestamp в обох таблицях зберігає інформацію про дату й час видачі привілею.

Таблиця 10.5 - Схема таблиці columns_priv у базі даних mysql

Поле	Тип
Host	char(60)
Db	char(60)
User	char(16)
Table_name	char(60)

Продовження таблиці 10.5 - Схема таблиці columns_priv у базі даних mysql

Поле	Тип
Column_name	char(59)
Timestamp	timestamp(14)
Column_priv	set('Select', 'Insert', 'Update', 'References')

10.3 Оптимізація проектування

Головне в базі даних - це щоб вона була якнайменше. Цього можна досягнути разом з поліпшеним проектуванням, що зменшує надмірність. Другий варіант пов'язаний із застосуванням найменшого з можливих типів даних у стовпцях. Де це тільки можливо, заберіть нульові значення й зробіть первинні ключі як можна коротше.

Уникайте неоднорідної довжини стовпців, де це можливо (наприклад, VARCHAR, TEXT, BLOB). Таблиці з фіксованою довжиною полів працюють швидше, однак разом з тим можуть займати більший простір.

10.4 Резервне копіювання баз MySQL

Зробити копію всіх статичних HTML- і інших документів просто. Набагато більш складним представляється завдання створення копії (далі backup) такої динамічної структури, як база даних MySQL. Основні труднощі, які виникають перед адміністратором розміщеного на хостингу сайту, звичайно бувають такі:

1. Відсутність фізичного доступу до файлів бази даних. Як правило, провайдери хостинга надають

можливість роботи з базою даних тільки через скрипти або спеціальний mysql-клієнт, але не дають прав на доступ безпосередньо до файлів, у яких утримуються дані з MySQL-бази.

2. Відсутність в адміністратора знань про те, як взагалі треба робити backup. Звичайно таке завдання виникає тільки, коли вже пізно. Тобто, у випадку аварії, вторгнення хакерів або в інших позаштатних ситуаціях.

3. У випадку, якщо веб-майстер не володіє в достатній мірі навичками роботи зі спеціалізованими утилітами з пакета MySQL, можуть виникати труднощі, пов'язані з обмеженнями, що накладаються хостинг-провайдером на користувальницькі аккаунти.

Копіювання бази MySQL

Існує програма mysqldump, що дозволяє швидко й просто робити операції по створенню резервних копій баз MySQL. Також mysqldump дає можливість робити дуже тонкі настроювання для керування процесом створення резервних копій баз даних або окремих таблиць.

Наприклад: є хостинг, є база даних DBNAME, що виділив Вам хостинг-провайдер. Є хост HOST, на якому розміщений сервер MySQL, логин LOGI до нього, порт PORT, на якому працює сервер, а також пароль PASS. Маючи всі ці дані, можна зробити dump (дамп, копію) бази DBNAME так (виконуємо в unix shell):

```
mysqldump -uLOGi -PPORT -hHOST -pPASS DBNAME > dump.txt
```

Після виконання даної команди у файлі dump.txt у нас буде копія MySQL-бази DBNAME. Це відбудеться тільки в тому випадку, звичайно, якщо всі параметри Ви задасте вірно, відповідно до настроювань свого хостинга.

Програма mysqldump робить вивід результатів прямо Вам на екран. Потрібно перенаправляти вивід у який-небудь файл. Наприклад, як у цьому випадку - " > dump.txt ".

Розглянемо більш тонкі настроювання mysqldump:

---databases дозволяє зробити так, що mysqldump включить у сценарій відновлення команди CREATE DATABASE /*!33333 IF NOT EXISTS*/ DBNAME і USE DBNAME. Це дозволить створювати робочі бази "з нуля";

---all-databases дозволяє зробити копії всіх баз даних, які існують на даному MySQL-сервері. Якщо ж потрібно зробити копії тільки деяких баз, потрібно просто вказати їх через пробіл при виклику mysqldump з командного рядка;

Ключ **---help**. Програма mysqldump має безліч версій. Подивитися, які можливості підтримуються конкретно Вашою версією, можна за допомогою цього ключа;

---add-drop-table - ключ, що змусить mysqldump додавати в підсумковий сценарій команду drop table перед створенням таблиць. Це дозволить уникнути деяких помилок при відновленні бази з резервної копії;

---no-data. За допомогою цього ключа можна швидко зробити копію структури таблиці/баз без самих даних;

---result-file=... - цей ключ можна використовувати для перенапрямку виводу у файл;

Автоматизація резервного копіювання

Існує програма - cron. Вона дозволяє запускати процеси в зазначений користувачем час або з певною періодичністю. Але Cron у загальному випадку існує тільки під Unix,

СПИСОК ЛІТЕРАТУРИ

1. Веллинг Л. Разработка Web-приложений с помощью PHP и MySQL / Л. Веллинг, Л. Томсон, пер. с англ. – 2-е изд. – М. : Издательский дом «Вильямс», 2004. – 800 с.

2. Дюбуа П. MySQL : учебное пособие / П. Дюбуа ; пер. с англ. — М. : Издательский дом «Вильямс», 2001. – 816 с.

3. Кухарчик А. PHP: обучение на примерах / А. Кухарчик – Мн. : Новое знание, 2004. – 237 с.

Навчальне видання

WEB-ПРОГРАМУВАННЯ

Конспект лекцій

для студентів спеціальності
(7) 8.05010102 "Інформаційні технології проектування"
денної та заочної форм навчання

Відповідальний за випуск зав. секції інформаційних технологій проектування канд. техн. наук, доц. О. В. Бондар

Редактор Н. В. Лисогуб

Комп'ютерне верстання В. В. Шендрик

Формат 60x84/16. Ум. друк. арк. 7,21. Обл.-вид.арк. 5,85. Тираж 40 пр. Зам. № .

Видавець і виготовлювач
Сумський державний університет
вул. Римського-Корсакова, 2, м. Суми, 40007
Свідоцтво суб'єкта видавничої справи ДК № 3062 від 17.12.2007.