Тема: Використання серверних мов сценаріїв

План

1. Технології серверної сторони.
2. Вимоги до додатків сторони сервера.
3. Технологія CGI.
4. Технологія серверів.
5. Технологія JSP.
6. Технологія .NET.
7. Технологія PHP.
8. Порівняльний аналіз базових технологій серверної сторони.
9. Вступ до PHP.
10. Використання PHP.
11. Контрольні запитання.
12. Завдання для самостійного виконання.

1. Технології серверної сторони

Для розробки додатків серверної сторони необхідно обрати базову технологію, на якій ці додатки будуть засновані.

На даний момент існують і успішно застосовуються різні види технологій побудови web-додатків серверної сторони. Усі такі додатки мають спільну мету — реалізацію бізнес-логіки на стороні сервера і генерацію коду для клієнта. Також у цих додатків однакова архітектура взаємодії сервера і клієнта, а також спільний протокол взаємодії — HTTP.

Загальна логіка роботи додатка серверної сторони представлена на Рис.1.

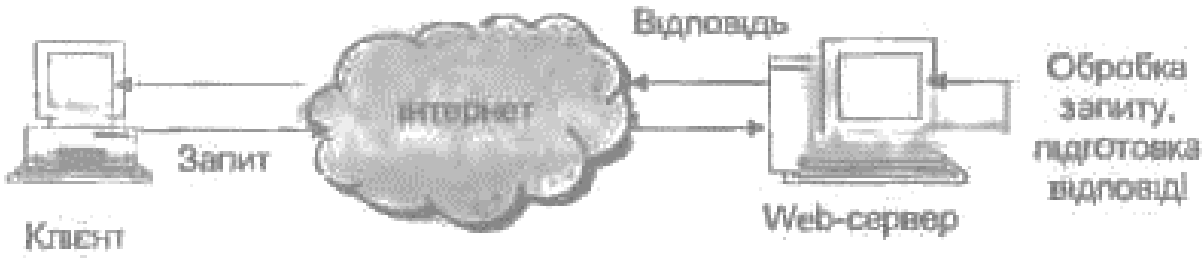

Рис. 1. Робота додатків серверної сторони

Як видно з Рис. 1, робота серверних додатків відбувається в три основних етапи:

1. Запит. Клієнт, використовуючи web браузер, ініціює запит до сервера.

2. Обробка запиту, підготовка відповіді. Після одержання запиту web-сервер проводить обробку запитуваного ресурсу. У випадку, якщо запитується статичний ресурс (HTML сторінка, малюнок, документ), ця інформація форматується для протоколу HTTP і передається клієнтові як відповідь.

Якщо ж запитується динамічний ресурс, запит передається на обробку відповідному контейнеру web-додатків, де і відбувається подальша робота.

3. Після формування дані передаються клієнтові за допомогою протоколу HTTP як відповідь. Відповідь містить дані (зазвичай, HTML-код або двійкові дані), а також додаткові параметри, передані в заголовках HTTP відповіді.

Робота додатків серверної сторони завжди відбувається за описаним вище сценарієм. Очевидно, що такий підхід створює труднощі у створенні web-додатків, основною з яких є відсутність постійного стану web-додатка (так звана stateless programming). Це означає, що додаток працює виключно в режимі «запит—відповідь» , не маючи даних про попередні кроки користувача або будь-якої іншої постійної інформації. Для вирішення цієї

проблеми застосовується поняття користувальницької сесії, яка дозволяє зберігати дані на сервері протягом сеансу роботи користувача.

Однак наявністю сесій проблеми цілком не усуваються. Чим більше можливостей надає платформа реалізації для додатків серверної сторони в подоланні цих складностей, тим швидше й ефективніше може здійснюватися розробка. Далі будуть розглянуті різні підходи до створення додатків серверної сторони, їхні переваги і недоліки, а також конкретні платформи.

2. Вимоги до додатків сторони сервера

Під час створення додатків серверної сторони необхідно виділити два основних підходи:

- безпосередня обробка запитів і формування відповідей;
- вбудовування програмного коду в шаблони web-сторінок.

Перший підхід надає найбільші можливості щодо управління обробкою і підвищення продуктивності. Він передбачає передачу всіх даних про запит коду, який безпосередньо виконується, що може як сформувати відповідь зі сторінкою для користувача, так і відкрити на передачу потік двійкових даних, наприклад для передачі зображення. Однак при такому підході всі дані для передачі формуються програмним шляхом, що сповільнює розробку простих сторінок і ускладнює взаємодію між розробником дизайну та програмістом. Прикладами такого підходу є технології CGI та Java Servlets.

Другий підхід використовує шаблони сторінок користувача, оформлені певним чином, що дозволяє вставляти в них ділянки програмного коду. Цей підхід особливо ефективний при створенні простих додатків, основна інформація в яких статична, а динамічна інформація може бути згенерована простими програмними конструкціями. У процесі розробки складних програмних систем цей варіант ускладнює взаємодію між компонентами й реалізацію складної архітектури. Також він менш ефективний за продуктивністю й обмежує можливості з реалізації складних сторінок.

