МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ ПОЛІТЕХНІЧНИЙ УНІВЕРСИТЕТ
ХЕРСОНСЬКИЙ ПОЛІТЕХНІЧНИЙ КОЛЕДЖ

МЕТОДИЧНІ НАСТАНОВИ (ВКАЗІВКИ) ДО ВИКОНАННЯ КУРСОВОГО ПРОЕКТУ З ДИСЦИПЛІНИ «WEB-ПРОГРАМУВАННЯ»

для студентів спеціальності
122 «Комп'ютерні науки»
Галузь знань: 12 «Інформаційні технології»

Рекомендовано до друку методичною радою
Херсонського політехнічного коледжу
Одеського національного політехнічного університету
Протокол №___ від _____2019 р.

Рекомендовано до друку Вченою радою
Одеського національного політехнічного університету
Протокол №___ від _____2019 р.

Херсон – 2019

Методичні настанови (вказівки) до виконання курсового проекту з дисципліни «Web-програмування» / Уклад.: В.М. Левицький, Ю.В. Арбузова — Херсон: ХПТК ОНПУ, 2019. — 67с.

Виробничо-практичне видання

МЕТОДИЧНІ НАСТАНОВИ (ВКАЗІВКИ)
ДО ВИКОНАННЯ КУРСОВОГО ПРОЕКТУ
З ДИСЦИПЛІНИ
«WEB-ПРОГРАМУВАННЯ»
для спеціальності
122 «Комп'ютерні науки»
Галузь знань: 12 «Інформаційні технології»

Укладачі: **Левицький Віктор Миколайович**, викладач спецдисциплін, спеціаліст II категорії
Арбузова Юлія Вікторівна, викладач спецдисциплін, спеціаліст вищої категорії, викладач-методист
Коректор: **Ботвинюк Оксана Василівна**, викладач гуманітарних дисциплін, спеціаліст першої категорії
Рецензент: **Сафонов Михайло Сергійович**, кандидат технічних наук викладач спецдисциплін, спеціаліст вищої категорії, завідувач кафедри комп'ютерної інженерії та інженерії програмного забезпечення

За редакцією укладачів
Надруковано з оригінал-макета замовника

Підп. до друку 30.01.2019. Формат 600 х 840 М 1/16.
Папір офсетний. Ум. друк. арк. 1,98. Гарнітура Times.
Спосіб друку – ризографія. Тираж 4 прим. Зам. № 178

Лабораторія організаційно-видавничої діяльності ХПТК ОНПУ
73000, м. Херсон, вул. Небесної Сотні, 23
тел. (0552) 22-55-38, тел./факс (0552) 22-27-43

ЗМІСТ

ВСТУП

Курсовий проект з дисципліни «Web-програмування» виконується студентами третього курсу спеціальності 122 «Комп’ютерні науки та інформаційні технології» після закінчення вивчення вище зазначеного курсу в V семестрі.

При виконанні курсової роботи студенти повинні досконало дослідити предметну область однієї з запропонованих тем. На основі аналізу предметної області.

В результаті виконання всіх етапів проектування повинен бути розроблений інформаційний web-портал, або програмний додаток та технічна документація до неї.

Темами курсових робіт є задачі різноманітних галузей народного господарства та розрахункові задачі, для вирішення яких широко використовуються принципи об’єктно-орієнтованого програмування. Об’єктний підхід розвиває у студентів вміння та навички будувати зв’язки між різними модулями, працювати та приймати рішення.

ЛЕКЦІЯ 1 ВВЕДЕННЯ В ВЕБ-ТЕХНОЛОГІЇ: СТРУКТУРА ТА ПРИНЦИПИ ВЕБ

Анотація

Інтернет: поняття, історія розвитку. Стандартизація в Інтернет. RFC-документи. Стек протоколів TCP / IP. Система доменних імен DNS. Структура і принципи WWW. Проксі-сервери. Протоколи Інтернет прикладного рівня.

План лекції

1) Що таке Інтернет?
2) Стек протоколів TCP/IP
3) Система доменних імен DNS
4) Структура і принципи WWW

1.1 Що таке Інтернет?

Інтернет це — найбільша в світі мережу, яка не має єдиного центру управління, але працює за єдиними правилами і надає своїм користувачам єдиний набір послуг. Інтернет можна розглядати як «мережу мереж», кожна з яких управляється незалежним оператором - постачальником послуг Інтернету (ISP, Internet Service Provider).

З точки зору користувачів Інтернет являє собою набір інформаційних ресурсів, розосереджених по різних мережах, включаючи ISP-мережі, корпоративні мережі, мережі та окремі комп'ютери домашніх користувачів. кожен окремий комп'ютер в даній мережі називається хостом (від англійського терміна host).

сьогоднішній Інтернет зобов'язаний своїй появі об'єднаної мережі ARPANET, яка починалася як скромний експеримент в новій тоді технології комутації пакетів. Мережа ARPANET була розгорнута в 1969 р і складалася спочатку

всього з чотирьох вузлів з комутацією пакетів, які використовуються для взаємодії жменьки хостів і терміналів. Перші лінії зв'язку, що з'єднували вузли, працювали на швидкості всього 50 Кбіт / с. Мережа ARPANET фінансувалася управлінням перспективного планування науково-дослідних робіт ARPA (Advanced Research Projects Agency) міністерства оборони США і призначалася для вивчення технології та протоколів комутації пакетів, які могли б використовуватися для кооперативних розподілених обчислень.

1.2 Стек протоколів TCP/IP

Ці протоколи з самого початку орієнтовані на глобальні мережі, в яких якість сполучних каналів не ідеально. Він дозволяє створювати глобальні мережі, комп'ютери в яких з'єднані один з одним самими різними способами від високошвидкісних оптоволоконних кабелів і супутникових каналів до комутованих телефонних ліній. TCP/IP відповідає моделі OSI досить умовно і містить 4 рівня. Прикладний рівень стека відповідає трьом верхнім рівням моделі OSI: прикладного, уявлення і сеансовому.

У мережі дані завжди передаються блоками щодо невеликого розміру. Кожен блок має префіксних частина (заголовок), що описує вміст блоку, і суфіксну, що містить, наприклад, інформацію для контролю цілісності переданого блоку даних.

Назва стека протоколів TCP/IP складається з назв двох різних протоколів. протокол IP (Internet Protocol) являє собою протокол нижнього (мережевого) рівня і відповідає за передачу пакетів даних в мережі. Він відноситься до так званим протоколам датаграмм і працює без підтверджень. Останнє означає, що при його використанні доставка пакетів даних не гарантується і не підтверджується. Чи не гарантується також і те, що пакети досягнуть пункту призначення в тій послідовності, в якій вони були відправлені.

До протоколів мережевого рівня відноситься також протокол міжмережевих керуючих повідомлень ICMP (Internet Control Message Protocol), призна-

чений для передачі маршрутизатором джерелу інформації про помилки при передачі пакета.

Очевидно, що набагато зручніше передавати дані по каналу, який працює коректно, доставляючи все пакети по порядку. Тому над протоколом IP працює протокол передачі даних більш високого (транспортного) рівня - TCP (Transmission Control Protocol). Посилаючи і приймаючи пакети через протокол IP, протокол TCP гарантує доставку всіх переданих пакетів даних в правильній послідовності.

Для ідентифікації мережевих інтерфейсів використовуються 3 типи адрес:

- апаратні адреси (або MAC-адреси);
- мережеві адреси (IP-адреси);
- символьні (доменні) імена.

У рамках IP протоколу для створення глобальної системи адресації, що не залежить від способів адресації вузлів в окремих мережах, використовується пара ідентифікаторів, що складається з номера мережі і номера вузла.

1.3 Система доменних імен DNS

Незважаючи на те, що апаратне і програмне забезпечення в рамках TCP/IP мереж для ідентифікації вузлів використовує IP-адреси, користувачі вважають за краще символьні імена (доменні імена).

Спочатку в локальних мережах з невеликого числа комп'ютерів застосовувалися плоскі імена, що складаються з послідовності символів без поділу їх на окремі частини, наприклад MYCOMP. Для встановлення відповідності між символьними іменами і числовими адресами використовувалися широкомовні запити. Однак для великих територіально розподілених мереж, що працюють на основі протоколу TCP/IP такий спосіб виявився неефективним. Тому для встановлення відповідності між доменним ім'ям і IP-адреса використовується спеціальна система доменних імен (DNS, Domain Name System), яка заснована на створюваних адміністраторами мережі таблиць відповідності.

