*Лекція № **13**.*
ТЕЗИ

Веб-сервер Apache (продовження). Віддача динамічного контенту

Відображення URL на файлову систему.

Згідно оснвного призначення httpd сервера, найважливішим його завданням є ретрансляція адреси ресурсу в назву файлу (чи програми). Див. попередню лекцію. Отже, одними з основних директив є наступні:

DocumentRoot шлях_до_директорії	Центральна директива в конфігу як самого сервера так і кожного вірт хоста. Вкзаує на місце в файловій системі, яке є входом в дерево директорій, видиме для веб-сервера
Alias URL-path file-path|directory-path напр. Alias /image /ftp/pub/image	“мапить” (відображає) URL на певне місце в файловій системі. Використовується як правило для директорій з-поза DocumentRoot
Redirect [status] URL-path URL ***RedirectPermanent*** URL-path URL status: permanenet, temp etc. Напр. Redirect permanent /one http://example.com/two	Редірект (перенаправлення) URL на інший ресурс. Статус має значення.
ScriptAlias URL-path file-path|directory-path	Відображає URL на файлову систему та ініціює директорію як таку, що містить виконувані сервером файли (програми, які повинен виконати сервер для віддачі клієнту результату їх виконання) Як правило ці директорії мають ще й спеціальні опції для безпечного виконання цих програм (наскільки це можливо).
UserDir directory-filename Напр. Запит http://www.foo.com/~bob/one/two.html буде транслюватися в залежності від цієї директиви наприклад так: UserDir public_html ~bob/public_html/one/two.html	Визначає домашні директорії користувачів. Небезпечна директива. Якщо цього непотрібно явно – уникати дозвляти користувачам викладати свої файли на веб. UserDir disabled
RewriteCond , ***RewriteMatch*** та інші директиви модулю ***mod_rewrite*** RewriteCond %{REQUEST_FILENAME} !-f	Складні правила перезапису URL. Часто використовуються і є дуже ефективними.

RewriteCond %{REQUEST_FILENAME} !-d RewriteCond %{REQUEST_FILENAME} !-l RewriteRule .* index.php [L]	
Інші директиви типу ***ErrorDocument, AliasMatch, ScriptAliasMatch*** etc.	

Віддача динамічного контенту

Динамічним - в контексті роботи веб-сервера - ***називається контент, який не існує в явному вигляді у файлі десь на сервері, а продукується сервером*** (з або і без допомоги сторінніх модулів чи програм) "***на льоту***".
Динамічний контент також характеризується непостійністю в тому сенсі що ***для одного і того ж URL*** в залежності від багатьох факторів ***сервер може віддавати зовсім різний контент***.
Такими факторами можуть бути наприклад попередньо передані клієнтом на сервер дані, інформація з заголовків запиту (напр. версія броузера, ОС), та навіть і просто час, коли запит робиться до сервера.

Віддача динамічного контенту з точки зору роботи сервера може здійснюватися кількома способами:

1. шляхом відпрацювання на сервері сторонньої програми,
2. шляхом відпрацювання певного DSO модулю в межах самого сервера.

Ці підходи є різними по-суті.

Прграми, які працюють на стороні веб-сервера називають cgi-програмами, веб-програмами, серверними сценаріями тощо.

CGI-програмами можуть бути як виконувані зкомпільовані файли, так і всеможливі скріпти (php, perl, python, shell).

Найчастіше для написання cgi-програм використовують скріптові мови. Це зумовлено в першу чергу порівняно частою зміною цих програм, а також тим, що з скріптами легше працювати і легше їх розуміти багатьом людям, дотичним до роботи веб-сайту (дизайнерам, системним адміністраторам хостера і т.п.)

Серверні сценарії діляться на такі, які включаються в HTML-файли і такі, які генерують HTML документ повністю.
Приклад першого типу — ***php (mod_php), jsp — Java Server Pages, технологія SSI (Server Side Includes).***
Другого — ***програми CGI, Java Servlets.***
При складній серверній аплікації другий підхід має більше переваг з причин меншого навантаження на сервер: модуль не повинен розбирати (parsing) кожен файл.

CGI

Історично першою технологією для віддачі динмічного контенту була технологія ***CGI — Common Gateway Interface.***

CGI дозволяє серверу виконувати зовнішні програми, які генерують HTMLкод (і звголовк в т.ч.). Цей код веб-сервер відправляє назад клієнту.

Як правило на сервері (напр для певного чи для кожного вірт. хоста) виділяється окрема директорія, в якій веб-серверу дозволяється виконання cgi-програм.
Ця директорія як правило конфігурується з дотриманням відповідних норм безпеки.

Як правило в cgi-bin директоріях забороняють як побудову автоіндексів, так навіть і віддачу звичайного статичного контенту.
Це робиться для того, щоб не можна було зтягнути програму як таку: можна отримати лише результат роботи програми (згенерований нею HTML).
Зтягнувши саму програму зловмисник по-перше може її використати у власних цілях, а по-друге може отримати надто багато інформації про роботу програми на сервері, що є вкрай небажаним з точки зору безпеки.