Прикладами цього підходу є технології PHP, ASP, JSP.

Крім різних підходів до генерації сторінок, платформи розробки по-різному задовольняють сучасні вимоги до створення складних web-систем. Наведемо найбільш важливі з цих вимог:

- незалежність від платформи;
- продуктивність, можливість масштабування;
- можливості розширення й інтеграції;
- простота використання, наявність засобів розробки;
- наявність необхідних програмних бібліотек.

Отже, ми визначили низку вимог, яким має відповідати сучасна платформа розробки. Нижче розглянуто найбільш популярні на даний момент платформи та їхні особливості з точки зору наведених критеріїв.

3. Технологія CGI

Технологія *CGI (Common Gateway Interface)* відрізняється від інших тим, що є найбільш низькорівневою і є стандартом інтерфейсу, що служить для зв'язку зовнішньої програми з web- сервером. Сам протокол розроблено таким чином, щоб можна було використовувати будь-яку мову програмування, яка може працювати зі стандартними пристроями вводу/виводу. Оскільки така можливість є на рівні операційної системи, то, якщо не потрібно складного скрипту, його можна оформити у вигляді командного файлу.

Розглянемо основні особливості технології CGI:

- CGI не висуває особливих вимог до платформи і web-сервера, тому працює на всіх популярних платформах і web-серверах. Також технологія не прив'язана до конкретної мови програмування і може використовуватись будь-якою мовою, що працює зі стандартними потоками вводу/виводу;

- продуктивність CGI-додатків не є високою. Основною причиною цього є те, що при черговому звертанні до сервера для роботи CGI-програми створюється окремий процес, що вимагає великої кількості системних ресурсів;
- технологія не передбачає вбудованих засобів масштабування, про це розробникам необхідно піклуватися окремо;
- CGI-програма є готовим до виконання файлом, що перешкоджає розширенню системи.

На цей час розробники віддають перевагу більш розвиненим та зручним платформам, які відрізняються більшою продуктивністю. Однак велика маса вже працюючих додатків вимагає брати до уваги технологію CGI.

Далі розглянемо найбільш популярні технології — PHP, JSP, Java Servlets, ASP.NET.

4. Технологія сервлетів

Технологія *Java Servlets* (*сервлети)* була розроблена компанією Sun Microsystems, щоб використовувати переваги платформи Java для вирішення проблем технології CG1- та API-роз- ширень сервера.

Технологія вирішує проблему продуктивності, виконуючи всі запити в одному процесі. Сервлети також можуть легко розділяти ресурси і не залежать від платформи, оскільки виконуються всередині JVM (*Java Virtual Machine* — віртуальна машина Java).

Технологія відрізняється широкими функціональними можливостями. Велика кількість бібліотек надає різноманітні засоби, необхідні в розробці програм.

Модель безпеки Java надає можливість точного управління рівнем доступу, наприклад дозволяючи доступ тільки до певної частини файлової системи. Обробка виключень Java робить сервлети більш надійним засобом, ніж розширення серверів мови C/C++.

Сервлет є класом Java і тому має виконуватися за допомогою JVM так званим *сервлет-контейнером* (servlet container, servlet engine). Сервлет-контейнер завантажує клас сервлета при першому звертанні до нього або відразу при запуску сервера за спеціальною вказівкою. Далі сервлет залишається завантаженим для обробки запитів, поки він не буде вивантажений, або до зупинки контейнера.

Технологія є поширеною і може бути використана з усіма популярними web-серверами (Enterprise Server від Netscape, Microsoft Internet Information Server (IIS), Apache, Java Web Server від Sun тощо).

Програмний інтерфейс дозволяє сервлетам обробляти запити на будь-якому рівні, за необхідності використовувати будь-які низькорівневі дані, такі як заголовки запитів, їхній тип тощо. Це забезпечує велику гнучкість для розробки нестандартних оброблювачів, наприклад під час роботи з двійковим або мультимедійним змістом.

Оскільки сервлети обробляються в одному процесі за допомогою створення потоків усередині нього, програмний код сервлетів має бути безпечним для потоку. Це накладає певну відповідальність на програміста, але за допомогою стандартних прийомів, таких як відмова від використання полів у класах сервлетів і збереження необхідних даних у контексті або зовнішньому сховищі, такі властивості коду легко досягаються. До того ж сервлети здобувають таку перевагу, як можливість масштабування.

Отже, сервлети забезпечують компонентний, платформо- незалежний метод для побудови web-додатків без обмежень продуктивності CGI-програм. Вони мають широкий діапазон доступних прикладних API, дозволяють використовувати всі переваги Java, легко розширюються і масштабуються, підтримуються всіма популярними web-серверами. З огляду на це для розробки великих web-систем рекомендується технологія сервлетів.

