Лабораторна робота №4

ВИКОРИСТАННЯ РЕГУЛЯРНИХ ВИРАЗІВ. ЗАСТОСУВАННЯ HTTP COOKIES І МЕХАНІЗМУ СЕСІЙ

Мета роботи:вивчення можливостей обробки текстових даних із застосуванням регулярних виразів, а також способів передачі даних між скриптами за допомогою HTTP cookies та механізму сесій.

Досліджуваний матеріал: функції та синтаксис регулярних виразів, HTTP cookies та механізм сесій користувача (ел. документ – посібник з мови).

1. Постановка задачі

Розробити і реалізувати на мові PHP серверний сценарій (скрипт) для обробки запитів користувача та подання результатів у вигляді генерованого документа HTML.

2. Порядок виконання роботи

1. Отримати у викладача індивідуальне завдання для виконання роботи.
2. Вивчити теоретичний матеріал.
3. Визначити завдання, які має вирішувати сценарій, що розробляється.
4. Розробити текстовий інтерфейс користувача відповідно до завдання, а також кінцевий вид документа, що генерується, з результатами роботи сценарію.
5. Реалізувати мовою PHP спроектований сценарій.
6. Протестувати локально розроблений сценарій.
7. Зробити висновки щодо роботи.

3. Зміст звіту

1. Постановка задачі. Опис завдань, які вирішуються серверним сценарієм.
2. Короткий опис алгоритму сценарію.
3. Опис використовуваних вхідних даних та функцій.
4. Лістинг вихідного коду сценарію з коментарями, а також результати його роботи (скриншот або текстове подання).
5. Висновки щодо роботи.

4. Основні теоретичні відомості

Основні відомості містяться у конспекті лекцій.

4.1. Синтаксис шаблонів (патернів) PCRE наведено нижче.

Регулярний вираз це патерн, який порівнюється з рядком-суб'єктом зліва направо. Більшість символів у патерні представляють себе і збігаються з відповідними символами в рядку-суб'єкті. Як тривіальний приклад - патерн The quick brown fox збігається з частиною рядка-суб'єкта, який повністю ідентичний йому.

Метасимволи

Потужність регулярним виразам надає можливість включати до патерну альтернативи та повторення. Вони кодуються в патерне метасимволами, які не уявляють себе, а інтерпретуються особливим чином.

Є два різних набори метасимволів: патерна, що розпізнаються в будь-якому місці, крім квадратних дужок, і ті, які розпізнаються в квадратних дужках.

****

загальний escape-символ для різних варіантів використання

^

початок затвердження суб'єкта (або рядка, у багаторядковому режимі)

$

кінець затвердження суб'єкта (або рядки, у багаторядковому режимі)

.

збігається з будь-яким символом, окрім newline (за замовчуванням)

[

початок визначення класу символів

]

кінець визначення класу символів

|

початок альтернативної гілки

(

початок субпатерну

)

кінець субпатерну

?

розширює значення (также квантифікатор 0 або 1, також мінімізатор квантифікатора

*****

квантифікатор 0 або більше

+

квантифікатор1 або більше

{

початок min/max квантифікатора

}

кінець min/max квантифікатора

Частина патерна у квадратних дужках називається "класом символів". У класі символів метасимволами є лише:

****

загальний escape-символ

^

заперечує цей клас, але якщо є першим символом

-

позначає діапазон символів

]

кінець класу символів

У наступних розділах розглядається використання кожного з цих метасимволів.

backslash/зворотний слеш

Символ backslash використовується по-різному. По-перше, якщо після нього йде неалфавітний символ, він скасовує будь-яке спеціальне значення, яке може мати символ. Таке використання зворотного слеша як escape-символу застосовується як усередині, і поза класів символів.

Наприклад, якщо ви хочете знайти збіг із символом "*", ви записуєте в патерні "*". Це буде працювати незалежно від того, чи може наступний символ інтерпретуватися як метасимвол, тому завжди надійніше записувати неалфавітний символ з "\", щоб специфікувати, що він уявляє себе. Особливо якщо ви хочете знайти збіг з backslash - тоді ви записуєте "\\".

Якщо патерн компілюється із опцієюPCRE_EXTENDED, то пробіли в патерні (крім пробілів у класі символів) і символи між "#" поза класом символів і наступним символом newline ігноруються.

По-друге, backslash надає спосіб кодування в патерні недрукованих символів видимим чином. Обмежень на появу недрукованих символів немає, за винятком двійкового нуля, який закінчує патерн, але якщо патерн готується шляхом редагування тексту, то зазвичай легше використовувати одну з наступних escape-послідовностей (замін), а не бінарний символ.

\a

alarm/попередження, тобто символ BEL (hex 07)

\cx

"control-x", де x це будь-який символ

\e

escape (hex 1B)

\f

formfeed/прогін сторінки (hex 0C)

\n

newline/новий рядок (hex 0A)

\r

carriage return/повернення каретки (hex 0D)

\t

tab/табуляція (hex 09)

\xhh

символ з 16-річним кодом hh

\ddd

символ з 8-річним кодом ddd, або backreference/зворотне посилання

Ефект від застосування \cx такий: якщо "x" це символ у нижньому регістрі, він конвертується у верхній регістр. Потім біт символу 6 (hex 40) інвертується. Таким чином, \cz стає hex 1A, \c{ стає hex 3B, а \c; ставати hex 7B.

Після "x" читаються не більше двох 16-річних цифр (букви можуть бути в будь-якому регістрі).

Після "\0" читаються не більше чотирьох 8-річних цифр. В обох випадках, якщо є менше двох цифр, використовуються ті, які представлені. Таким чином, послідовність "\0\x\07" специфікує два бінарних нулі з наступним символом BEL. Переконайтеся, що ви надали дві цифри після початкового нуля, якщо наступний символ є 8-річним числом.

Обробка backslash з наступними цифрами, відмінними від 0, складніша. Поза класом символів, PCRE читає його та будь-які наступні символи як 10-річне число. Якщо число менше 10 або якщо у виразі є щонайменше стільки захоплюючих лівих дужок, вся послідовність вважається back reference\зворотним посиланням. Опис того, як цей механізм працює, буде дано пізніше у дискусії про субпатерни у дужках.

Усередині класу символів, або якщо 10-річне число більше 9 і немає такої кількості захоплюючих субпатернів, PCRE зчитує до трьох 8-річних цифр, що йдуть після backslash, і генерує один байт з останніх значних 8 бітів цього значення. Будь-які наступні цифри представляють себе. Наприклад:

\040

інший спосіб запису space

\40

те саме, за умови, що є менше 40 попередніх захоплюючих субпатернів

\7

це завжди back reference/зворотне посилання

\11

може бути back reference або інший спосіб запису tab

\011

це завжди tab

\0113

це tab з наступним символом "3"

\113

це символ з 8-річним кодом 113 (оскільки не може бути більше 99 зворотних посилань)

\377

це байт, що складається з бітових 1

\81

це back reference або бінарний нуль з наступними двома символами "8" та "1"

Зауважте, що 8-річні значення 100 або більше повинні не вводитися з провідним 0, оскільки читається не більше трьох 8-річних цифр.