У мережах TCP/IP використовується доменна система імен, що має ієрархічну (у вигляді дерева) структуру. Дана структура імен нагадує ієрархію імен, використовувану в багатьох файлових системах. Запис доменного імені починається з наймолодшою складової, потім після точки слід наступна по старшинству символьна частина імені і так далі. Послідовність закінчується кореневим іменем, наприклад: company.yandex.ru.

Побудована таким чином система імен дозволяє розділяти адміністративну відповідальність з підтримки унікальності імен в межах свого рівня ієрархії між різними людьми або організаціями.

Сукупність імен, у яких кілька старших складових частин збігаються, утворюють домен імен.

Кореневої домен управляється центральними органами Інтернету: IANA і Internic.

Домени верхнього рівня призначаються для кожної країни, а також для різних типів організацій. Імена цих доменів повинні слідувати міжнародним стандартом ISO 3166. Для позначення країн використовуються дволітерні абревіатури, наприклад ru (Російська Федерація), us (США), it (Італія), fr (Франція).

Для різних типів організацій використовуються трьохбуквені абревіатури:

- net - мережеві організації;
- org - некомерційні організації;
- com - комерційні організації;
- edu - освітні організації;
- gov - урядові організації.

Адміністрування кожного домена покладається на окрему організацію, яка делегує адміністрування піддоменів іншим організаціям. Для отримання доменного імені необхідно зареєструватися у відповідній організації, якій організація InterNIC делегувала свої повноваження за розподілом доменних імен.

В TCP/IP мережах відповідність між доменними іменами та IP-адреса мо-

же встановлюватися як локальними засобами, так і централізованими службами. Спочатку відповідність задавалося за допомогою створюваного вручну на хості файлу hosts.txt, що складається з рядків, що містять пару виду "доменне ім’я - IP - адреса ". Однак з активним ростом Інтернету таке рішення виявилося немасштабіруемим.

Альтернативне рішення - централізована служба DNS, яка використовує розподілену базу відображень "доменне ім’я - IP - адреса ". Сервер домену зберігає тільки імена, які закінчуються на наступному нижче по дереву рівні. Це дозволяє розподіляти більш рівномірно навантаження по вирішенню імен між усіма DNS-сервер. кожен DNS - сервер крім таблиці відображення імен містить посилання на DNS-сервери своїх піддоменів.

Існують дві схеми дозволу DNS-саме.

Нерекурсивна процедура:

DNS-клієнт звертається до кореневого DNS-сервера із зазначенням повного доменного імені; DNS-сервер відповідає клієнту, вказуючи адресу наступного DNS-сервера, який обслуговує домен верхнього рівня, заданий в наступній старшій частині імені; DNS-клієнт робить запит наступного DNS-сервера, який відсилає його до DNS-сервера потрібного поддомена і т.д., поки не буде знайдений DNS-сервер, в якому зберігається відповідність запитаного імені IP-адресою. Сервер дає остаточну відповідь клієнту.

Рекурсивна процедура:

DNS-клієнт запитує локальний DNS-сервер, що обслуговує піддомен, якому належить клієнт; далі Якщо локальний DNS-сервер знає відповідь, він повертає його клієнту Якщо локальний сервер не знає відповідь, то він виконує ітеративні запити до кореневого сервера. Після отримання відповіді сервер передає його клієнту.

1.4 Структура і принципи WWW

Мережа WWW утворюють мільйони веб-серверів, розташованих по всьому світу. Веб - сервер є програмою, що запускається на підключеному до мережі комп'ютері і передавальної дані по протоколу HTTP.

Для ідентифікації ресурсів (часто файлів або їх частин) в WWW використовуються ідентифікатори ресурсів URI (Uniform Resource Identifier). Для визначення місцезнаходження ресурсів в цій мережі використовуються локатори ресурсів URL (Uniform Resource Locator). такі URL -локатори являють собою комбінацію URI і системи DNS.

Доменне ім'я (або IP - адреса) входить до складу URL для позначення комп'ютера (його мережевого інтерфейсу), на якому працює програма веб-сервер.

На клієнтському комп'ютері для перегляду інформації, отриманої від веб-сервера, застосовується спеціальна програма - веб-браузер. Основна функція веб-браузера - відображення гіпертекстових сторінок (веб-сторінок). Для створення гіпертекстових сторінок в WWW спочатку використовувався мову HTML. Безліч веб-сторінок утворюють веб-сайт.

ЛЕКЦІЯ 2 ВВЕДЕННЯ В КЛІЄНТ-СЕРВЕРНІ ТЕХНОЛОГІЇ ВЕБ. ПРОТОКОЛ HTTP

Анотація

Протокол HTTP. Схема HTTP-сеансу. Склад HTTP-запиту. Забезпечення безпеки передачі даних HTTP. Cookie.

План лекції

1) Протокол HTTP
2) HTTP-запит
3) Відповідь сервера
4) Ідентифікація ресурсу

2.1 Протокол HTTP

HTTP (HyperText Transfer Protocol - RFC 1945 року, RFC 2616) - протокол прикладного рівня для передачі гіпертексту.

Центральним об’єктом в HTTP є ресурс, на який вказує URL в запиті клієнта. Зазвичай такими ресурсами є що зберігаються на сервері файли. На відміну від багатьох інших протоколів, HTTP є протоколом без пам’яті. Це означає, що протокол не зберігає інформацію про попередні запити клієнтів і відповідях сервера.

Усе програмне забезпечення для роботи з протоколом HTTP поділяється на три основні категорії:

1) Сервери - постачальники послуг зберігання та обробки інформації (обробка запитів).
2) Клієнти - кінцеві споживачі послуг сервера (відправка запитів).
3) Проксі-сервери для підтримки роботи транспортних служб.

"Класична"схема HTTP-сеанса виглядає так.

1) Встановлення TCP-з’єднання.
2) Запит клієнта.
3) Відповідь сервера.
4) Розрив TCP-з’єднання.

Таким чином, клієнт посилає серверу запит, отримує від нього відповідь, після чого взаємодія припиняється. зазвичай запит клієнта є вимога передати HTML-документ або який-небудь інший ресурс, а відповідь сервера містить код цього ресурсу.

2.2 HTTP-запит

До складу HTTP-запиту, переданого клієнтом серверу, входять наступні компоненти.

1) Рядок стану (іноді для її позначення використовують також терміни рядок-статус, або рядок запиту).
2) Поля заголовка.
3) Порожня стрічка.
4) Тіло запиту.

Рядок стану має такий вигляд:

метод_запроса URL_ресурса версія_протокола_HTTP

Розглянемо компоненти рядка стану, при цьому особливу увагу приділимо методам запиту.

Метод, вказаний в рядку стану, визначає спосіб впливу на ресурс, URL якого заданий в тому ж рядку. Метод може приймати значення GET, POST, HEAD, PUT, DELETE і т.д. Незважаючи на велику кількість методів, для веб-програміста по-справжньому важливі лише два з них: GET і POST.

GET. Згідно формального визначення, метод GET призначається для отримання ресурсу з зазначеним URL. Отримавши запит GET, сервер повинен прочитати зазначений ресурс і включити код ресурсу до складу відповіді клієнту. URL ресурсу може вказувати на виконуваний код програми, який, при

дотриманні певних умов, повинен бути запущений на сервері. У цьому випадку клієнтові повертається не код програми, а дані, згенеровані в процесі її виконання.

POST. Згідно з тим же формального визначення, основне призначення методу POST - передача даних на сервер. Як і у випадку з методом GET, URL, заданий в рядку стану, вказує на конкретний ресурс.

Методи HEAD і PUT є модифікаціями методів GET і POST.

Версія протоколу HTTP, як правило, задається в наступному форматі:

HTTP/версія.модіфікація

Поля заголовка, наступні за рядком стану, дозволяють уточнювати запит, тобто передавати серверу додаткову інформацію. Поле заголовка має такий вигляд:

Ім’я_поля: Значення

Призначення поля визначається його ім’ям, яке відокремлюється від значення двокрапкою. У багатьох випадках при роботі в Веб тіло запиту відсутня. При запуску CGI-сценаріїв дані, що передаються для них в запиті, можуть розміщуватися в тілі запиту.

Нижче представлений приклад HTML-запит, згенерованого браузером

```
GET http://oak.oakland.edu/ HTTP / 1.0
Connection: Keep-Alive
User-Agent: Mozilla / 4.04 [en] (Win95; I)
Host: oak.oakland.edu
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg,
Accept-Language: en
Accept-Charset: iso-8859-1, *, utf-8
```

Отримавши від клієнта запит, сервер повинен відповісти йому. Подібно запитом клієнта, відповідь сервера також складається з чотирьох перерахованих нижче компонентів.