5. Технологія JSP

Технологія *Java Server Pages (JSP)* від компанії Sun Microsystems

з’явилась як надбудова над технологією Java Servlets. Вона забезпечує більш швидку і просту розробку web-додатків за допомогою застосування *шаблонного підходу.*

Для розуміння архітектури і переваг JSP необхідно знати технологію Java Servlets, оскільки вони тісно пов’язані. Сторінки Java Server Pages є шаблонами сторінок HTML та схожі із шаблонами PHP і ASP. Основною відмінністю від інших подібних технологій є те, що код, який перебуває всередині спеціальних тегів, не інтерпретується при звертанні до сторінки, а попередньо компілюється в Java Servlet. Статичні ділянки шаблона перетворюються у виклики функцій для їхнього розміщення і в потік виводу. Код компілюється так, ніби він перебуває всередині сервлета. Компіляція JSP-сторінок у сервлети є трудомісткою, але проводиться один раз: або при першому звертанні до сторінки, або при запуску сервлет-контейнера.

Технологія JSP вдало поєднує шаблонний підхід до побудови сайтів і всі переваги Java-платформи. Завдяки цьому технологія набула поширення як серед професійних комерційних розробників, так і при створенні відкритих некомерційних проектів.

Важливим кроком до розширення шаблонного підходу стали так звані *бібліотеки тегів (tag libraries).* Вони забезпечують можливість інтегрувати стандартні, сторонні або власні програмні компоненти в сторінки. Простота створення і використання бібліотек тегів стали причиною їх великої популярності. Завдяки роботі на основі Java, технологія JSP не прив’язана до конкретної апаратної або програмної платформи. Тому JSP є вдалим рішенням для використання в гетерогенних середовищах.

Проте продуктивність технології обмежена об’єктивними особливостями архітектури. По-перше, сторінки мають бути відкомпільовані в сервлети, що забирає значний час. По-друге сервлети виконуються в середовищі Java, тобто в режимі інтерпретації. Однак ці обмеження компенсуються додатковими можливостями. Сучасні контейнери

підтримують кластеризацію серверів, що перекладає навантаження на апаратне забезпечення. Це є економічно виправданим рішенням. Задача ж компіляції в сервлети є разовою і виконується або при першому звертанні, або при запуску сервлет-контейнера. У такий спосіб це не позначається на загальній продуктивності системи за достатній період часу.

Основними перевагами JSP є:

- простота розробки, характерна для шаблонного підходу;
- наявність великої кількості сторонніх бібліотек, легкість їхнього використання;
- потужні та різноманітні середовища розробки.

З огляду на зазначене, JSP є перспективною технологією розробки для створення web-сайтів. Однак при створенні складних web-систем даються взнаки обмеження, що накладаються шаблонним підходом.

6. Технологія .NET

Технологія .NET є новою розробкою компанії Microsoft і являє собою новий етап у розвитку засобів взаємодії між додатками. Вона доступна як доповнення .NET Framework до сімейства операційних систем Microsoft Windows, а також у продукті Windows Server 2003. Також ведуться роботи зі створення .NET Framework на інших операційних системах. Платформа .NET спрощує розробку додатків і підвищує надійність коду. Зокрема, вона забезпечує автоматичне керування часом життя об’єктів, доступ до нейтральних до мов бібліотек класів, обробку виключень і налагодження.

Основа .NET — Common Language Runtime (загальне середовище виконання мов) — спирається на системні служби операційної системи і керує виконанням коду, написаного будь-якою сучасною мовою програмування. Набір базових класів дає доступ до сервісів платформи, які розробники можуть використовувати з будь-якою мовою програмування. Common Language Runtime і базові класи разом складають основу .NET

платформи.

NET пропонує високорівневі сервіси:

- ADO.NET — покоління ADO, що використовує XML і SOAP для обміну даними;
- ASP.NET — версія ASP, що дозволяє використовувати .NET-сумісну мову для програмування web-сторінок;
- Windows Forms і Web Forms — набір класів для побудови користувальницького інтерфейсу локальних і web-орієнтованих додатків.

Розгортання систем на платформі .NET здійснюється таким чином. Вихідні коди компілюються не в команди процесора x86 або інші машинні коди. Замість цього компілятор створює код «проміжною мовою Microsoft» (Microsoft Intermediate Language — MSIL). Файл, що містить MSIL, може виконуватися на платформі будь-якого процесора, якщо операційна система надає .NET CLR.

Важливою складовою частиною платформи .NET є середовище ASP.NET (раніше використовувалася назва ASP+). Можливості ASP.NET досить великі, так що її складно назвати наступною версією ASP. В її основі лежить інша платформа, і основними мовами програмування для неї обрані C# і Visual Basic, замість колишніх скриптових мов. Водночас, технологія дозволяє створювати ASP-сторінки будь-якою мовою.

У ASP.NET закладено все для того, щоб зробити весь цикл розробки web-додатка більш швидким, а підтримку — простішою.

Наведемо основні можливості та принципи роботи KSP.NET:

- компілювання коду при першому звертанні; *I*
- широкий вибір бібліотек компонентів, що поставляються з .NET;
- підтримка потужного засобу розробки — Visual Studio.NET;
- мовна незалежність у межах платформ, для яких реалізоване спільне

мовне середовище виконання CLR;

- можливість розширення за допомогою багатопроцесорних та кластерних рішень;
- нові можливості з обробки помилок;
- об’єктно-орієнтовані мови розробки (мова C#);
- розширені можливості повторного використання компонентів.