Усі послідовності, що визначають однобайтне значення, можуть використовуватися як усередині, так і поза класами символів. Крім того, всередині класу символів послідовність "b" інтерпретується як символ backspace (hex 08). Поза класом символів вона має інше значення (див. далі).

Третій варіант використання backslash - специфікація загального типу символів:

\d

будь-яке 10-річне число

\D

будь-який символ - не 10-річне число

\s

будь-який пробіловий символ

\S

будь-який непробільний символ

\w

будь-який "word/словниковий" символ

\W

будь-який "non-word/несловниковий" символ

Кожна пара escape-послідовностей поділяє повний набір символів на два різні набори. Будь-який цей символ збігається з однією, і лише з однією парою.

Символ "word" це будь-яка літера, або цифра, або символ підкреслення, тобто будь-який символ, який може бути частиною "word" у Perl. Визначення літер та цифр контролюється таблицями символів PCRE і може змінюватись, якщо має місце підстановка з локальною специфікою (див. раніше "Підтримка локалізації"). Наприклад, при локалізації "fr" (French) використовуються деякі символи з кодами вище 128 для введення букв з акцентами і вони збігаються з \w.

Ці послідовності типів символів можуть з'являтися як усередині, так і за межами класів символів. Кожен із них збігається з одним символом відповідного типу. Якщо поточна точка збігу є кінцем рядка-суб'єкта, всі вони зазнають невдачі, оскільки немає символу для порівняння.

Четвертий варіант - використання backslash для деяких простих тверджень. Затвердження специфікує умову, яка має бути знайдена в певній точці під час встановлення, не використовуючи жодних символів з рядка-суб'єкта. Використання субпатернів для складніших тверджень розглядається далі. Твердження зі зворотними слешами це:

\b

межа слова

\B

не межа слова

\A

початок суб'єкта (не залежить від багаторядкового режиму)

\Z

кінець суб'єкта або newline в кінці (не залежить від багаторядкового режиму)

\z

кінець суб'єкта (не залежить від багаторядкового режиму)

Ці твердження не можуть з'являтися в класах символів (але зауважте, що "b" має інше значення, а саме символ backspace, усередині класу символів).

Кордон слова це така позиція в рядку-суб'єкті, де поточний і попередній символи не збігаються з \w або \W (тобто один збігається з \w, а інший - з \W), або початок або кінець рядка, якщо перший або останній символ збігається з \w відповідно.

Твердження \A, \Z і \z відрізняються від традиційних circumflex і dollar (описано далі) тим, що вони збігаються тільки з початком і кінцем рядка-суб'єкта, незалежно від встановлених опцій. На них не впливають опціїPCRE_NOTBOLабоPCRE_NOTEOL. Різниця між \Z та \z у тому, що \Z збігається до newline, тобто є останнім символом рядка, а також кінцем рядка, тоді як \z збігається лише наприкінці рядка.

Circumflex та dollar

Поза класом символів, у режимі підстановки за замовчуванням, символ circumflex (^) є твердженням, яке є true, тільки якщо поточна точка збігу є початком рядка-суб'єкта. Усередині класу символів circumflex має зовсім інше значення (див. далі).

Circumflex не повинен бути першим символом патерна, якщо використовуються кілька альтернатив, але повинен бути першим у кожній альтернативі, в якій з'являється, якщо патерн збігається з цією гілкою. Якщо всі наявні альтернативи починаються з circumflex (^), тобто якщо патерн обмежений для збігу тільки на початку суб'єкта, говориться, що це "заякорений/anchored" патерн. (Є також інші конструкції, які можуть викликати заякорювання патерна.)

Символ dollar є твердженням, яке TRUE, тільки якщо поточна точка збігу знаходиться в кінці рядка-суб'єкта або відразу після символу newline, який є останнім символом рядка (за замовчуванням). Dollar не повинен бути останнім символом патерна, якщо дано кілька альтернатив, але має бути останнім символом у будь-якій гілки, в якій він з'являється. Dollar не має спеціального значення у класі символів.

Значення dollar може бути змінено так, щоб він збігався тільки з кінцем рядка, через установку опціїPCRE_DOLLAR_ENDONLYпід час компіляції чи підстановки. Це не впливає на затвердження \Z.

Значення символів circumflex та dollar змінюється, якщо встановлена опціяPCRE_MULTILINE. В цьому випадку вони збігаються відразу після або відразу до внутрішнього символу \n, відповідно, на додаток до збігу на початку і в кінці рядка-суб'єкта. Наприклад, патерн /^abc$/ збігається з рядком-суб'єктом "def\nabc" у багаторядковому режимі, але не інакше. Відповідно, патерни, які заякорені в однорядковому режимі, якщо всі гілки починаються з "^", не є заякореними в багаторядковому режимі. ОпціяPCRE_DOLLAR_ENDONLYігнорується, якщо встановленаPCRE_MULTILINE.

Зауважте, що послідовності \A, \Z та \z можуть використовуватися для збігу з початком і кінцем суб'єкта в обох режимах, і якщо всі верви на початку патерну з \A завжди заякорені, незалежно від того, встановленаPCRE_MULTILINEчи ні.

FULL STOP/ПОВНИЙ ЗУПИНОК

Поза класом символів точка в патерні збігається з одним із символів суб'єкта, включаючи символ, що не друкується, але не з символом (за замовчуванням) newline. Якщо встановлено опціюPCRE_DOTALL, точки збігаються також із символами newline. Обробка точки залежить від обробки circumflex і dollar, їхріднить тільки те, що вони обидва вводять символи newline. Крапка не має спеціального значення у класі символів.

Квадратні дужки

Відкриваюча квадратна дужка вводить клас символів, що закінчується квадратною дужкою, що закриває. Квадратна дужка, що закриває, сама по собі не є спеціальним символом. Якщо квадратна дужка, що закриває, необхідна як член класу символів, вона повинна бути першим символом даних класу (після початкового circumflex, якщо він є) або повинна замінюватися за допомогою backslash.

Клас символів збігається з одиночним символом рядка-суб'єкта; Символ повинен бути з набору символів, визначеного в даному класі, якщо тільки першим символом у класі не є circumflex - у цьому випадку символ рядка-суб'єкта повинен не входити в набір, визначений у даному класі. Якщо circumflex необхідний як член класу, переконайтеся, що він не є першим символом, або escape-ируйте його за допомогою зворотного слеша (\).

Наприклад, клас символів [aeiou] збігається з будь-якою голосною буквою в нижньому регістрі, а [^aeiou] збігається з будь-яким символом, що не є голосною буквою в нижньому регістрі.Зверніть увагу, що circumflex це просто зручний символ для позначення символів, які не використовуються для збіги. Він не є твердженням: він просто використовує символи з рядка-суб'єкта і зазнає невдачі, якщо покажчик знаходиться наприкінці рядка.