1) Рядок стану.
2) Поля заголовка.
3) Порожня стрічка.
4) Тіло відповіді.

2.3 Відповідь сервера

Відповідь сервера клієнту починається з рядка стану, яка має такий вигляд:

Версія_протокола Код_ответа Поясніtельное_сообщеніе

Версія_протокола задається в тому ж форматі, що і в запиті клієнта, і має таке ж значення.

Код_ответа - це тризначне десяткове число, яке представляє в закодованому вигляді результат обслуговування запиту сервером.

Поясніtельное_сообщеніе дублює код відповіді в символьному вигляді. Це рядок символів, яка не обробляється клієнтом. Вона призначена для системного адміністратора або оператора, що займається обслуговуванням системи, і є розшифровкою коду відповіді.

З трьох цифр, складових код відповіді, перша (старша) визначає клас відповіді, інші дві представляють собою номер відповіді всередині класу. Так, наприклад, якщо запит був оброблений успішно, клієнт отримує наступне повідомлення:

HTTP/1.0 200 OK

Перша цифра не може бути більше 5 і визначає наступні класи відповідей.

1 - спеціальний клас повідомлень, які називаються інформаційними. Код відповіді, що починається з 1, означає, що сервер продовжує обробку запиту. При обміні даними між HTTP-клієнтом і HTTP-сервером повідомлення цього класу використовуються досить рідко.

2 - успішна обробка запиту клієнта.

3 - перенаправлення запиту. Щоб запит був обслужений, необхідно вжити додаткових заходів.

4 - помилка клієнта. Як правило, код відповіді, що починається з цифри 4, повертається в тому випадку, якщо в запиті клієнта зустрілася синтаксична помилка.

5 - помилка сервера. З тих чи інших причин сервер не в змозі виконати запит.

Нижче представлений приклад відповіді сервера на запит, наведений в попередньому розділі. У тілі відповіді міститься вихідний текст HTML-документа.

```
HTTP / 1.1 200 OK
Server: Microsoft-IIS / 5.1
X-Powered-By: ASP.NET
Date: Mon, 20 OCT 2008 11:25:56 GMT
Content-Type: text / html
Accept-Ranges: bytes
Last-Modified: Sat, 18 Oct 2008 15:05:44 GMT
ETag: "b66a667f948c92: 8a5"
Content-Length: 426

<Html>
<Body>
<!-- Тіло HTML -->
.........
</ Body>
</ Html>
```

Поля заголовка і тіло повідомлення можуть бути відсутніми, але рядок стану є обов’язковим елементом, так як вказує на тип запиту/відповіді.

Поле з ім’ям Content-type може зустрічатися як в запиті клієнта, так і у відповіді сервера. Як значення цього поля вказується MIME-тип вмісту запиту або відповіді. MIME-тип також передається в поле заголовка Accept, прису-

тнього в запиті.

Специфікація MIME (Multipurpose Internet Mail Extension - багатоцільове поштове розширення Internet) спочатку була розроблена для того, щоб забезпечити передачу різних форматів даних в складі електронних листів. Однак застосування MIME не вичерпується електронною поштою.

Відповідно до специфікації MIME, для опису формату даних використовуються тип і підтип. Тип визначає, до якого класу належить формат вмісту HTTP-запит або HTTP -Відповісти. Підтип уточнює формат. Тип і підтип відокремлюються одна від одної косою рискою:

тип/підтип

Оскільки в переважній більшості випадків у відповідь на запит клієнта сервер повертає вихідний текст HTML-документа, то в поле Content-type відповіді зазвичай міститься значення text/html. тут ідентифікатор text визначає тип, повідомляючи, що клієнту передається символьна інформація, а ідентифікатор html описує підтип, тобто вказує на те, що послідовність символів, що міститься в тілі відповіді, являє собою опис документа на мові HTML.

Перелік типів і підтипів MIME досить великий. У таблиці 2.4 наведені приклади MIME-типу, найбільш часто зустрічаються в заголовках HTML-запит і відповідей.

2.4 Ідентифікація ресурсу

Для однозначної ідентифікації ресурсів в мережі Інтернет використовуються унікальні ідентифікатори URL.

Однаковий ідентифікатор ресурсу URI (Uniform Resource Identifier) являє собою коротку послідовність символів, що ідентифікує абстрактний або фізичний ресурс. URI не вказує на те, як отримати ресурс, а тільки ідентифікує його.

URL (Uniform Resource Locator) - це URI, який, крім ідентифікації ресурсу, надає ще й інформацію про місцезнаходження цього ресурсу.

URN (Uniform Resource Name) - це URI, який ідентифікує ресурс в певному просторі імен, але, на відміну від URL, URN не вказує на місцезнаходження цього ресурсу.

URL має наступну структуру:

```
<Схема>://<логін>:<пароль>@<хост>:<порт>/<URL-шлях>
```

де:

схема - схема звернення до ресурсу (зазвичай мережевий протокол); логін - ім’я користувача, що використовується для доступу до ресурсу; пароль - пароль, асоційований з вказаним ім’ям користувача; хост - повністю прописане доменне ім’я хоста в системі DNS або IP-адреса хоста; порт - порт хоста для підключення; URL-шлях - уточнююча інформація про місце знаходження ресурсу.

Загальноприйняті схеми (протоколи) URL включають протоколи: ftp, http, https, telnet, а також інші.

ЛЕКЦІЯ 3 СТРУКТУРА WEB-ДОДАТКІВ

Анотація

Програми, що виконуються на клієнт-машині. Програми, що виконуються на сервері. Насичені інтернет-додатки. Введення в Jscript: типи даних, оператори, функції та об'єкти. Коротка характеристика VBScript. Java-аплети. ActionScript - загальна характеристика. XAML і Microsoft Silverlight. Поняття про DOM. DHTML. Регулярні вирази.

План лекції

1) Програми, що виконуються на сервері
2) Програми, що виконуються на клієнті
3) Мови сценаріїв сумісні з ECMA
4) Java-аплети
5) XAML і Microsoft Silverlight

3.1 Програми, що виконуються на сервері

Код програми, що працює на сервері, не передається клієнту. При отриманні від клієнта спеціального запиту, який передбачає виконання такої програми, сервер запускає її і передає параметри, що входять до складу запиту. Засоби для генерації подібного запиту зазвичай входять до складу HTML-документа.

Результати своєї роботи програма оформляє у вигляді HTML-документа і передає їх веб-сервера, а останній, в свою чергу, доповнює отримані дані HTTP-заголовком і передає їх клієнту.

3.2 Програми, що виконуються на клієнті

Одним з типів програм, призначених для виконання на клієнт-машині, є сценарії, наприклад, JavaScript (VBScript). Оригінальний текст сценарію є ча-

стиною веб-сторінки, тому сценарій JavaScript передається клієнту разом з документом, до складу якого він входить. обробляючи HTML-документ, браузер виявляє вихідний текст сценарію і запускає його на виконання.

До всіх програм, які передаються з сервера на клієнт-машини і запускаються на виконання, пред'являється одна загальна вимога: ці програми повинні бути позбавлені можливості звертатися до ресурсів комп'ютера, на якому вони виконуються. Така вимога цілком обґрунтована. Адже передача по мережі і запуск Java-апплетів і JavaScript-сценаріїв відбувається автоматично без участі користувача, тому робота цих програм повинна бути абсолютно безпечною для комп'ютера. Іншими словами, мови, призначені для створення програм, що виконуються на клієнт-машині, повинні бути абсолютно непридатні для написання вірусів і подібних програм.

Насичений інтернет-додаток (Rich Internet application) - ще один підхід, який полягає у використанні Adobe Flash або Java-аплетів для повної або часткової реалізації призначеного для користувача інтерфейсу, оскільки більшість браузерів підтримує ці технології (як правило, за допомогою плагінів).

При використанні Java-апплетів до складу HTML-документа включається спеціальний дескриптор, що описує розташування файлу, що містить код аплету, на сервері. Після того як клієнт отримує HTML-код документа, що включає аплет, він генерує додатковий запит серверу. Після того як сервер пересилає клієнту код аплету, сам аплет запускається на виконання.

При використанні насичених інтернет-додатків доводиться стикатися з такими проблемами:

- необхідність забезпечення безпечного середовища виконання ("пісочниця");
- для виконання коду має бути дозволено виконання сценаріїв;
- втрата в продуктивності (тому що Виконуємо на клієнтській стороні);
- потрібно багато часу на завантаження;

Для розробки насичених інтернет-додатків використовуються пакети Curl, Adobe

Flex і Microsoft Silverlight.