Очевидно, що платформи .NET і ASP.NET надали нові можливості для розробки web-систем. Вони відповідають всім сучасним вимогам і дозволяють значно прискорити і спростити розробку складних додатків. Проте на даний момент .NET у повному обсязі існує тільки для платформи Windows. Розробки з її переносу на інші системи ще не завершено. Що стосується розробки сайтів, то ASP.NET прив’язана до сервера IIS, і хоч архітектура .NET дозволяє перенести додатки ASP.NET на іншу платформу, але на даний момент реальна можливість відсутня. Тому багатоплатформність поки ще не забезпечується платформою .NET. Слід відзначити, що така система повинна мати можливості інтеграції з платформою .NET (особливо web-сервіси), оскільки її майбутнє широке використання не викликає сумнівів.

7. Технологія PHP

Технологія PHP *(Personal Home Page)* набула значного поширення завдяки своїй безкоштовності і підтримці найпопулярніших платформ. Вона базується на принципі побудови сторінок із шаблонів, що вперше з’явились у Active Server Pages, але розвиває доповнює його.

Сторінки PHP мають вигляд звичайних HTML-сторінок, у яких можуть використовуватися спеціальні теги вигляду <*?php* і ?>. Між тегами вставляються рядки програмного коду спеціальною мовою сценаріїв PHP.

Принцип шаблонів дозволив розроблювачам створювати програми (набагато швидше і без помилок, властивих традиційним CGI-програмам, що

видають HTML-вміст у потік виводу. На сьогоднішній день діапазон систем, побудованих на шаблонах, вельми значний — від простих сторінок з вибірками з бази даних до великих додатків електронної комерції, заснованих на XML.

Шаблонні системи користуються великою популярністю серед розробників, оскільки найбільше підходять для типових сайтів.

Розглянемо основні переваги і недоліки платформи:

- застосовувана у PHP мова — проста і зручна, однак не є в повному обсязі об'єктно-орієнтованою;
- для PHP розроблено великі бібліотеки, а також значна кількість вбудованих функцій для розв'язання найрізноманітніших задач;
- при використанні PHP із web-сервером Apache є можливість ефективного виконання ядра як розширення сервера. В інших випадках продуктивність платформи невисока;
- власних засобів масштабування PHP не має, усі функції з кластеризації цілком лягають на web-сервер і розробників;
- можливості інтеграції обмежені включенням модулів і використанням зовнішніх функцій, що не відповідає сучасним вимогам.

Шаблонний підхід PHP, при усіх великих можливостях, містить і серйозні недоліки. Із загальних недоліків цього підходу, що стосуються як до PHP, так і ASP та JSP, необхідно виділити такі:

- файл-сторінку може підтримувати тільки спеціаліст високої кваліфікації, який добре володіє як програмування, так і HTML;
- один файл в конкретний момент часу може редагувати тільки одна людина. Це означає, що працює або програміст, або дизайнер. Тобто не можна забезпечити поділ праці там, де він потенційно можливий;

- збереження бізнес-логіки у файлах-сторінках у розподіленому за керуючими елементами вигляді призводить до утруднення її винесення в об’єкти другого рівня.

Як загальний підсумок розгляду платформи можна зауважити, що завдяки простоті використання, наявності великого числа функцій і бібліотек, поширеності і підтримці більшості існуючих web-серверів і платформ, PHP є дуже зручним засобом розробки невеликих систем. У той же час обмеження щодо продуктивності, масштабування, за мовою програмування, а також щодо розширення й інтеграції перешкоджають використанню платформи при розробці масштабних систем.

8. Порівняльний аналіз базових технологій серверної сторони

У попередньому матеріалі були розглянуті найбільш популярні базові технології побудови додатків серверної сторони. З розглянутого можна виділити основні підходи до архітектури серверних додатків.

1. *Окреме виконання запитів.* При кожному запиті динамічного вмісту запускається окрема програма для обробки запитів. Програма генерує контент, який передається клієнтові. Цей підхід використовується в класичних CGI-скриптах.

2. *Нагромадження процесів, що виконуються.* Підхід аналогічний попередньому, але якщо запит виконується повторно, то нового запуску програми не відбувається, а обробка передається існуючому процесу. Даний підхід застосовується в технологіях Java Servlets, Fast CGI.

3. *Шаблони сторінок.* При запиті шаблони заповнюються динамічним контентом, який зазвичай (але необов’язково), створюється інтерпретаційною мовою сценаріїв. Підхід застосовується в технологіях ASP, JSP, PHP.

4. *Розширення web-сервера.* Web-сервер звертається до особливих розширень для обробки динамічного змісту. Розширення специфічні для web-

сервера. Цей підхід використовується в IS API, NSAPI, *mod_perl*.

Кожний із розглянутих підходів має свої можливості й обмеження, і, відповідно, свою сферу застосування. Модель окремого виконання запитів істотно обмежує продуктивність. Варіант нагромадження процесів є розвитком цієї технології, підвищує продуктивність, зберігаючи при цьому максимальну гнучкість розробки.