Якщо встановлено збіг без урахування регістру, то будь-яка буква класу представляє символи як нижнього, так і верхнього регістрів, тому, наприклад, безреєстровий [aeiou] збігається з "A" і з "a", безреєстровий [^aeiou] збігається з не- "A" і з не-"a", в той час як реєстрова версія збігається.

Символ newline ніколи не розглядається спеціально у класах символів, незалежно від того, встановлені опції PCRE_DOTALL або PCRE_MULTILINE чи ні. Клас, такий як [^a], завжди збігатиметься з newline.

Знак "мінус" (дефіс) можна використовувати для специфікації діапазону символів у класі символів. Наприклад, [dm] збігається з будь-якою літерою від d до m включно. Якщо знак мінус необхідний як член класу, він повинен вводитися заміною за допомогою backslash або з'являтися в позиції, де він не може інтерпретуватися як діапазон, що позначає, тобто. зазвичай, як перший або останній символ класу.

Неможливо мати літеральний символ "]" як кінець діапазону. Патерн [W-46] інтерпретується як клас із двох символів ("W" і "-") з наступним літеральним рядком "46]", тому він збігається з "W46]" або з "-46]". Однак, якщо "]" escape-ований зі зворотним слешем, він інтерпретується як кінець діапазону, тому [W-\]46] інтерпретується як єдиний клас, що містить діапазон, з наступними двома окремими символами. 8-річна або 16-річна вистава "]" також може використовуватися для вказівки на кінець діапазону.

Діапазони оперують у ASCII-кодуванні символів. Вони також можуть використовуватися для символів, специфікованих числами, наприклад [000-037]. Якщо діапазон літер встановлено при вимкненому розпізнаванні регістру символів, він збігається з літерами у будь-якому регістрі. Наприклад, [Wc] еквівалентно [][\^_`wxyzabc], збігаючись без урахування регістру і при використанні таблиць символів для локалізації "fr", [xc8-xxb] збігається з акцентованими символами E в будь-якому регістрі.

Типи символів \d, \D, \s, \S, \w та \W також можуть з'являтися в класах символів і додавати до класу символи, з якими вони збігаються. Наприклад, [\dABCDEF] збігається з будь-яким 16-річним числом. Символ circumflex (^) можна використовувати з типами символів верхнього регістру, щоб специфікувати більш обмежений набір символів - лише типи нижнього регістру. Наприклад, клас [^\W_] збігається з будь-якою літерою чи цифрою, але не символом підкреслення.

Всі неалфавітні символи, крім \, -, ^ (на початку) та заключного], є неспеціальними в класах символів, але не зашкодить вводити їх за допомогою escape-послідовностей.

Vertical bar/Вертикальна характеристика

Символи вертикальної риси (|) використовуються для поділу альтернативних патернів. Наприклад, патерн

gilbert | sullivan

збігається з "gilbert" або "sullivan". Можна вводити будь-яку кількість альтернатив, а також порожні альтернативи (збігаються з порожнім рядком). Процес підстановки відчуває альтернативи по черзі, зліва направо, і використовується перший знайдений збіг. Якщо альтернативи знаходяться у субпатерні (визначено далі), "використання" означає збіг залишку головного патерну, а також альтернативи у субпатерні.

Встановлення внутрішніх опцій

Налаштування опційPCRE_CASELESS,PCRE_MULTILINE,PCRE_DOTALLiPCRE_EXTENDEDможуть бути змінені з патерна за допомогою послідовності літер-опцій Perl, укладених між символами "(?" та ")". Ці літери-опції такі:

i дляPCRE_CASELESS

m дляPCRE_MULTILINE

s дляPCRE_DOTALL

x дляPCRE_EXTENDED

Наприклад, (?im) встановлює багаторядковий пошук збігу без урахування регістру. Можна також скасовувати установку опції, передуючи букві дефісом, і комбінувати установку та скасування опцій, як тут (?im-sx), де встановлюютьсяPCRE_CASELESSiPCRE_MULTILINEта скасовуютьсяPCRE_DOTALLiPCRE_EXTENDED. Якщо літера з'являється як до, так і після дефісу, опція скасовується.

Область видимості цих змін опцій залежить від цього, де у патерне зміни з'являються. Для установок поза субпатерном (визначено далі), ефект буде таким же, як і при установці та скасуванні опцій на початку збігу. Наступні патерни поводяться абсолютно однаково:

(?i)abca(?i)bcab(?i)cabc(?i)

що, у свою чергу, те саме, що компіляція патерну abc з установкоюPCRE_CASELESS. Інакше висловлюючись, такі установки " верхнього рівня " застосовуються до всього патерну (якщо відсутні інші зміни у субпатернах). Якщо є кілька параметрів однієї опції на верхньому рівні, використовується найправіша установка.

Якщо зміна опції виникає всередині субпатерну, ефект може бути різним. Ця зміна поведінки була зроблена в Perl 5.005. Зміна опції в субпатерні впливає тільки на ту частину субпатерну, яка слідує за ним; так

(a(?i)b)c

збігається з abc і з aBC і більше ні з чим (припускаючи, щоPCRE_CASELESSне використовується). Це означає, що опції можуть бути змінені для отримання різних установок у різних частинах патерну. Будь-які зміни, зроблені в одній альтернативі, впливають на наступні гілки всередині того самого субпатерну. Наприклад,

(a(?i)b|c)

збігається з "ab", "aB", "c" і "C", хоча при збігу з "C" перша гілка залишається до установки опції. Це відбувається тому, що установки опцій відбуваються на етапі компіляції. Інакше поведінка буде непередбачуваною.

PCRE-специфічні опціїPCRE_UNGREEDYіPCRE_EXTRAможуть бути змінені так само, як і Perl-сумісні опції шляхом використання символів U і X відповідно. Установка прапора (?X) є спеціальної тому плані, що він повинен з'являтися в патерне раніше, ніж буде включена будь-яка додаткова можливість, навіть якщо вона на верхньому рівні. Найкраще поміщати його при старті.

Субпатерни

Субпатерни обмежені дужками (круглими), які можуть вкладатись. Маркування частини патерну як субпатерну виконує дві дії:

1. Локалізує набір альтернатив. Наприклад, патернcat(aract|erpillar|)збігається з одним із слів: "cat", "cataract" або "caterpillar". Без дужок він збігається з "cataract", "erpillar" або з порожнім рядком.

2. Встановлює субпатерн як захоплюючий субпатерн (як визначено вище). Коли весь патерн збігається повністю, частина рядка-суб'єкта, що збіглася з субпатерном, передається назад, що викликає за допомогою аргументу ovector функції pcre_exec(). Відкриваючі дужки обчислюються ліворуч (починаючи з 1) для отримання кількості захоплюючих субпатернів.