3.3 Мови сценаріїв сумісні з ECMA

JavaScript - інтерпретована мова програмування, стандартизований міжнародною організацією ECMA в специфікації ECMA-262. Мови JavaScript, JScript і ActionScript є розширенням стандарту ECMA-262.

Назва "ECMAScript"стало фактично компромісом між організаціями, залученими в процес стандартизації, зокрема Netscape і Microsoft. Хоча JavaScript і JScript прагнули до сумісності з ECMAScript, вони мають ряд додаткових можливостей не передбачених специфікацією ECMA.

Синтаксис JScript багато в чому аналогічний мови JavaScript, проте, крім додавання клієнтських скриптів на веб-сторінки і деяких інших функцій, JScript може використовуватися і для інших цілей, наприклад:

- автоматизація адміністрування систем Microsoft Windows;
- створення сторінок ASP.

Мова JScript отримала подальший розвиток у вигляді мови JScript.NET, яка орієнтована на роботу в рамках платформи Microsoft.NET

JScript - інтерпретована, об’єктно-орієнтована мова. Хоча він має істотно меншу кількість можливостей, ніж такі об’єктно-орієнтовані мови як C ++ та Java.

Можливості мови істотно обмежені:

- мова не дозволяє розробляти окремі програми;
- сценарії на JScript можуть виконуватися тільки за допомогою інтерпретатора, зокрема веб-браузером.
- JScript - мова без суворого контролю типів. Тому не потрібно оголошувати тип змінних явно.

Крім того, у багатьох випадках JScript виконує перетворення автоматично, коли вони необхідні. Наприклад, при складання рядки і числа, число буде перетворено в рядок.

Код на JScript пишеться в текстовому форматі, і організований в інструкції, блоки, що складаються з пов’язаних наборів інструкцій, і коментарів.

ActionScript - об’єктно-орієнтована мова програмування, один з діалектів EcmaScript, який додає інтерактивність, обробку даних і багато іншого в вміст Flash-додатків. ActionScript виповнюється віртуальною машиною (ActionScript Virtual Machine), яка є складовою частиною програми Flash Player. ActionScript компілюється в байткод, який включається в SWF - файл.

SWF-файли виконуються Flash Player. Сам Flash Player існує у вигляді плагіна до веб-браузеру, а також як самостійне який виконувався додаток. У другому випадку можливе створення виконуваних exe-файлів, коли swf - файл включається у Flash Player.

3.4 Java-аплети

Java - аплет - це програма, написана на мові Java і відкомпілювати в байт-код. Виполнет в браузері з використанням віртуальної Java-машини (JVM). Аплети використовуються для надання інтерактивних можливостей веб-додатків, які неможливі в HTML. Так як байт-код Java від платформи незалежний, то Java -апплети можуть виконуватися браузерами на багатьох операційних платформах.

Java - сервлети є серверними додатками, але вони відрізняються від аплетів мовою, функціями та іншими характеристиками.

призначені Java -апплети для виконання в безпечному середовищі з метою запобігання їх доступу до локальних ресурсів комп’ютера клієнта.

Код аплету завантажується з веб-сервера, і браузер

або вставляє аплет в веб-сторінку; або відкриває окреме вікно з власним призначеним для користувача інтерфейсом аплету.

Аплет може бути впроваджений в веб-сторінку за допомогою використання HTML тега <applet>, або (що рекомендується) тега <object>.

Можна назвати наступні переваги Java-апплетов:

- працюють практично на більшості операційних платформ;
- підтримуються більшістю браузерів;
- кешуються в більшості браузерів, що істотно прискорює їх завантаження при поверненні на веб-сторінку;
- після першого запуску аплета, коли Java-машина вже виконується і швидко запускається, виконання аплетів відбувається значно швидше;
- завантажуються зі швидкістю, порівнянної з програмами на інших компільованих мовах, наприклад C ++, але у багато разів швидше ніж на JavaScript.

При цьому у Java -апплетов є і недоліки:

- потрібна установка Java-розширення, які доступні за замовчуванням не у всіх браузерах;
- проблеми реалізації Java-розширень для 64-розрядних процесорів;
- не можуть запускатися до першого завантаження віртуальної Java-машини, що може займати значний час;
- розробка користувальницького інтерфейсу з використанням аплетів є більш складним завданням в порівнянні з HTML;
- не мають прямого доступу до локальних ресурсів комп'ютера клієнта;
- деякі аплети прив'язані до використання певного середовища часу виконання Java (JRE).

3.5 XAML і Microsoft Silverlight

XAML (eXtensible Application Markup Language) - мова інтерфейсів платформи Windows Vista.

Модель додатків Vista включає об'єкт Application. Його набір властивостей, методів і подій дозволяє об'єднувати веб-документи в пов'язане додаток. Об'єкт Application контролює виконання програми та генерує події для призначеного для користувача коду. Документи додатки створюються за допомогою мови XAML, який описує, перш за все, призначений для користувача інтер-

фейс. Логіка додатка управляється процедурних кодом (C#, VB і ін.). XAML включає основні чотири категорії елементів: панелі, елементи управління, елементи, пов'язані з документом і графічні фігури.

Microsoft Silverlight є офіційною назвою заснованої на XML і. NET технології під кодовим ім'ям WPF / E (Windows Presentation Foundation Everywhere), що є альтернативною Adobe Flash. Являє собою підмножина Windows Presentation Foundation, в якому реалізовані векторна графіка, анімація і засоби відтворення відео. У версії 1.1 включає в себе повну версію .NET CLR - звану CoreCLR, що дозволить розробляти Silverlight додатки на будь-якому з мов. NET. Silverlight v.1.0 містить підключається модуль браузера для обробки XAML і кодеки для відтворення мультимедійного вмісту в форматах WMV, WMA і MP3.

Microsoft Silverlight являє браузеру внутрішню модель DOM, керовану з JavaScript коду. оскільки мова XAML заснований на XML, то документ, що визначає завантаження клієнта призначений для користувача інтерфейс - текстовий і тому цілком придатний для індексування пошуковими системами. Використовуючи модель DOM, JavaScript може динамічно оновлювати вміст Silverlight, аналогічно DHTML.

Також можна викликати методи управління презентацією (запуску анімації або призупинити відтворення відео, наприклад).

Silverlight - додаток починається з виклику об'єкта Silverlight з HTML сторінки, що завантажує XAML файл. XAML файл містить об'єкт Canvas, який виступає підкладкою для інших елементів. Об'єкти XAML здатні генерувати події, перехоплює з JavaScript.

ЛЕКЦІЯ 4 СЕРВЕРНІ ВЕБ-ДОДАТКИ

Анотація

Стандарт CGI. Сценарії. Сценарні мови: класифікація за швидкодією. Мова Python. Мова Ruby. Технологія ASP. Інтерфейс ISAPI.

План лекції

1) Стандарт CGI
2) Обмін даними з CGI-сценарієм

4.1 Стандарт CGI

Коло завдань, що вирішуються Web-сервером, обмежений. В основному він зводиться до підтримки HTTP-взаємодії і доставці клієнту Web-документів. Будь-які "нестандартні"дії реалізуються за допомогою спеціальної програми, яка взаємодіє з веб-сервером і клієнтом. Ця взаємодія підпорядковується певним правилам.

Основний набір таких правил - стандарт CGI (Common Gateway Interface - інтерфейс загального шлюзу), який визначає порядок запуску програми на комп’ютері-сервері, способи передачі програмі параметрів і доставки результатів її виконання клієнту. Програма, написана за правилами CGI, називається CGI-сценаріем (CGI script), хоча це не означає, що на сервері не може виконуватися двійковий файл.

Завдяки цьому інтерфейсу для розробки додатків можна використовувати будь-яка мова програмування, яка має в своєму розпорядженні засоби взаємодії зі стандартними пристроями введення / виводу. Такими можливостями володіють також сценарії для вбудованих командних інтерпретаторів операційних систем.

Виконання будь-якої програми (в тому числі CGI-сценарія) можна умовно

розділити на п’ять етапів.

1) Запуск програми.
2) Ініціалізація і читання вихідних даних.
3) Обробка даних.
4) Висновок результатів виконання.
5) Завершення програми.

Відмінності між CGI-сценаріем і консольним додатком стосуються першого, другого і четвертого етапів виконання.

Кожен раз, коли веб-сервер отримує запит від клієнта, він аналізує вміст запиту і повертає відповідний відповідь:

Якщо запит містить вказівку на файл, що знаходиться на жорсткому диску, то сервер повертає в складі відповіді цей файл;

Якщо запит містить вказівку на програму і необхідні для неї аргументи, то сервер виконує програму і результат її роботи повертає клієнту.