Шаблонний підхід є зручним для розробки невеликих систем, однак у разі збільшення складності він починає гальмувати процес розробки і не є придатним для великих систем.

Він також відрізняється невисокою продуктивністю, хоча дослідження показують, що у визначених умовах може демонструвати досить високі показники і конкурувати з підходом 2.

Розширення web-сервера не є найбільш зручним засобом розробки, оскільки жорстко прив’язують систему до певного web-сервера, але демонструють максимальну продуктивність і дають найбільшу гнучкість у розробці.

Розглянемо платформи згідно з вимогами, визначеними раніше.

CGI не входить в огляд, оскільки є незручним у використанні і має низьку ефективність, а розширення серверів занадто сильно прив’язані до конкретних програмних продуктів.

За схемою обробки запитів платформи розподіляються в такий спосіб:

- PHP-шаблони. У разі виконання на web-сервері Apache інтерпретатор може бути розширенням сервера (в експериментальному режимі IIS);
- Java Servlets. Відрізняються нагромадженням процесів для кожного сервлета;
- JSP-шаблони. При їх обробці виконується попередня компіляція в Java Servlets, що дозволяє використовувати схему нагромадження процесів;

• ASP.NET-шаблони. Використовується схема попередньої компіляції, а не інтерпретації коду. В результаті використовується розширення web-сервера IIS. Можуть використовуватися і низькорівневі оброблювачі.

Основні характеристики платформ для оцінки порівняємо в зведеній таблиці 1, де символ « -» означає повну відсутність підтримки, «-/+» — недостатня підтримка, «+/-» — підтримка не в повному обсязі, і «+» — повна підтримка. Для порівняльних характеристик, таких як мова реалізації або продуктивність, оцінки відповідають ступеню переваги технології.

Таблиця 1 . Порівняння технологій

Характеристики для порівняння	PHP	Java Servlets	JSP	ASP .NET
Багатоплатформність	+/-	+	+	-/+
Продуктивність	-/+	+/-	+/-	+
Масштабованість	-	+	+	+
Мова реалізації	+/-	+	+	+
Можливості розширення й інтеграції	-	+	+/-	+
Простота використання, наявність засобів розробки	+/-	+/-	+	+
Наявність необхідних програмних бібліотек	+	+	+	+
Розподіл дизайну і логіки	+/-	-/+	+/-	+
Засоби візуальної розробки	-/+	+/-	+	+
Можливість побудови компонентної архітектури	-	+	+/-	+

З наведеного порівняння можна зробити висновок, що платформи найбільш популярного типу — шаблонні, — не підходять для розробки великих web-систем, оскільки схема їхньої роботи ускладнює побудову багатокомпонентної архітектури. При використанні систем нешаблонного типу розробка ускладнюється відсутністю можливості швидко і зручно модифікувати дизайн сайту, оскільки він визначається вже сформованим програмним кодом.

Що стосуються візуалізації, то вона присутня тільки при використанні шаблонних платформ, причому винятково при розробці системи. Це призводить до того, що велика web-система, яка вимагає частого поновлення, не повинна будуватися винятково на існуючих базових платформах, а тому є необхідною система управління сайтом, зо поєднує різні підходи.

9. Вступ до PHP

PHP — це мова серверних скриптів (*server scripting language),* що вбудовується в HTML, інтерпретується і виконується па сервері.

PHP є препроцесором HTML. Його робота побудована за схемою, зображеною на Рис. 2.

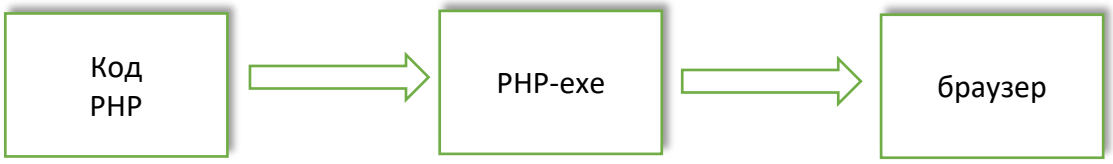

Рис. 2. Схема роботи PHP

До того, як сервер «віддасть» файл браузеру, його переглядає препроцесор-інтерпретатор. Для того, щоб це відбувалося, файли, які підлягають обробці препроцесором, повинні мати певне розширення (зазвичай це *.phtml* або *.php,* але ці значення можна змінити) і містити (хоча і необов'язково) код для препроцесора.

Перед відправленням сторінки PHP-код «програється» на сервері і браузеру видається результат у вигляді HTML-сторінки, яка може дуже відрізнятися від тієї, що зберігається на сервері. Звичайні ж сторінки, що мають розширення *.html* чи *.htm,* web- сервер відправлятиме браузеру без будь-якої обробки.

Основна відмінність від CGI-скриптів, написаних іншими мовами (наприклад, Perl або C) — це те, що в CGI-програмах програміст сам пише HTML-код і, використовуючи PHP, він вбудовує свою програму-скрипт у готову HTML-сторінку, використовуючи теги (наприклад, *<? php i? >*).