Наприклад, якщо рядок "the red king" зіставляється з патерном ((red | white) (king | queen)) будуть захоплені підрядки "red king", "red" і "king", і вони будуть пронумеровані 1, 2 і 3.

Фактично таке виконання звичайними дужками двох функцій який завжди допомагає. Трапляється, коли необхідний угруповання субпатернів без необхідності захоплення. Якщо після відкритої дужки йде "?:", субпатерн не виконує захоплення і не враховується при підрахунку кількості субпатернів, що захопили. Наприклад, якщо рядок "the white queen" зіставляється з патерном((?:red|white) (king|queen)), то будуть захоплені підрядки "white queen" і "queen", і вони будуть пронумеровані 1 і 2. Максимальна кількість захоплюваних підрядок - 99, а максимальна кількість всіх субпатернів, захоплюючих та не захоплюючих, дорівнює 200.

Як зручна абревіатура, якщо будь-які установки опцій потрібні на початку незахоплюючого субпатерну, літери опцій можуть з'являтися між "?" та ":". Таким чином, два субпатерна (? i: saturday | sunday) (?: (? i) saturday | sunday) збігаються з одним і тим же набором рядків. Оскільки альтернативні гілки пробуються зліва направо, а опції не відновлюють значення, поки не буде досягнуто кінця субпатерну, установка опцій в одній гілки не впливає на наступні гілки, і тому наведені вище патерни збігаються з "SUNDAY", а також з "Saturday".

Повторення

Повторення специфікується квантифікаторами, які можуть йти після будь-якого з наступних елементів:

одиночного символу, можливо, escape-ованогометасимвола .класу символівзворотного посилання/back reference (див. наступний розділ)субпатерна в дужках (якщо це не затвердження/assertion, див. далі)

Квантифікатор загального повторення специфікує мінімальну та максимальну кількість допустимих збігів, маючи два числа у фігурних дужках, розділені комою. Число має бути менше 65536, а перше має бути менше або дорівнює другому. Наприклад: z {2,4} збігається з "zz", "zzz" або "zzzz". Фігурна дужка, що закриває, сама по собі не є спеціальним символом. Якщо друге число відсутнє, але кома є, верхньої межі немає; якщо відсутні друге число і кома, квантифікатор специфікує точну кількість необхідних збігів. Таким чином, [aeiou] {3,} збігається з як мінімум трьома 3 послідовними голосними, але може і з великою кількістю, а d {8} збігається точно з 8 цифрами. Фігурна дужка, що відкриває, яка з'являється в позиції, де квантифікатор неприпустимий, або дужка, яка не відповідає синтаксису квантифікатора, вважається літеральним символом. Наприклад, {,6} це не квантифікатор, а літеральний рядок із 4 символів.

Квантифікатор {0} припустимо, змушуючи вираз поводитися так, ніби попередній елемент та квантифікатор не існують.

Для зручності (і зворотної сумісності) три найбільш поширені квантифікатори мають односимвольні скорочення: * еквівалентний {0,}+ еквівалентний {1,}? еквівалентний {0,1} Можна конструювати нескінченні цикли, ввівши після субпатерну, який не збігається з жодним символом, квантифікатор, що не має верхньої межі, наприклад: (a?) * Ранні версії Perl і PCRE є джерелами помилок у процесі компіляції таких патернів . Однак, оскільки трапляються випадки, коли це необхідно, такі патерни приймаються, але якщо будь-яке повторення такого субпатерну фактично не збігається з жодними символами, цикл форсовано переривається.

За замовчуванням квантифікатори є "жадібними", тобто вони збігаються максимально можливу кількість разів (до максимально допустимої кількості разів), не викликаючи невдачі виконання решти патерну. Класичний приклад, коли це створює проблеми – спроба знайти збіги у коментарі С-програм. Коментарі з'являються між символами /* та */, а всередині можуть з'являтися окремі символи * та /. Спроба знайти збіг з С-коментарями, застосувавши патерн/*.**/до рядка/* перший коментар */ не коментар /* другий коментар */

зазнає невдачі, оскільки відбувається збіг із цілим рядком через жадібність елемента .*.

Однак, якщо після квантифікатора йде питання, він перестає бути жадібним і збігається мінімально можливу кількість разів, тому патерн/*.*?*/вірно виконується з С-коментарями. Значення різних інших квантифікаторів не зміниться, лише переважна кількість збігів. Не плутайте це використання питання питання з його використанням як власне квантифікатор. Оскільки він може використовуватися подвійно, вони іноді може з'являтися подвоєним: що збігається з однією цифрою, переважно, але може збігатися і з двома, якщо це єдиний спосіб збігу частини патерну, що залишилася.

Якщо встановлено опціюPCRE_UNGREEDY(відсутня в Perl), то квантифікатори не жадібні за замовчуванням, але окремі можуть бути жадібними, якщо після них стоїть знак питання. Інакше кажучи, питання інвертує поведінка за умовчанням.

Коли субпатерн у дужках квантифікований мінімальною кількістю повторень, яка більша за 1, або має обмеження максимуму, для відкомпільованого патерну потрібно більше місця, пропорційно розміру мінімуму або максимуму.

Якщо патерн починається з .* або з .{0,} та встановлено опціюPCRE_DOTALL(еквівалентна Perl'івській /s), дозволяючи таким чином збіг . з символами нового рядка, то патерн неявно заякорюється, оскільки все, що йде слідом, випробовуватиметься щодо кожної символьної позиції в рядку-суб'єкті, тому після першої немає іншої позиції для відновлення спроб знайти повний збіг. PCRE розглядає такий патерн так, якби йому передувало \A. Коли відомо, що рядок-суб'єкт не містить символів нового рядка, краще встановити PCRE_DOTALLякщо патерн починається з .*, щоб отримати цю оптимізацію, або, альтернативно, використовувати ^ для явного позначення заякорювання.

Коли захоплюючий субпатерн повторюється, захопленим значенням є підрядок, що збігається з останньою ітерацією. Наприклад, після того як(tweedle[dume]{3}\s*)+збігається з "tweedledum tweedledee", значенням захопленого підрядка буде "tweedledee". Однак, якщо є вкладені захоплюючі субпатерни, відповідні захоплені значення можуть бути встановлені у попередніх ітераціях. Наприклад, після того як/(a|(b))+/збігається з "aba", значенням другого захопленого підрядка буде "b".

Посилання/BACK REFERENCES

Поза класом символів, зворотний слеш з наступною цифрою більше 0 (і можливими наступними цифрами) є зворотним посиланням на попередній захоплюючий субпатерн (тобто зліва від себе) в патерні, припускаючи, що була достатня кількість попередніх захоплюючих лівих дужок.

Однак, якщо 10-річне число, що йде після backslash, менше 10, воно завжди вважається зворотним посиланням і викликає помилку лише тоді, коли у всьому патерні немає достатньої кількості захоплюючих лівих дужок. Іншими словами, дужки, на які посилаються, не повинні бути ліворуч від посилання для числа менше 10. Див. розділ "Зворотний слеш/Backslash" раніше детальну інформацію про обробку чисел, що йдуть після зворотного слеша.