CGI визначає:

яким чином інформація про сервер і запиті клієнта передається програмі в формі аргументів і змінних оточення;

яким чином програма може передавати назад додаткову інформацію про результати (наприклад про тип даних) в формі заголовків відповіді сервера.

Існує два способи розпізнавання файлів, що містять тексти CGI-сценаріев. Перший спосіб полягає в тому, що при установці веб-сервера один з каталогів спеціально виділяється для зберігання сценаріїв. Зазвичай такий каталог отримує ім’я cgi-bin (або Scripts для веб-сервера IIS). В цьому випадку, якщо клієнт запитує файл з каталогу cgi-bin, сервер сприймає такий запит як команду на запуск сценарію. Файли з інших каталогів інтерпретуються як HTML-документи.

Другий спосіб використовує розширення файлу. Під час налаштування сервера вказується, що файли з певними розширеннями містять коди сценаріїв.

Ідентифікація по розширенню використовується відносно рідко. Найчасті-

ше всі сценарії поміщаються в cgi-bin, scripts або в інший каталог, спеціально виділений для їх зберігання.

Висновок результатів виконання CGI-сценарія здійснюється надзвичайно просто. Для того щоб дані були передані клієнту, досить вивести їх в стандартний вихідний потік. Однак, розробляючи CGI-сценарій, не слід забувати про те, що він все ж відрізняється від консольної програми і має такі особливості.

Інформація, передана клієнту, повинна відповідати протоколу HTTP, тобто складатися із заголовка і тіла відповіді. Як правило, отримавши дані від сценарію, сервер самостійно додає перший рядок заголовка.

HTTP / 1.0 200 OK

Формування інформаційних полів, що входять до складу заголовка, - завдання сценарію. Щоб дані, передані сценарієм, були правильно інтерпретовані клієнтом, необхідно, щоб в заголовку присутні як мінімум поле Content-type. За заголовком повинна слідувати порожній рядок. При відсутності полів заголовка реакція браузера буде непередбачуваною. У подібних випадках браузер зазвичай намагається відобразити отриману інформацію як текстовий файл.

Найприродніший формат для браузера - формат HTML. Результати роботи сценарію зазвичай оформляються у вигляді веб-сторінки, тобто повертаються дані слід доповнити дескрипторами HTML. Таким чином, відповідь CGI -сценарія клієнту зазвичай виглядає так:

```
Content-type: text/html

<Html>
<Head>
<Title> відповідь сценарію </title>
</ Head>
<Body>
.....................
```

```
</ Body>
</ Html>
```

Зверніть увагу на порожній рядок після вираження Content-type: text/html. Вона обов'язково має бути присутня у відповіді, в іншому випадку клієнт сприйме всі наступні дані як продовження заголовка.

Після компіляції програми необхідно скопіювати виконуваний файл в каталог cgi-bin (або в інший каталог, призначений для розміщення виконуваних файлів) з якого він може запускатися веб-сервером на виконання за запитом клієнта.

4.2 Обмін даними з CGI-сценарієм

Для виклику даного сценарію досить включити в веб-сторінку наступний фрагмент HTML-коду:

```
<form method = "post" action = "/cgi-bin/hello.exe">
<input type = "submit">
</form>
```

Якщо сценарій викликається з форми, йому передаються ті дані, які користувач ввів за допомогою інтерактивних елементів, що відображаються на веб-сторінці - передача інформації CGI-сценарію здійснюється в два етапи: спочатку браузер передає дані веб-сервера, потім веб-сервер передає їх сценарієм.

У більшості випадків крім кнопки Submit форма містить інші інтерактивні елементи, кожен з яких має ім'я (атрибут NAME) і значення (атрибут VALUE, або послідовність символів, введена користувачем). З імен елементів і їх значень формується рядок параметрів, яка має такий вигляд.

ім'я = значення & ім'я = значення &. . . & Ім'я = значення

кожен параметр являє собою ім'я керуючого елемента та його значення, розділені знаком рівності, а кілька таких пар об'єднують рядок за допомогою символу "&". Якщо до складу імені або значення входить символ "&"або -

то подібні символи кодуються послідовність знака відсотка "% за яким слідують дві шістнадцяткові цифри, що визначають код символу. Так, наприклад, послідовністю "%21"кодується знак оклику "!". Як правило, при передачі параметрів трехсімвольнимі послідовностями замінюються всі знаки, крім латинських букв, цифр і символу пробілу (останній замінюється знаком"+").

Таким чином, перед використанням рядка параметрів її треба декодувати. Алгоритм декодування надзвичайно простий і включає в себе наступні дії:

Виділити з рядка параметрів пари ім’я = значення. Виділити з кожної пари ім’я і значення. У кожному імені і кожному значенні замінити символи "+"пробілами. Кожну послідовність з символу "%"і двох шістнадцятирічних і перетворити в ASCII-символ.

Атрибут method дескриптора <form> має або значення "GET або значення "POST". Значення "GET"і "POST"визначають два різних методу передачі параметрів сценарієм:

Якщо атрибут method має значення "GET рядок параметрів передається разом з URL викликається сценарію. Роздільником між URL і рядком параметрів є символ "?". Якщо атрибут method має значення "POST рядок параметрів передається в тілі HTTP-запиту.

Розглянемо, як повинен вести себе CGI - сценарій, щоб правильно обробити дані в залежності від методу, використаного при передачі даних, рядок параметрів доставляється CGI -сценарію різними способами.

якщо атрибут METHOD дескриптора <FORM> мав значення "GET рядок параметр передається серверу в якості значення змінної оточення QUERY_STRING.

При використанні методу POST дані доставляються сценарієм по-іншому. Вони передаються через стандартний потік введення (STDIN). щоб сценарій зміг визначити, скільки символів слід читати зі стандартного вводу, веб- сервер встановлює значення змінної оточення CONTENT_LENGTH, рівним довжині рядка параметрів.

Отримавши управління, сценарій в першу чергу повинен з’ясувати, з допо-

могою якого методу виконувалася передача параметрів. ця інформація міститься в змінній оточення REQUEST_METHOD.

Таким чином, в простому випадку, щоб виконати обробку рядка параметрів, досить знати призначення трьох змінних оточення: REQUEST_METHOD, QUERY_STRING і CONTENT_LENGTH.

Приклад сценарію на мові Perl, який повертає клієнту рядок параметрів, наведено нижче. Сценарій визначає, який метод використовувався для передачі даних, читає рядок параметрів і передає її клієнту, попередньо доповнивши HTML-дескрипторами.

```
$Method = $ENV{’REQUEST_METHOD’};

if ($ method eq "GET")
{$pars = $ ENV{’QUERY_STRING’};  }
else
{$length = $ENV{’CONTENT_LENGTH’};  }

read (STDIN, $pars, $length);

print "Content-type: text/html \n\n";
print "<HTML> <BODY> \n";
print "<P> METHOD =", $ method;
print "<P> String of parameters: <P> \n";
print $ pars;
print "</HTML> </BODY> \n";
```

ЛЕКЦІЯ 5 МОВИ CGI-СЦЕНАРІЇВ

Анотація

Стандарт CGI. Сценарії. Сценарні мови: класифікація за швидкодією. Мова Python. Мова Ruby. Технологія ASP.

План лекції

1) Perl
2) PHP
3) Python
4) Ruby
5) Server-side JavaScript

5.1 Perl

Perl - високорівнева інтерпретується динамічний мова програмування загального призначення, створений Ларрі Уоллом, лінгвістом за освітою. Назва мови офіційно розшифровується як Practical Extraction and Report Language («практичну мову для отримання даних та складання звітів»).

5.2 PHP

PHP (англ. PHP: Hypertext Preprocessor - «PHP: препроцесор гіпертексту»; спочатку Personal Home Page Tools - «Інструменти для створення персональних веб-сторінок») - скриптова мова загального призначення, інтенсивно застосовується для розробки веб-додатків. В даний час підтримується переважною більшістю хостинг-провайдерів і є одним з лідерів серед мов, що застосовуються для створення динамічних веб-сайтів.

5.3 Python

Python - високорівнева мова програмування загального призначення з акцентом на продуктивність і читаність коду. Мова Python поєднує в собі мінімалізм синтаксису ядра і великий обсяг корисних функцій стандартної бібліотеки.

Python підтримує структурну, об'єктно-орієнтовану, функціональну, імперативну і аспектно-орієнтовану парадигми.