PHP називається мовою *серверних скриптів* — на відміну від JavaScript/Jscript/VBScript, які є мовами *клієнтських скриптів.* Це означає, що PHP-скрипт виконується на сервері, а клієнту передається результат його роботи, тоді як в JavaScript код повністю передається на клієнтську машину і тільки там виконується браузером.

Простіше за все роботу *php* показати на прикладі. Такий вигляд має web-сторінка з елементами *php:*

```
<html>
<head>
    <title>Приклад</title>
</head>
    <body>
        <?php echo "Привіт, я PHP-програма!";?>
    </body>
</html>
```

Після виконання цього скрипту отримаємо сторінку, в якій буде написано:

Привіт, я PHP-програма!

Відкривши вихідний текст даної сторінки, ми побачимо таке:

```
<html>
<head>
    <title>Example</title>
</head>
<body>
    Привіт, я PHP-програма!
```

</body>

</html>

Можливості PHP

Коротко кажучи, мовою PHP можна зробити все, що можна зробити за допомогою CGI-програм. Наприклад, обробляти дані з форм, генерувати динамічні сторінки, одержувати і посилати кукіси (cookies). До того ж у PHP передбачена підтримка багатьох баз даних (БД), що робить написання web-додатків з використанням БД дуже простим.

На додаток до *всього php* «розуміє» протоколи IMAP, SNMP, NNTP, POP3 і HTTP, а також має можливість працювати з сокетами *(sockets)* і спілкується за іншими протоколами.

Стисла історія PHP

Початком *php* можна вважати осінь 1994 року, коли *Rasmus Lerdorf* вирішив поширити можливості своєї домашньої сторінки і написати невеликий движок для виконання найпростіших задач. Такий движок був готовий до початку 1995 року й отримав назву Personal Home Page Tools (звідси і скорочення PHP). Його можливості були досить обмежені.

До середини 1995 року з’явилася друга версія, що називалася PHP/FI Version 2. Приставка FI приєдналася з іншого пакета Rasmusa, що обробляв форми (Form Interpreter HP/FI компілювався всередину Apache і використовував стандартний API Apache). Виявилося, що PHP-скрипти працюють швидше за аналогічні CGI-скрипти, бо серверу не було необхідності породжувати новий процес. Мова PHP за можливостями наблизилася до Perl, найбільш популярної мови для написання CGI-програм. Була додана підтримка багатьох відомих баз даних (наприклад, MySQL і Oracle). Інтерфейс до GD-бібліотеки дозволяв генерувати gif-зображення «на

льоту». З цього моменту почалося широке розповсюдження PHP/FI.

Наприкінці 1997 Zeev Suraski і Andi Gutmans вирішили переписати внутрішній движок з метою виправити помилки інтерпретатора і підвищити швидкість виконання скриптів. У 1998 році вийшла нова версія, що була названа *php3,* яка здобула дуже широку популярність серед розробників. На цей час поточна версія — *php5.*

Обґрунтування вибору PHP

Розробникам web-додатків немає необхідності говорити, що web-сторінки — це не тільки текст і картинки. Гідний уваги сайт повинен підтримувати деякий рівень інтерактивності з користувачем: пошук інформації, продаж продуктів, конференції тощо. До недавніх часі все це традиційно реалізувалося CGI-скриптами, написаними мовою Perl. Але виявилося, що CGI-скрипти дуже погано масштабуються. Кожний новий виклик CGI-скрипту вимагає від ядра породження нового процесу, а він займає процесорний час і витрачає оперативну пам’ять.

PHP пропонує інший варіант — він працює як частина web- сервера, і тим самим схожий на ASP від Microsoft або ColdFusion від Allaire.

Синтаксис *php* дуже схожий на синтаксис C або Perl. Спеціалісти, які знайомі з програмуванням, дуже швидко зможуть почати писати програми мовою *php.* У цій мові немає точної типізації даних і немає необхідності в діях щодо виділення чи вивільнення пам’яті.

Програми, написані мовою PHP, на відміну від Perl-програм читаються досить легко.

Пара *PHP — MySQL* є крос-платформною. Це означає, що можна, працюючи в Windows, розробляти програми, при значені для роботи під Unix. Крім того, PHP може працювати яг зовнішній CGI-процес або як звичайний інтерпретатор скриптів або як модуль, що приєднується до web-сервера

Apache або IIS І нарешті, оскільки даний продукт розробляється спільними зусиллями багатьох компаній, існує велика кількість документації і списків розсилки, до яких можна звернутися у випадку виникнення будь-яких питань. Знайдені помилки виправляються досить швидко, усі пропозиції і зауваження розглядаються і, якщо вони виявляються цінними, реалізуються в новій версії.

Використання PHP

Перша PHP-сторінка. Створимо текстовий файл під ім'ям *test.php* (вихідний код першої php-сторінки):

```
<html>
<body>
<?php
        $myvar="Hello, World";
        echo $myvar;
?>
</body>
</html>
```

Тепер відкриємо браузер і наберемо в ньому URL створеної сторінки, наприклад *http://localhost/test.php*. На екрані в браузері побачимо таке:

Hello, World

Відкривши вихідний код сторінки в браузері, побачимо таке:

```
<html>
<body>
        Hello, World
</body>
```

</html>

Це відбулося тому, що PHP-движок на сервері продивився сторінку, знайшов в ній PHP-код, обробив його і видав результат, що web-сервером був відправлений до браузера.