Зворотне посилання збігається з усім тим, із чим збігається захоплюючий субпатерн у поточному рядку-суб'єкті, а не з тим, з чим збігається сам субпатерн. Тому патерн(sens|respons)e and \1ibility збігається з "sense and sensibility" і "response and responsibility", але не з "sense and

responsibility". Якщо збіг з урахуванням регістру діє у момент появи зворотного посилання, то регістр символів враховується. Наприклад,((?i)rah)\s+\1збігається з "rah rah" і з "RAH RAH", але не з "RAH rah", хоча оригінальний захоплюючий субпатерн збігається без урахування регістру.

В одному субпатерні може бути більше одного зворотного посилання. Якщо субпатерн на даний момент не використовується у певному збігу, то будь-які зворотні посилання на нього зазнають невдачі. Наприклад, патерн(a|(bc))\2завжди зазнає невдачі, якщо починає збігатися з "a" раніше, ніж з "bc". Оскільки може бути до 99 зворотних посилань, всі цифри, що йдуть після backslash, вважаються частиною потенційної зворотної посилання. Якщо патерн продовжується цифровим символом, повинен використовуватися певний обмежувач для закінчення зворотного посилання. Якщо опція

PCRE_EXTENDEDвстановлена, це може бути пробіл. Інакше може використовуватися порожній коментар. Зворотне посилання, яке з'являється всередині дужок, до яких вона звертається, зазнає невдачі, якщо спочатку використовується субпатерн; так, наприклад, (a1) ніколи не збігається. Однак такі посилання можуть використовуватися всередині субпатернів, що повторюються. Наприклад, патерн(a|b\1)+збігається з будь-якою кількістю "a", а з "aba", "ababaa" etc. При кожній ітерації субпатерна зворотне посилання збігається з рядком символів відповідно попередньої ітерації. Щоб це працювало, патерн має бути таким, щоб перша ітерація не повинна була співпадати зі зворотним посиланням. Це можна зробити за допомогою чергування, як у попередньому прикладі або квантифікатором з мінімумом 0.

Твердження/Assertions

Затвердження це перевірка символів, що йдуть слідом або попередніх точці збігу, не використовуючи реально жодних символів. Прості твердження, кодовані як \b, \B, \A, \Z, \z, ^ і $, були розглянуті раніше. Більш складні твердження кодовані як субпатерни. Є твердження двох видів: ті, що дивляться вперед/ahead від поточної позиції, в рядку-суб'єкті, і ті, що дивляться назад/behind.

Субпатерн затвердження збігається звичайним чином, крім того, що він не викликає зміни поточної позиції збігу. Твердження вперед починаються з (?= для позитивного затвердження і з (?! - для негативного затвердження. Наприклад, w+(?=;)збігається зі словом з наступною точкою з комою, але не включає точку з комою в збіг, ifoo(?) !bar)збігається з будь-якою появою "foo", після якої не йде "bar".Зверніть увагу, що схожий патерн(?!foo)barне знайде ніяких входжень "bar", перед якими йде що-небудь, крім "foo"; він знайде, проте, будь-яке входження "bar", оскільки затвердження (?!foo) завжди TRUE, якщо наступні три символи це "bar".

яке може збігатися з двома різними розмірами, але це допускається, якщо переписати його з двома відгалуженнями верхнього рівня:(?<=abc|abde)Твердження назад/lookbehind реалізовані так, що, для кожної альтернативи, тимчасово зсувається поточна позиція на фіксовану ширину, а потім робиться спроба знайти збіг. Якщо перед поточною позицією недостатньо символів, збіг вважається невдалим. Твердження назад у поєднанні з once-only субпатернами можуть особливо стати в нагоді для пошуку збігів в кінці рядків; Приклад дано в кінці цього розділу для once-only субпатернів. тимчасово зсувається поточна позиція на фіксовану ширину, а потім робиться спроба знайти збіг. Якщо перед поточною позицією недостатньо символів, збіг вважається невдалим. Твердження назад у поєднанні з once-only субпатернами можуть особливо стати в нагоді для пошуку збігів в кінці рядків; Приклад дано в кінці цього розділу для once-only субпатернів. тимчасово зсувається поточна позиція на фіксовану ширину, а потім робиться спроба знайти збіг. Якщо перед поточною позицією недостатньо символів, збіг вважається невдалим. Твердження назад у поєднанні з once-only субпатернами можуть особливо стати в нагоді для пошуку збігів в кінці рядків; Приклад дано в кінці цього розділу для once-only субпатернів.

Різні твердження (будь-якого виду) можуть слідувати один за одним. Наприклад,(?<=\d{3})(?<!999)fooзбігається з "foo" з попередніми трьома цифрами, які не є "999". Зауважте, що кожне твердження застосовується незалежно у тій самій точці рядка-суб'єкта. Спочатку виконується перевірка, що попередні три символи це цифри, потім перевіряється, що ці три цифри не є числом "999". Цей патерн не збігається з "foo" із попередніми шістьма цифрами, перші з яких є цифрами, а останні три не утворюють "999". Наприклад, він не збігається з "123abcfoo". Це зробить патерн(?<=\d{3}...)(?<!999)fooНа цей раз перше твердження переглядає попередні шість символів, перевіряючи, що перші три є цифрами, а потім друге твердження перевіряє,

Твердження можуть вкладатися у будь-якому поєднанні. Наприклад,(?<=(?<!foo)bar)bazзбігається з "baz" з попереднім "bar", перед яким, у свою чергу, немає "foo", а(?<=\d{3}(?!). 999)...)fooце інший патерн, який збігається з "foo" з попередніми трьома цифрами та будь-якими трьома символами - не "999".

Субпатерни тверджень не є захоплюючими субпатернами і не можуть повторюватися, оскільки немає сенсу стверджувати те саме кілька разів (це дивлячись у якій країні... - прим. перекл.). Якщо твердження будь-якого типу містить захоплюючі субпатерни, вони обраховуються для нумерації захоплюючих субпатернів всього патерну. Проте захоплення підрядок виконується лише позитивних тверджень, оскільки це немає сенсу для негативних тверджень.

Твердження обраховую максимум до 200 субпатернів.

Once-only/"Одноразові" субпатерни

І при мінімальній, і при максимальній кількості повторень, невдача того, що йде слідом, нормально викликає повторне обчислення повторюваного елемента, з метою перевірити, чи не дасть збігу повторення, іншу кількість разів, що залишилася, частини патерну. Іноді потрібно запобігти цьому зміни природи збігу, або щоб викликати невдачу раніше, ніж це могло б бути, якщо автор патерна знає, що більше немає точок для роботи.