Його основні архітектурні риси:

- динамічна типізація
- автоматичне керування пам'яттю
- повна інтроспекція
- механізм обробки виключень
- підтримка багатопоточних обчислень
- зручні високорівневі структури даних

Код в Python організовується у функції та класи, які можуть об'єднуватися в модулі (які в свою чергу можуть бути об'єднані в пакети).

Для всіх основних платформ Python має підтримку характерних для даної платформи технологій (наприклад, Microsoft COM/DCOM). Існує навіть спеціальна версія Python для віртуальної машини Java - Jython, що дозволяє інтерпретатору виконуватися на будь-якій системі, що підтримує Java, при цьому класи Java можуть безпосередньо використовуватися з Пітона і навіть бути написаними на Python. Кілька проектів забезпечують інтеграцію з платформою Microsoft. NET, основні з яких - IronPython і Python.Net.

5.4 Ruby

Ruby - інтерпретована мова високого рівня для швидкого і зручного об'єктно-орієнтованого програмування. Ruby володіє незалежної від операційної системи реалізацією багатопоточності, суворої динамічною типізацією, "складальником сміття"і багатьма іншими можливостями. Багато особливості синтакси-

су і семантики мови Perl запозичені в Ruby.

Перша загальнодоступна версія Ruby з'явилася в 1995 р

Ruby - повністю об'єктно-орієнтована мова:

Всі дані є об'єктами, на відміну від багатьох інших мов, де існують примітивні типи. Кожна функція є методом. змінні Ruby містять не самі об'єкти, а посилання на них. Присвоєння - це не передача значення, а копіювання посилання на об'єкт. В Ruby можна додавати методи не тільки в будь-класи, а й в будь-які об'єкти. Наприклад, можна додати до деякої рядку довільний метод.

Ruby поставляється з великою стандартної бібліотекою. Це, перш за все, бібліотеки для роботи з різними мережевими протоколами на стороні сервера і клієнта, кошти для роботи з різними форматами представлення даних (XML, XSLT, YAML, PDF, RSS, CSV, WSDL). Також є бібліотеки для роботи з архівами, датами, кодуваннями, матрицями, засоби для системного адміністрування, розподілених обчислень, підтримки багатопоточності і т. Д.

5.5 ASP

ASP (Active Server Pages) - технологія, розроблена компанією Microsoft, що дозволяє легко створювати додатки для Веб.

Програмування на ASP дає розробникам доступ до інтерфейсу програмування додатків Internet Information Server за допомогою мови сценаріїв VBScript і JScript.

ASP працює на платформі операційних систем лінії Windows NT і на вебсерверах Microsoft IIS.

IIS розрізняє код, що виконується на сервері, і вміст, що відправляється клієнтові за допомогою ASP. DLL, аналізуючи файл ASP на наявність початкового «%"і кінцевого "%>"тегів і виконуючи код, розташований між ними, за допомогою WSH.

Розглянемо приклад:

```
<% Language = VBScript%>
```

```
<HTML>
<BODY>
<%
Response.Write ( "<p> Hello world! </ P>")
%>
</ BODY>
</ HTML>
```

5.6 Server-side JavaScript

ЛЕКЦІЯ 6 СТВОРЕННЯ CGI-СЦЕНАРІЇВ НА МОВІ PERL

Анотація

Стандарт CGI. Сценарії. Сценарні мови: класифікація за швидкодією. Мова Python. Мова Ruby. Технологія ASP.

План лекції

1) Запуск сценарію
2) Скалярні змінні
3) Робота з файлами
4) Регулярні вирази

6.1 Запуск сценарію

Мова Perl (Practical Extraction and Report Language) - це мова програмування, сильними сторонами якого вважаються його багаті можливості для роботи з текстом, в тому числі реалізовані за допомогою регулярних виразів. Також мова відома тим, що має величезну колекцію додаткових модулів CPAN .

Щоб запустити програму на мові Perl на виконання, її компіляція не потрібно, вона цілком може виконуватися під керуванням інтерпретатора. щоб файл з вихідним текстом Perl можна було запускати на виконання, треба щоб перша його рядок виглядала так:

```
#! Путь_к_інтерпретатору_Perl
```

Основними типами даних в мові є: скаляри, масиви (скалярні), хеш-таблиці (асоціативні масиви), функції, файлові дескриптори і константи.

Змінні різних типів відрізняються знаком, який стоїть перед ім’ям змінної:

$a - скаляр або покажчик

@b - скалярний масив

%c - асоціативний масив (хеш-таблиця)

&d - функція

F - дескриптор введення-виведення або константа

6.2 Скалярні змінні

Скалярні змінні використовуються для зберігання одиночних значень. Вони можуть містити числа, рядки і посилання на інші об’єкти. Перед ім’ям скалярної змінної необхідно ставити знак долара ’$’. Тип скалярної змінної не фіксований і визначається динамічно в залежності від контексту.

скалярний масив є впорядкованим списком скалярів. Кожен елемент масиву має порядковий номер (індекс), за допомогою якого до нього можна отримати доступ. Нумерація елементів починається з нуля.

Перед ім’ям змінної типу скалярний масив вказується знак @, а для доступу до певного елементу масиву необхідно ставити знак $, так як певний елемент масиву є скаляром:

```
@winter = ( "грудень", "січень", "лютий");
print "Другий місяць зими", $winter [1], "\ n";
```

Хеш-таблиця являє собою асоціативний масив, що дозволяє асоціювати рядок (ключ) з скаляром (значення). Рядок при цьому називається ключем, а скаляр в хеш-таблиці - значенням. Перед ім’ям змінної-списку необхідно ставити знак відсотка%, а для доступу до певного елементу масиву ставлять знак $.

Фактично хеш-таблиця являє собою масив, де в непарних позиціях знаходяться ключі, а на парних - значення.

Використання асоціативних масивів нагадує використання масивів скалярних значень, проте індексація проводиться не цілими числами, а ключовими словами. Крім того, індекси полягають не в квадратні, а в фігурні дужки.

Так, наприклад, для того щоб привласнити значення трьох елементів масиву %dict з індексами first, second і third, можна скористатися одним з двох способів, зазначених нижче.

```
$dict { 'first'} = "перший";
$dict { 'second'} = "другий";
$dict { 'third'} = "третій";
```

або

```
 %dict { 'first', 'second', 'third'} =
"перший другий третій";
```

Крім того, існує спосіб одночасно записати в асоціативний масив і ключові слова, і їх значення. Зробити це можна за допомогою наступного виразу:

імя_массіва = (ключ 1, значення 1, ключ 2, значення 2, ...);

Для прикладу, наведеного вище, це вираз буде виглядати так:

```
%dict = ( "first", "перший",
"Second", "другий",
"Third", "третій");
```

Розглянемо наведений наступний фрагмент програми на мові Perl.

```
while (<STDIN>)
{Print;  }
```

Незважаючи на те, що формально програма складена правильно, вона на перший погляд може здатися безглуздою. Однак при запуску вона поводиться точно так само, як і одна з програм "відлуння". У мові Perl існує зумовлена ??скалярная змінна $_, яка використовується за замовчуванням. Саме в неї поміщаються дані, прочитані зі стандартного вводу, і з неї береться значення для виведення в STDOUT.

Крім $_ в Perl є й інші визначені змінні:

$] - номер версії Perl. $. - номер рядка, прочитаної з файлу останньої. $! - повідомлення про помилку. $$ - ідентифікатор поточного процесу. $Ť - час в секундах з початку 1970 року до запуску даної програми. $O - ім'я файлу, в

якому міститься виконувана програма. $1 ... $ 9 - фрагменти тексту, відмічені при виконанні операції зіставлення з шаблоном.

Подібно визначеним скалярним змінним, в Perl існують масиви, які мають спеціальну значення. Найбільш важливий з них - асоціативний масив %ENV, що містить поточні значення змінних оточення. Щоб отримати значення змінної оточення, треба звернутися до елементу даного масиву, вказавши в якості індексу ім’я змінної оточення. Так, наведене нижче вираз записує в скалярную змінну $path_string значення змінної оточення PATH.

$path_string = $ENV’PATH’;

Одна з перших рядків CGI-сценарія на Perl, може виглядати так

$method = $ENV’REQUEST_METHOD’;

6.3 Робота з файлами

Для роботи з файлами і потоками в Perl передбачені спеціальні файлові дескриптори.