Це вже не статична HTML-сторінка з фіксованим текстом, а справжня програма, ще залежно від поданих до неї даних може видавати різноманітні результати. Ця можливість цілком змінює підхід до публікації документів у Інтернеті, перетворюючи їх зі статичних в динамічні, що змінюються в залежності від дій користувача. В нашій першій сторінці-програмі ці можливості звичайно не такі вже й потужні, адже єдине, що вона робить — це виводить текст, який було б простіше написати вручну. Тим не менше вже через декілька сторінок стає зрозуміло, наскільки гнучкий і потужний інструмент з'явився в руках у розробника. Розглянемо детальніше нашу першу PHP-сторінку.

Перше, на що треба звернути увагу в наведеному вище коді, це обмежувачі. Знайдіть рядок, що починається з *<?php*.

Для PHP-движка цей код означає початок блоку команд, які треба обробити і виконати. Закінчується блок обмежувачем ?>. Іншими словами, символи *<?php і ?>* виконують роль дужок. Все, що знаходиться поза ними, PHP-движок пропускає і відправляє у Web без; будь-якої обробки, виконуючи лише тільки те, що перебуває всередині цих дужок.

Потужність PHP полягає в тому, що PHP-код можна вставляти в будь-яке місце HTML-сторінки. Пізніше ми розглянемо деякі з корисних прийомів.

Замість дужок *<? php ...? >* можна використати і скорочену нотацію <? ...?>.

Деяким програмістам, що працюють також з ASP, зручніше писати дужки, використовуючи комбінацію <% ... %>. PHP можна налагодити на використання й таких дужок.

Можливий ще один з варіантів вставки PHP-сценарію, показаний нижче:

< SCRIPT LANGUAGE=’PHP’>

Інструкції

<SCRIPT>

Ще одна деталь, на яку слід звернути увагу, — це крапка з комою в кінці кожного рядка коду. Це роздільник, призначений для відокремлення одного набору команд від іншого. Взагалі весь PHP-код можна писати в одному рядку, поділяючи команди крапкою з комою. Але читати такий код буде незручно, тому в наших прикладах після кожної крапки з комою ми ставили два порожніх рядки, щоб ясніше виділити групи команд. Не забувайте про крапку з комою в кінці рядка (це найчастіша помилка у програмістів-початківців).

Нарешті, можна помітити, що перед словом *myvar* є символ *$* (долар). Цей символ повідомляє PHP, що перед ним — змінна. Ми присвоїли (використовуючи символ «=») рядок *"Hello, World"* змінній *$myvar*. Змінні, окрім рядків, можуть містити числа і масиви. В будь-якому випадку будь-яка змінна завжди позначається символом *$*.

Функції

Широкі можливості мови PHP містяться в його функціях. Взагалі, *функція* — це блок команд, що виконує яку-небудь операцію. Якщо скомпілювати PHP з усіма наявними для нього доповненнями, то можна отримати доступ до більш ніж 700 функцій.

Кожна функція має свою назву і синтаксис. У першому прикладі була використана функція *echo,* яка виводить рядок, укладений у лапки, або змінну, що іде слідом.

Розглянемо уважніше першу PHP-сторінку. В принципі, вона не відрізняється від звичайної HTML-сторінки. Тільки замість розширення *.html*

(або *.htm)* ми присвоїли їй розширення *.php.* Для web-сервера це розширення стало сигналом, що дану сторінку перед відправленням треба пропустити через PHP-движок. Рядки *<html><body> ... </body></html>* будуть проігноровані PHP-движком. Він зверне увагу тільки на те, що написане всередині дужок *<?php ...?>*. У результаті ми отримаємо: *Hello, World. В*принципі, всю HTML-сторінку ми могли б згенерувати з допомогою PHP-команд, наприклад:

```
<? php
        echo "<html>";
        echo "<body>";
        $myvar="Hello World";
        echo $myvar;
        echo "</body>";
        echo "</html>";
?>
```

Результат був би той же. Однак з точки зору програмування взагалі, цей код — неякісний, адже PHP-движку доведеться обробити вже шість рядків коду замість колишніх двох. Навіщо утрудняти PHP-движок виводом тегів *<html>, <body>, </body>, </html>,* якщо вони й так виводяться в сторінці? При написанні коду слід пам’ятати про продуктивність. Завжди необхідно намагатися покращити або змінити код так, щоб PHP-движок витрачав якомога менше часу на його обробку. Деякі поради з оптимізації коду наведемо декілька пізніше.

10. Використання PHP

Використання будь-яких сценарних (скриптових) мов програмування зазвичай зводиться до використання вбудованих та створення власних функцій.

Розглянемо, як ефективно використовувати PHP.

Однією з дуже корисних функцій мови PHP є *phpinfo().* Вона виводить на екран всю інформацію про PHP-движок, встановлений на сервері, причому в зручній формі таблиці.