Розглянемо, наприклад, патерн \d+foo у застосуванні до рядка-суб'єкта123456barПісля збігу всіх шести цифр і невдачі збігу з "foo", нормальною дією буде - спробувати знову тільки вже з п'ятьма цифрами \d+ item, а потім з чотирма, і так далі , Перш ніж остаточно зазнати невдачі. Одноразові субпатерни надають засіб для специфікування дій, як тільки частина патерну збіглася: тоді він не обчислюється повторно, і пошук збігів негайно завершується за першої невдачі збігу з "foo". Нотується це іншим видом спеціальних дужок, що починаються з (?>(?>\d+)barЦей вид дужок "переглядає" частину патерну, що міститься в них, якщо вона збіглася, а подальша невдача в патерні запобігає повернення і повторний перегляд. елементам, однак,

Альтернативний опис такий, що субпатерн цього типу збігається з рядком символів, з яким міг би збігтися ідентичний окремий субпатерн, якщо він заякорений у поточній точці рядка-суб'єкта.

Одноразові субпатерни є захоплюючими субпатернами. Прості випадки, на кшталт вищенаведеного прикладу, можна подати як максималістську побудову, яка поглинає все, що може. Так, тоді як \d+ та \d+? підготовлені так, щоб уточнювати число цифр для збігу з ними, щоб збіглася решта патерну (?>\d+) може збігатися тільки з повною послідовністю цифр.

Ця конструкція, зрозуміло, може містити довільно ускладнені субпатерни і вкладатися.

Одноразові субпатерни можна використовувати у поєднанні із твердженнями назад для специфікації ефективного збігу в кінці рядка-суб'єкта. Розглянемо такий простий патерн: abcd $ застосований до довгого рядка, з яким він не збігається. Оскільки збіг виконується зліва направо, PCRE шукатиме кожну "a" в суб'єкті, а потім подивиться, що з наступного збігається із залишком патерну. Якщо патерн специфікований так:^.*abcd$то початкове .* збігається спочатку з усім рядком, але якщо це зазнає невдачі (оскільки немає наступних "a"), виконується повернення і проглядається все, крім останнього символу, потім все, крім останніх двох символів і так далі. Пошук "a" покриває весь рядок, праворуч наліво, і нічого не може бути краще. Однак, якщо патерн записано так: ^(?>.*)(?< =abcd)то може бути повернення назад для елемента .* item; він може збігтися тільки з цілим рядком. Наступне твердження назад виконує просту перевірку останніх чотирьох символів. Якщо вона зазнає невдачі, збіг зазнає невдачі негайно. Для довгих рядків цей підхід має значну відмінність у часі виконання.

Якщо патерн містить необмежену повторення всередині субпатерну, який сам може повторюватися необмежену кількість разів, використання once-only субпатерну виявляється єдиним способом уникнути невдалих збігів, які тривають досить тривалий час. з необмеженою кількістю підрядків, які складаються з не-цифр, або з цифр, ув'язнених в <>, з наступним ! або?. Якщо збіг є, він виконується швидко. Однак, якщо застосує патерн
каа буде

витрачено багато часу, перш ніж буде повідомлено про невдачу. Це тому, що цей рядок може бути поділена на два повторення великою кількістю способів, і всі вони будуть випробувані. (Приклад використовує [!?] замість простого символу в кінці, так як і PCRE, та Perl містять оптимізацію, яка дозволяє прискорити виявлення невдачі, якщо використовується одиночний символ. вони запам'ятовують останній одиночний символ, необхідний для збігу, і видають невдачу раніше, якщо цей символ відсутній у рядку.) Якщо змінити патерн так((?>\D+)|<\d+>)*[!?]послідовності не-цифр не можуть бути розірвані, і невдача виявляється швидше.

Умовні субпатерни

Можна змусити субпатерн у процесі збігу підкорятися умовно або вибирати з двох альтернативних субпатернів, залежно від результату затвердження або від того, чи збігся попередній захоплюючий субпатерн чи ні. Ось дві можливі форми умовного субпатерну:(?(condition)yes-pattern)(?(condition)yes-pattern|no-pattern)Якщо умова виконана, використовується yes-pattern; інакше використовується no-pattern (якщо є). Якщо субпатерне є більше двох альтернатив, виникає помилка компіляції.

Є умови двох видів. Якщо текст між дужками складається із послідовності цифр, то умова виконана, якщо захоплюючий субпатерн цього числа раніше збігся. Розглянемо наступний патерн, який містить незначну прогалину для зручності читання (припустимо наявність опціїPCRE_EXTENDED) і для поділу патерну на три частини для полегшення обговорення:(\( )?[^()]+(?(1) \))Перша частина збігається з необов'язковою дужкою, що відкриває, і, якщо символ є, встановлює

його як першу збіглу підрядок.Друга частина збігається з одним або більше символів, які не є дужками. Третя частина це умовний субпатерн, який перевіряє, чи збігається перший набір дужок чи ні. Якщо збігся, тобто якщо суб'єкт починається з дужки, що відкриває, умова буде TRUE, буде виконуватися yes-pattern і необхідна закриваюча дужка. Інакше оскільки no-pattern відсутня, субпатерн не збігається ні з чим. Іншими словами, цей патерн збігається з послідовністю не-дужок, можливо, укладеної у дужки.

Якщо умова перестав бути послідовністю цифр, вона має бути твердженням. Це може бути позитивне або негативне lookahead або lookbehind твердження. Розглянь патерн, знову містить незначний пробіл і дві альтернативи в другому рядку: }-\d{2} |\d{2}-\d{2}-\d{2})Умова це позитивне випереджаюче/lookahead твердження, яке збігається з необов'язковою послідовністю не-літер з наступною літерою. Іншими словами, воно перевіряє наявність мінімум однієї літери у суб'єкті. Якщо букву знайдено, знову перевіряється збіг суб'єкта з першою альтернативою; інакше перевіряється збіг із другою альтернативою. Цей патерн збігається з рядками однієї з двох форм: dd-aaa-dd або dd-dd-dd, де aaa – це букви, а dd – це цифри.

Коментарі

Послідовність символів (?# означає початок коментаря, який триває до наступної дужки, що закриває. Вкладення дужок не допускається. Символи, що утворюють коментар, не є частиною збігу патерну.

Якщо встановлено опціюPCRE_EXTENDED, не escape-ований символ # поза класом символів починає коментар, який продовжується до наступного символу нового рядка в патерні.

Рекурсивні паттерни

Розглянемо проблему збігу рядка у дужках, коли допускається необмежене вкладення дужок. Без використання рекурсії, найкраще, що можна зробити, це використовувати патерн, який збігається на деяку фіксовану глибину вкладення. Неможливо обробити вкладення довільно велику глибину. Perl 5.6 має експериментальну можливість, що

дозволяє (крім іншого) виконувати рекурсію регулярних виразів. Спеціальний елемент (?R) надано для цього специфічного case/варіанту рекурсії. PCRE-патерн вирішує проблему дужок (припускаючи, що опція PCRE_EXTENDEDвстановлена так, що пробіл ігнорується):\(((?>[^()]+) | (?R))* \) Спочатку він збігається з відчиняючою дужкою. Потім - з будь-якою кількістю підрядків, які можуть бути або послідовностями не-дужок, або рекурсивним збігом самого патерну (тобто коректно укладеного в дужки підрядком). Нарешті йде закриваюча дужка.