Файлові дескриптори є покажчик на файл, пристрій або PIPE канал, відкриті для запису, читання або для запису і читання. Оператор «<>"в Perl називається діамантовим оператором (diamond operator). Він визначає операцію читання рядки з потоку, дескриптор якого міститься в кутових дужках:

$str = <STDIN>; # Читання рядки з дескриптора STDIN (стандартного потоку введення) @lines = <F>; # Читання всіх рядків з пов’язаного з дескриптором файлу F. print STDOUT $str; # Друк в STDOUT (стандартний потік виводу)

Для зв’язування файлу з файловим дескриптором використовується функція open. Нижче наводяться варіанти використання цієї функції:

open дескриптор_потока> имя_файла файл відкривається для виводу даних. Якщо файл з вказаним ім’ям відсутня, створюється новий файл.

open дескриптор_потока > > имя_файла файл відкривається в режимі, що дозволяє записувати дані в кінець файлу.

open дескриптор_потока +> имя_файла відкритий файл стає доступним для читання і для запису.

функція

Close дескриптор_файла

закриває файл, пов'язаний із зазначеним дескриптором.

6.4 Регулярні вирази

До складу мови Perl входять засоби пошуку та заміни, причому, задаючи шаблон для пошуку, можна використовувати регулярні вирази. Це означає, що складні операції, що зустрічаються в спеціалізованих додатках, можна легко реалізувати в будь-який Perl-програми.

Оператор пошуку m // записується в такий спосіб:

m / шаблон /

якщо значення змінної $_ містить підрядок, відповідну зазначеному шаблоном, оператор пошуку повертає значення true.

Розглянемо наступний приклад:

```
$_ = <INPUT>;
if (m / Scripts /)
{Print "У URL є каталог Scripts \ n";  }
else
{Print "У URL немає каталогу Scripts \ n";  }
```

Оператор заміни s /// записується в такий спосіб:

s / шаблон пошуку / вираз для заміни / [набір модифікаторів]

При виконанні оператора s /// проводиться пошук відповідності шаблоном, і якщо пошук завершується успішно, знайдена подстрока замінюється зазначеним виразом. Подібно оператору m //, оператор s /// використовує змінну $_. Нижче наведено найпростіший приклад застосування оператора s ///.

$_ = "CGI-сценарій написаний на мові C"; s/C$/Perl/; print;

В результаті виконання сценарію на консоль буде виведена наступна рядок:

CGI-сценарій написаний на мові Perl

Оскільки символ C міститься в абревіатурі CGI, тому в шаблоні пошуку вказано, що він повинен бути останнім у рядку.

За останніми роздільником в операторі s /// можуть слідувати один або кілька модифікаторів. Призначення деяких модифікаторів наведено нижче.

g - глобальний пошук. Якщо цей модифікатор не вказано, після виявлення першого відповідності оператор s /// закінчить свою роботу. Тому при відсутності модифікатора g буде вироблено не більше однієї заміни. i - вказує, що при пошуку слід ігнорувати регістр символів. e - вказує, що послідовність символів для заміни слід інтерпретувати не як підрядок, а як вираження Perl.

У вираженні для підстановки можуть бути присутніми змінні 1– 9, і в цьому випадку необхідно вказати модифікатор e. Так, наприклад, якщо потрібно інтерпретувати десяткове число як код символу, можна воспользоват’ся наступним виразом:

s/([0-9]+)/chr($1)/e;

щоб пошук або заміна проводилися в рядку, що міститься в потрібній змінної, треба використовувати наступне вираз:

Мінлива = õператор_поіска_ілі_замени

Так, наприклад, для перетворення шістнадцяткових чисел, що містяться в змінній $string, в десяткове уявлення можна використовувати інструкцію:

$string = š/([0-9A-Fa-f]+)(H h)/hex($1)/ge;

ЛЕКЦІЯ 7 ВСТУП У ПРОГРАМУВАННЯ НА МОВІ PHP

Анотація

Прочитавши цю лекцію ви дізнаєтеся, як PHP вбудовується в XHTML, а також навчитеся писати коментарі та виводити текст на екран браузера

План лекції

1) З'єднання HTML і PHP
2) Оператори виводу даних
3) Термінатор інструкції
4) Коментарі

7.1 З'єднання HTML і PHP

Код PHP зазвичай об'єднується з тегами HTML. PHP є вбудовуваною мовою - це означає, що можна переміщатися між чистим кодом HTML і PHP, не жертвуючи можливістю читання тексту.

Щоб вбудувати код PHP в HTML, PHP повинен задаватися відокремлено, за допомогою початкового і кінцевого тегів PHP. Теги PHP кажуть сервера Web, де починається і закінчується код PHP. аналізатор PHP розпізнає три варіанти початкового і кінцевого тегів.

стиль XML

```
<?php
Блок коду PHP
?>
```

Перший варіант тегів PHP називається тегами в стилі XML і є кращим стилем. Він працює в документах Розширюваної мови розмітки (XML). Цей метод повинен використовуватися при з'єднанні PHP з документами XML і XHTML. Приклади в цьому підручнику застосовують цей формат тегів XML.

скорочений стиль

```
<?
Блок коду PHP
?>
```

Скорочений стиль є найпростішим, проте, він не рекомендується, так як вступає в протиріччя з оголошеннями документів XML.

Стиль сценарію (script)

```
<script language = "php">
Блок коду PHP
</script>
```

Цей стиль використовує найдовшу запис і схожий на стиль тегів, які застосовуються з JavaScript. Цей стиль є кращим при використанні редактора HTML, який не розпізнає інші стилі тегів. Так як більшість нових редакторів XHTML розпізнають стиль тегів XML, то використання цього стилю не рекомендується.

Блоки сценарію можуть розміщуватися в будь-якому місці документа HTML, в тому місці, де сценарій створює і показує свій висновок. Наступний приклад сторінки XHTML ілюструє використання трьох форматів тегів сценарію.

```
<!DOCTYPE html>
<html lang = "en">
<head>
  <title> Сторінка Web </ title>
</ head>
<body>

<P>
```

```
<? Php echo "Це базовий документ PHP";?>
</P>

<P>
<?  print "PHP - це здорово!";?>
</P>
<P>
<script language = "php">
$myvar = "Hello World!";
echo $myvar;
</script>
</P>
</body>
</html>
```

У попередньому прикладі три блоки PHP включені в документ XHTML. Перший блок використовує відкриває і закриває теги <?php ...?>. Сегмент коду використовує оператор PHP echo для виведення рядка "Це базовий документ PHP "у вікні браузера.

Другий блок застосовує теги <?...?> Для позначки початку і кінця коду PHP. Цей розділ застосовує оператор PHP print (інше ім’я оператора echo) для виведення на екрані тексту "PHP - це здорово! ".

Нарешті, третій блок використовує блок сценарію <script language = "php» ... </ script> для визначення початку і кінця коду PHP. У коді рядок "Hello World!"присвоюється змінної $myvar, а оператор echo виводить значення $myvar у вікні браузера.

Це базова сторінка PHP.

PHP - це здорово!

Hello World!

Приклад показаного вище коду включає теги XHTML, теги PHP, оператори PHP і роздільники. коли користувач запитує сторінку PHP, сервер обробляє весь код PHP. коли сторінка PHP проглядається в вікні браузера, виводиться тільки текст між відкриває і закриває тегами XHTML або PHP. Ніякої реальний код PHP не видно при перегляді вихідного коду в вікні браузера. Причина у тому, що інтерпретатор PHP виконує сценарій на сервері і замінює код результатом виведення роботи сценарію. Тільки цей висновок передається браузеру. Це одна з характеристик, яка робить PHP серверним мовою сценаріїв, на відміну від JavaScript, мови сценаріїв клієнта. висновок контенту

7.2 Оператори виводу даних

PHP містить два основних оператора для виведення тексту в браузері Web: echo і print.

Обидва оператора, echo і print, кодуються між відкриває і закриває тегами блоку коду PHP і можуть перебувати в будь-якому місці в документах XHTML.

Оператори echo і print використовують наступний формат:

echo - використовується для виведення однієї або декількох рядків.

echo "виводиться текст";

print - використовується для виведення рядка. У деяких випадках оператор print пропонує більшу функціональність, ніж оператор echo. Це буде розглянуто далі в підручнику. Поки print можна вважати іншим ім’ям оператора echo.

print "виводиться текст";

Наступні приклади демонструють використання і розміщення команд echo і print в документі XHTML.

У більшості випадків необхідно виводити цілі параграфи у вікні браузера або створювати переноси рядків при виведенні контенту. за замовчуванням оператори echo і print не створюють автоматичні переноси рядків, необхідно використовувати тег <p> або
 для створення параграфів або переносів рядків. Роздільники, створювані в редакторі XHTML за допомогою повернен-

ня каретки, прогалин і табуляції, ігноруються процесором PHP.