Створимо сторінку з наведеним нижче кодом і викличемо її з браузера:

```
<html>
    <body>
    <? php
       phpinfo ();
    ? >
    </body>
</html>
```

У результаті одержимо сторінку, яка містить корисні відомості не тільки про сам PHP-движок, але й про web-сервер, його розширення і можливості.

Виведення тексту в HTML-сторінку

Найпростіший спосіб спілкування з користувачем через web-сторінку — надіслати йому в сторінці необхідний текст. Це можна зробити за допомогою функції *print* або *echo:*

```
<?php print "Hello, world.";?>
```

```
<? Php echo "Hello, world."; ? >
```

Print — це функція, що відправляє браузеру текст. Між словом «print» і символом «;» буде розміщений рядок, що обмежується лапками. Все, що знаходиться всередині лапок, буде відправлене браузеру.

Отже, текст «Hello, World» можна відправити декількома способами. Створимо файл *print.php* з таким текстом:

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

```
<? php
print 'Тут використовується функція print";
print "<p>";
echo "А тут використана функція echo.",
“”,
"P. S. Можна додати другий текстовий рядок",
“”,
"Текстові рядки поділяються комами.";
print "<p>";
printf ("Тут використовується функція printf. ");
print "<p>";
printf ('Функція printf звичайно використовується для форматованого виводу чисел і рядків.");
print "<p>";
printf ("Не забувайте про дужки, коли користуєтеся функцією printf. ");
? >
</html>
</body>
```

У результаті виконання коду в браузері буде відображений такий результат:

Тут використовується функція print.

А тут використана функція echo. P.S. Можна додати другий текстовий рядок. Текстові рядки поділяються комами.

Тут використовується функція printf.

Функція printf звичайно використовується для форматованого виводу чисел і рядків.

Не забувайте про дужки, коли користуєтеся функцією printf.

Таким чином, функція *print* — найпростіший засіб відправлення тексту в браузер.

Функція *echo* працює так же, як і *print,* однак дозволяє додавати до першого текстового рядка інші рядки, поділяючи їх комами.

Функція *printf* відображає числа в певному форматі, наприклад, виводить дробове число з певною кількістю нулей після коми, тому в функції *printf* використання дужок обов'язкове.

Працюючи з дужками, слід узяти до уваги таке:

- *echo* ніколи не використовується з дужками;
- *printf* завжди використовується з дужками;
- *print* використовується з дужками і без них.

Робота з формами

Покажемо, як у PHP обробляти дані, отримані від HTML- форм. Від читача вимагаються міцні знання мови HTML і принципів роботи HTML-форм, а також розуміння різниці між двома засобами передачі даних у них *(див.* методи GET та POST).

Найчастіше серверні скрипти використовуються для обробки результатів заповнення форм. Наприклад, у гостьовій книзі відвідувач вводить дані до форми, яка після цього обробляється на сервері. Відповідаючи на будь-яке опитування, користувач встановлює значення певних полів форми.

Нагадаємо, які теги й атрибути повинна містити форма:

<FORM NAME = 'ім'я_форми'

ACTION = 'шлях_до_оброблювача_форми'

METHOD = ‘метод_передачі_змінних'

Поля введення...

</FORM>

Передусім розберемося, що таке «оброблювач». Це скрипт на сервері, до якого будуть передані значення полів вводу.

Кожне поле вводу має атрибут *NAME*, що буде переданий до оброблювача разом зі своїм значенням. Існує два методи передачі даних: GET і POST. їхня відмінність полягає в тому, що при використанні методу GET значення полів приєднуються до URL, зазначеного в атрибуті *ACTION*. Відбувається це таким чином:

http://site.domain/action.php?Ім’я=значення&... Ім’я=значення

Пари *«Ім’я=значення»* створюються для кожного елемента вводу, ім’я якого вказане атрибутом *NAME*.

У випадку використання методу POST значення полів передаються в заголовку запиту до сервера. Формат передачі при цьому способі нам взагалі не цікавий. Просто візьмемо до уваги, що значення передаються «непомітно» для звичайного користувача.

У процесі виконання скрипту мовою PHP створюються змінні з іменами, які відповідають іменам полів.

Припустимо, що ми створили форму такого вигляду:

```
<FORM ACTION='mult.php' METHOD=’GET’>
<INPUT TYPE='text’  NAME=’first'  SIZE='4'  MAXLENGTH='4‘>
<INPUT TYPE='text’  NAME='second'  SIZE='4'
      MAXLENGTH='4'>
<INPUT TYPE='submit' VALUE='Помножити'>
```

</FORM>

Скрипт, що міститься в файлі *mult.php,* може мати такий вигляд:

<? php

header (’Content-'type: text/html");

echo "$first помножити на $second дорівнює", $first*$second;

? >

Контрольні запитання

1. Дайте порівняння сучасним технологіям серверної сторони.
2. Які особливості технології CGI?
3. Які особливості технології сервлетів?
4. Які особливості технології JSP?
5. Які особливості технології .NET?
6. Які особливості технології PHP?
7. У чому переваги PHP?