Цей особливий приклад патерну містить вкладене нескінченне повторення, і, таким чином, використання одноразового субпатерну для збігу з рядками з не-дужок дуже важливо, коли патерн застосовується до рядків, які не збігаються. Наприклад, якщо його застосувати до (aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa) то він швидко дасть "немає збігів". Однак, якщо одноразовий/once-only субпатерн не використовується, пошук можуть кроїти рядок-суб'єкт, і всі вони повинні бути перевірені, перш ніж буде видано повідомлення про невдачу пошуку.

Значення, встановлені для будь-якого субпатерну, беруться із зовнішнього рівня рекурсії, на якому встановлюється значення субпатерну. Якщо вищенаведений патерн підставити щодо (ab(cd)ef) то значенням для захоплюючих дужок буде "ef", яке є останнім значенням, що приймається на верхньому рівні. Якщо додати додаткові дужки\((((?>[^()]+) | (?R))*)\)

то рядок, що захоплюється ними, буде "ab(cd)ef" - вміст дужок верхнього рівня. пам'ять потім через pcre_free. Якщо неможливо виділити пам'ять, він зберігає дані тільки для перших 15 захоплюючих дужок, тому що немає способу видати помилку out-of-memory зсередини рекурсії.

Продуктивність

Деякі елементи, які можуть з'являтися у патернах, працюють ефективніше, ніж інші. Більш ефективно використовувати клас символів, такий як [aeiou], ніж набір альтернатив, такий як (a|e|i|o|u). В цілому, простіша конструкція є більш ефективною. Книга Jeffrey Friedl'a містить велику дискусію щодо оптимізації регулярних виразів для підвищення продуктивності.

Якщо патерн починається з .* та встановлено опціюPCRE_DOTALLПатерн неявно заякорюється PCRE, оскільки він може збігтися тільки на початку рядка-суб'єкта. Однак, якщоPCRE_DOTALLне встановлена, PCRE не може виконати цю оптимізацію, оскільки метасимвол . не збігається тоді із символом нового рядка/newline, і, якщо рядок-суб'єкт містить newlines, патерн може збігтися із символом, що йде безпосередньо після одного із символів нового рядка, замість того, щоб збігатися лише на самому початку. Наприклад, патерн(.*) second

збігається в суб'єкті "first\nand second" (де \n це символ нового рядка) з першим захопленим підрядком "and". Для цього PCRE намагається збігатися спочатку після кожного символу нового рядка в суб'єкті.

Якщо ви використовуєте такий патерн з рядками-суб'єктами, які не містять newlines, найкраща продуктивність буде досягнута установкоюPCRE_DOTALL, або якщо розпочати патерн з ^.* для явного вказівки заякорювання. Це утримає PCRE від необхідності сканувати суб'єкт у пошуках newline для рестарту з нього.

Уникайте створення паттернів, які містять нескінченні повторення. Вони можуть зайняти багато часу, якщо застосувати їх до рядка, який не містить збігів. Розглянемо фрагмент патерну

(a+)*

Він може збігтися з "aaaa" 33 різними способами, і ця кількість збільшується дуже швидко зі збільшенням довжини рядка. (Повторення * може збігтися 0, 1, 2, 3 або 4 рази, і для кожного випадку/case, відмінного від 0, повторення + можуть збігатися різну кількість разів.) Якщо залишок патерну такий, що весь збіг зазнає невдачі, PCRE має, в принципі, спробувати виконати всі можливі варіанти, і це може вимагати величезної кількості часу.

За допомогою оптимізації можна відловити найпростіші випадки, такі як

```
(a+)*b
```

де слідом йде літеральний символ. Перш ніж покладатися на стандартну процедуру пошуку збігів, PCRE перевіряє, чи є "b" далі в рядку-су-

б'єкті, і якщо ні, збіг негайно завершується невдачею. Однак, коли наступного літералу немає, ця оптимізація не може бути використана. Відчуйте різницю, порівнявши поведінку

```
(a+)*\d
```

з поведінкою вищенаведеного патерну. Перший видає невдачу одразу, коли застосовується до рядка символів "a", а другий витрачає значний час на пошук у рядках довжиною понад 20 символів.

4.2. Cookies

PHP прозоро підтримує HTTP cookies. Cookies це механізм зберігання даних броузером віддаленої машини для відстеження або ідентифікації відвідувачів, що повертаються. Ви можете встановити cookies за допомогою функцій setcookie() або setrawcookie(). Cookies є частиною HTTP-заголовка, тому setcookie() має викликатися до будь-якого виведення даних у броузер. Це те саме обмеження, яке має функція header(). Ви можете використовувати функції буферизації виводу, щоб затримати виведення результатів роботи скрипта до того моменту, коли буде відомо, чи знадобиться встановлення cookies або інших заголовків HTTP.

Будь-які cookies, надіслані серверу броузером клієнта, будуть автоматично перетворені на змінні PHP, подібно до даних методів GET і POST. На цей процес впливають конфігураційні директиви register_globals та variables_order. Для призначення кількох значень однієї cookie просто додайте [] до її імені.

У PHP 4.1.0 і вище глобальний масив $_COOKIE завжди автоматично заповнюється значеннями отриманих cookies. У попередніх версіях визначається масив $HTTP_COOKIE_VARS але тільки коли включена директива track_vars. (Ця директива завжди включена з PHP 4.0.3.)

Cookie є невеликим пакетом інформації, переданим web-сервером і що зберігається на клієнтському комп'ютері. У cookie можна зберегти корисні дані, що описують стан сеансу користувача, щоб у майбутньому завантажити їх і відновити параметри сеансового зв'язку між сервером і клієнтом. Cookie використовуються на багатьох сайтах Інтернету для розширення можливостей користувача та підвищення ефективності сайту за рахунок відстеження дій та особистих уподобань користувача. Можливість зберігання цих відомостей відіграє ключову роль на сайтах електронної комерції, що підтримують персональне налаштування та цільову рекламу.

Внаслідок того, що cookie зазвичай зв'язуються з конкретним користувачем, вони часто зберігають унікальний ідентифікатор користувача (UIN). Цей ідентифікатор заноситься до бази даних на сервері і використовується як ключ для вибірки з бази всієї інформації, пов'язаної з цим ідентифікатором. Звичайно, збереження UIN у cookie не є обов'язковою вимогою; Ви можете зберегти будь-яку інформацію за умови, що її загальний обсяг не перевищує 4 Кбайт (4096 байт).