У наступному прикладі тег параграфа XHTML включається в оператор PHP echo. В PHP теги XHTML можна застосовувати в операторах print і echo для форматування виводу. У цих випадках висновок необхідно укладати в подвійні лапки ( ), щоб гарантувати, що браузер не інтерпретує тег буквально і не виведе його як частину рядка виводу.

echo «p> Параграф 1 </p>"; echo «p> Параграф 2 </p>";

Без використання тега параграфа XHTML попередні оператори echo виводитимуть контент в наступному вигляді:

Параграф 1 Параграф 2

При включенні тегів параграфів оператори виводяться як два окремих параграфа.

параграф 1

параграф 2

7.3 Термінатор інструкції

Кожен рядок коду PHP повинна завершуватися термінатором інструкції (ознакою кінця), в якості якого використовується крапка з комою (;). Термінатор інструкції застосовується для відділення одного безлічі інструкцій від іншого.

Відсутність термінатора інструкції може призводити до помилок в роботі інтерпретатора PHP і висновку помилок у вікні браузера. Наступний фрагмент коду показує поширену помилку відсутності термінатора інструкції.

```
<!DOCTYPE html>
<html lang = "en">
<head>
  <title> Сторінка Web </ title>
</ head>
<body>
```

```
<P>
<?
echo "Цей рядок породжує помилку"
echo "У попередньому рядку відсутня термінатор інструкції";
?>
</P>

</body>
</html>
```

У цьому прикладі в першому операторі echo пропущений термінатор інструкції. Так як інтерпретатор PHP не може визначити кінець першого оператора echo і початок другого, то виникає проблема. Залежно від налаштувань Обробки помилок (Error Handling) інтерпретатора, буде виводитися повідомлення про помилку або браузер виведе просто порожню сторінку.

Далі показано типове повідомлення про помилку, яке виводиться, коли пропущений термінатор інструкції:

```
Parse error: syntax error,
unexpected T_PRINT in C:\\ApacheRoot\\test.php on line 11
```

Помилка при розборі: синтаксична помилка, непередбачений T_PRINT в ... в рядку 11

Як можна бачити, повідомлення про помилки PHP досить загадкові за своєю природою. Відсутність термінатора інструкції є поширеною помилкою серед новачків PHP. Згодом з’явиться звичка перевіряти кожен рядок на наявність термінатора інструкції. Коментарі в коді

7.4 Коментарі

Коментарі застосовуються в PHP для запису власних зауважень під час процесу розробки коду. Такі коментарі можуть визначати призначення сегмента коду або їх можна використовувати для виключення блоків коду у час тестування і налагодження сценаріїв.

синтаксичний аналізатор PHP ігнорує коментарі. Коментарі в PHP можна визначити одним з наступних способів:

// - простий коментар PHP;

- альтернативний простий коментар PHP;

/*...*/ - багаторядкові блоки коментарів.

ЛЕКЦІЯ 8 ВСТУП У ПРОГРАМУВАННЯ НА МОВІ PHP

Анотація

Змінні - основа будь-якої мови програмування. Ця лекція про них, крім того, описані деякі вбудовані функції PHP, а також розказано про відмінність одиночних і подвійних лапок в PHP

План лекції

1) Скалярні змінні
2) Оператори виводу даних
3) Термінатор інструкції
4) Коментарі

8.1 Скалярні змінні

Змінні є тимчасовим місцем зберігання, використовуваним для представлення значень у сценарії PHP. В PHP є два основних типи змінних: скалярні і масиви. Скалярні змінні містять тільки одне значення в даний момент часу, а змінні масиви - список значень. Змінні масиви обговорюються в наступному розділі. скалярні змінні PHP містять значення наступних типів.

Цілі - цілі числа або числа без десяткового дробу (1, 999, 325 812 841).

Числа з плаваючою точкою - числа, що містять десяткову точку (1.11, 2.5, .44).

Рядки - текстова або числова інформація. Строкові дані завжди визначаються за допомогою лапок ("Hello World "478-477-5555").

Булеві значення - використовуються для значень true (істина) або false (брехня).

імена змінних PHP всіх типів починаються зі знака "$". Імена змінних можуть містити літери, цифри, і символ підкреслення (_); вони не можуть,

однак, починатися з цифри. В PHP імена змінних розрізняють регістр символів. Наступні змінні в PHP інтерпретуються як дві різні змінні.

$myvar

$MYVAR

Допустимі імена змінних:

$myvar

$F_Name

$address1

$my_string_variable

Неприпустимі імена змінних:

Myvar

$1stvar

$&62##

Скалярним змінним PHP привласнюють значення в наступному форматі:

$username = "jdoe"

$first_name = "John"

$last_name = "Doe"

```
<!DOCTYPE html>
<html lang = "en">
<head>
  <title> Сторінка Web </title>
</head>
<body>
<script language = "php">
$string_var = "Моя програма PHP";
$integer_var = 500;
$float_var = 2.25;
```

```
echo $string_var;
echo $integer_var;
echo $float_var;
</script>
</P>
</body>
</html>
```

Змінні масиви PHP можна створювати і привласнювати їм значення за допомогою конструкції array () або явно.

Змінну можна з'єднувати з іншими змінними або тегами HTML за допомогою оператора PHP - точки (.). У попередньому блоці коду значення змінних виводяться в наступному форматі:

Моя програма PHP5002.25

Щоб створити повернення каретки або перенесення рядка, можна приєднати тег XHTML
 в кінці кожної пременной:

```
<? Php
$string_var = "My PHP program".  "<br/>";
$integer_var = 500.  "<br/>";
$float_var = 2.25;

echo $string_var;
echo $integer_var;
echo $float_var;

?>
```

Тепер після кожної змінної вставляється перенесення рядка, що призводить до висновку кожного значення на окремому рядку.

My PHP Program

500

2.25

8.2 Інтерполяція

PHP підтримує також процес, званий інтерполяцією - заміну змінної в рядку її вмістом. Замість з’єднання змінних і літералів, їх можна об’єднувати подвійних лапках ( ). Інтерполяція є властивістю тільки подвійних лапок. Змінні і літерали можна об’єднати всередині одиночних лапок. При використанні подвійних лапок значення змінної виводиться разом з літералом. При використанні одиночних лапок виводиться "буквально"ім’я змінної разом з рештою рядком. Наступний приклад ілюструє властивість інтерполяції PHP.

```
<!DOCTYPE html>
<html lang = "en">
<head>
  <title> Сторінка Web </ title>
</ head>
<body>
<?php
$fname = "John";
$lname = "Doe";

echo "The user \ ’s name is $fname $lname";
?>
</body>
</html>
```

Цей код створює такий же висновок, як і попередній приклад. Тут змінні об’єднуються за допомогою літеральної рядки, укладеної в подвійні лапки.

З’єднання (конкатенація) не потрібно.

8.3 Форматоване виведення

Крім виведення стандартного тексту можна застосовувати для виведення тексту фіксованої варіант конструкції print з ім’ям sprintf. Оператор вимагає завдання форматує рядки і значення для форматування.

sprintf("%01.2f $var) - виводить значення ’$var’ як число з двома знаками після крапки.

Оператор sprintf показаний нижче:

```
<!DOCTYPE html>
<html lang = "en">
<head>
  <title> Сторінка Web </ title>
</ head>
<body>
<?php
$amount = 35;
$tax = 2.50;
$total = $ amount + $ tax;
echo "$".  sprintf ( "% 01.2f", $total);
?>
</body>
</html>
```

Виведення валюти показаний нижче:

$37.50

ЛЕКЦІЯ 9 L 1

9.1 l1 s1

fsv sdvs

ЛЕКЦІЯ 10 L 1

10.1 l1 s1

ЛЕКЦІЯ 11 L 1

11.1 l1 s1

ЛЕКЦІЯ 12 L 1

12.1 l1 s1

ЛЕКЦІЯ 13 L 1

13.1 l1 s1

ЛЕКЦІЯ 14 L 1

14.1 l1 s1

ЛЕКЦІЯ 15 L 1

15.1 l1 s1

ЛЕКЦІЯ 16 L 1

16.1 l1 s1

ЛЕКЦІЯ 17 L 1

17.1 l1 s1

ЛЕКЦІЯ 18 L 1

18.1 l1 s1

ЛЕКЦІЯ 19 L 1

19.1 l1 s1

ЛЕКЦІЯ 20 L 1

20.1 l1 s1

ЛЕКЦІЯ 21 L 1

21.1 l1 s1

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