ScriptAlias <URL-path> <directory-path> — відповідна директива.
Напр.
ScriptAlias /cgi-bin/ /web/cgi-bin/

Інший спосіб для організації cgi-bin директорії — використання директиви Options:

```
<Directory /home/site/htdocs/somedir>
    Options +ExecCGI
</Directory>
```

Якщо ж хочемо працювати тільки з певними розширеннями, то можна додати ще таку директиву:

```
AddHandler cgi-script .cgi .pl
```

Приклад елементарної cgi-програми:

```
#!/usr/bin/perl
print "Content-type: text/html\n\n";
print "Hello, World.";
```

Динамічний контент на основі DSO модулів.

Як альтернатива CGI сервер може бути доповнено ***модулями (DSO) підтримки мов сценаріїв***.
Такий модуль забезпечує середовище для виконання сценарію а також надає API для доступу до зовнішніх джерел даних а також до даних, отриманих від клієнта.
В apache кожен такий модуль реєструється як окремий MIME-тип і окремо визначається розширення файлів, які будуть оброблятися цим модулем.

Такий підхід до віддачі динамічного контенту є зараз суттєво поширенішим (перший домінував раніше).
Причини — значне збільшення долі динамічного контенту в порівнянні із статичним (навіть найпростіші сайти зараз стараються використовувати CMS - Content Management System), а модульний підхід дає виграш як в часі так і в використанні ресурсів сервера.

На сьогоднішній день існує багато модулів сторонніх розробників для підтримки різноманітних мов сценаріїв. Наприклад дуже поширеними є mod_php та mod_perl.

Ще один модуль, що широко використовувався раніше і менше використовується зараз: mod_ssi — Server Side Includes.

Конфігурація у випадку DSO модуля може виглядати приблизно так:

```
LoadModule php5_module  libexec/apache22/libphp5.so
AddType application/x-httpd-php .php
php_admin_flag register_globals Off
php_admin_flag safe_mode On
DirectoryIndex index.php index.html index.htm default.html default.htm
```

Всі ці директиви мають суттєве значення. ПОЯСНИТИ.

Особливістю DSO-підходу до віддачі динамічного контенту є те, що всі сценарії виконуються від одного і того ж користувача (від якого працюють робочі екземпляри веб-сервера) якщо тільки не використовується технологія ***suexec***.
Це безперечно можна віднести до мінусів цієї технології. Власне для захисту від несанкціонованого доступу до даних на рівні файлової системи були придумані такі речі як наприклад safe_mode (захищений режим) в php.
В цьому режимі виконання скріптів можна обмежити в доступі до інших файлів на сервері (як за власником так і за піддеревом).

В будь-якому випадку підключення стороннього модуля для обробки скріптів вимагає скурпульозного вивчення документації до нього. Системний адміністратор повинен розуміти потенційну небезпеку, пов'язану з виконуваними файлами на сервері і розуміти, що він відповідає за безпеку даних як всіх користувачів так і системи вцілому.

Ще однією великою темою, яку однак не будемо розглядати в межах даного курсу є технології і проблеми ***кешування контенту***. Важливо просто знати, що такі технології є і використовуються досить широко.

Хостінгова платформа LAMP.

LAMP — Linux Apache MySQL PHP — найпоширеніша на даний момен хостінгова платформа у світі.

У випадку купівлі хостингового пакету користувач отримує:

- ftp-доступ до певного простору у файловій системі на сервері провійдера,
- право виконання php-сценаріїв (скріптів) через веб з цього простору
- трансляцію URL-ів певного доменного імені на файли в цьому просторі
- доступ до бази даних MySQL (як правило тільки із скріптів на сервері).

Опціонально можуть надавітися також інші послуги, наприклад:

- використання директив різного виду в локальних .htaccess файлах
- доступ до журнальних файлів сервера
- аналзітори журнальних файлів сервера (можливість перегляду статистичної інформації про відвідуваність)
- певні типові сценарії і т.п.

При організації LAMP-хостінгу один сервер обслуговує сотні і тисячі віртуальних хостів. Дешевизна — одна з основних переваг LAMP.

Однак особливу увагу слід приділяти безпеці: різні люди, з різним рівнем конпетенції, з різною метою викладають на сервер сценарії різного походження. Все це повинно безпечно працювати в рамках наявних ресурсів.
Для цього існує ряд підходів, таких як

- safe_mode, safe_mode_gid (див вище)
- обмеження на використання дискового простору
- обмеження пам'яті на користувацький процес
- обмеження часу, відведеного на відпрацювання сценарію
- обмеження по кількості запитів до віртуального хоста (по часу)
- обмеження по трафіку на віртуальний хост
- обмеження на використання бази даних (к-сть запитів, тривалість, розмір і т.д.)
- suexec
- chroot (крайня міра, достатньо ресурсоємка і дорога)

є й інші.

Інакше кажучи слід розуміти, що послуга хостінгу є як і однією з найпоширеніших послуг Інет, так і однією з найскладніших з точки зору адміністрування.