Компоненти cookie

В cookie зберігаються інші компоненти, за допомогою яких розробник може обмежувати використання cookie з позицій домену, шляху, терміну дії та безпеки. Нижче наведено опис різних компонентів cookie:

- Ім'я-- ім'я cookie є обов'язковим параметром, яким програма посилається на cookie. Можна провести аналогію між ім'ям cookie та ім'ям змінної.
- Значення- Фрагмент даних, пов'язаний з ім'ям cookie. У цих даних може зберігатися будь-яка інформація - ідентифікатор користувача, колір тла, поточна дата тощо.
- Термін дії- Дата, що визначає тривалість існування cookie. Як тільки поточна дата та час перевищують заданий термін дії, cookie стає недійсним та перестає використовуватися. Відповідно до специфікації cookie встановлювати термін дії для cookie необов'язково. Тим не менш, PHP для роботи з cookie вимагають, щоб термін дії встановлювався. Згідно зі специфікацією, якщо термін дії не вказаний, cookie стає недійсним наприкінці сеансу (тобто, коли користувач залишає сайт).
- Домен- Домен, який створив cookie і може читати його значення. Якщо домен складається з декількох серверів і доступ до cookie повинен бути дозволений всім серверам, ім'я домену можна задати у формі .phprecipes.com. У цьому випадку всі потенційні домени третього рівня, що належать сайту PHPrecipes (наприклад, wap.phprecipes.com або news.phprecipes.com), зможуть працювати з cookie. З міркувань безпеки cookie можуть встановлюватися лише домену сервера, який намагається створити cookie. Цей компонент необов'язковий; якщо він не вказаний, за замовчуванням використовується ім'я домену, з якого була підлога; чено значення cookie.
- Шлях-- URL, з якого надається доступ до cookie. Будь-які спроби отримати доступ до cookie за межами цього шляху припиняються. Цей компонент необов'язковий; якщо він не заданий, за умовчанням використовується шлях до документа, який створив cookie.
- Безпека-- параметр, який показує, чи допускається читання cookie у небезпечному середовищі. За промовчанням використовується значення FALSE.

Встановлення cookies

Установка cookies виконується за допомогою функції setcookie:

setcookie

Синтаксис:

*bool*setcookie (string name [, string value [, int expire [, string path [, string domain [, int secure]]]]])

Ця функція має такі аргументи:

- name- ім'я встановлюваного cookie;
- value- значення, яке зберігається в cookie з ім'ям $name;
- expire- час у секундах з початку епохи, після якого поточний cookie стає недійсним;
- path- шлях, яким доступний cookie;
- domain- Домен, з якого доступний cookie;
- secure- директива, яка визначає, чи доступний cookie не за запитом HTPPS. За промовчанням ця директива має значення 0, що означає можливість доступу до cookie за звичайним запитом HTTP.

Приклад простого застосування з cookies

Створимо простий сценарій, який підраховує за допомогою cookies кількість звернень відвідувача до сторінки.

У cookie з ім'ям counter зберігатиметься кількість відвідувань сторінки користувачем:

```
<?
 $counter++;
 setcookie("counter", $ counter);
 echo("Ви відвідали цю сторінку $counter разів");
?>
```

При роботі з cookie необхідно враховувати важливий момент, що полягає в тому, що cookie треба обов'язково встановлювати перед відправкою в броузер будь-яких заголовків, оскільки самі cookie встановлюються у вигляді заголовків. Тому якщо встановити cookies після будь-якого тексту, що надсилається в броузер, виникне хибна ситуація.

Розглянемо це питання докладніше, навіщо модифікуємо код, наведений у лістингу, помістивши перед установкою cookie текст:

```
 Якщо тут помістити текст, то виникне помилка, оскільки
 буде надіслано заголовок Content-type: text/html.
<?
 $counter++;
 setcookie("counter", $ counter);
 echo("Ви відвідали цю сторінку $counter разів");
?>
```

В даному випадку ми отримаємо повідомлення про помилку: headers already sent.

Зауважимо, що у цих лістингах ми звертаємося до змінної $counter, у якій зберігається значення cookie, як до глобальної, що вимагає наявності включеної директиви register_globals. Якщо ця директива вимкнена, то значення, що зберігається в cookie, можна отримати через глобальні масиви $HTTP_COOKIE_VARS["name"] і $_COOKIE["name"]:

```
<?
 $_COOKIE[ 'counter'] ++;
 setcookie("counter", $ counter);
 echo'Ви відвідали цю сторінку '.$_COOKIE['counter'].' раз';
?>
```

Встановлення терміну придатності cookies

За замовчуванням cookies встановлюються на один сеанс роботи з броузером, проте можна задати їм більш тривалий термін існування. Це дуже зручна та корисна властивість, оскільки в цьому випадку користувачу не потрібно надавати свої дані знову при кожному відвідуванні сайту.

Як мовилося раніше, термін придатності встановлюється у секундах щодо початку епохи. У PHP існують функції time та mktime для роботи з датою та часом, що дозволяють переводити поточний час у кількість секунд від початку епохи. Функція time просто переводить поточний системний час у кількість секунд, що минули початку епохи:

time

Синтаксис:

time*();*

Удосконаленим варіантом функції time є функція mktime:

mktime

Синтаксис:

*int*mktime*(*[int hour [, int minute [, int second [, int month [, int day[, int year [, int is_dst]]]]]]])

Аргумент is_dst цієї функції визначає, чи ця дата потрапляє в період літнього часу і може приймати наступні значення:

- -1 (За замовчуванням. Це означає, що властивість не задано);
- 0 (Тимчасовий інтервал не припадає на період літнього часу);
- 1 (Тимчасовий інтервал припадає на період літнього часу).

Приклади встановлення термінів придатності cookies:

```
<?
 /* цей cookie дійсний протягом 10 хв після створення */
 setcookie("name", $ value, time() + 600);
 /* дія цього cookie припиняється опівночі 25 січня 2010 року */
```

```
setcookie("name", $value, mktime(0,0,0,01,25,2010));
/* дія цього cookie припиняється о 18.00 25 січня 2010 року */
setcookie("name", $value, mktime(18,0,0,01,25,2010));
?>
```

Видалення cookie

Видалити cookie легко. Для цього потрібно викликати функцію setcookie і передати їй ім'я того cookie, який підлягає видаленню:

```
setcookie("name");
```

Інші встановлені cookie не видаляються.

Проблеми безпеки, пов'язані з cookies

Іноді cookies доводиться зберігати конфіденційні дані, і в цьому випадку розробник повинен подбати про те, щоб інформація, що зберігається в cookie, не була передана третім особам. Існує кілька методів захисту інформації, що зберігається в cookie:

- встановлення області видимості cookies;
- шифрування;
- обмеження доступу для доменів;
- надсилання cookies за захищеним запитом.

Найкращим рішенням є комплексне застосування всіх цих методів.

Встановлення області видимості cookie

Оскільки, за замовчуванням, доступ до cookie походить з кореневого каталогу, це може створити "дірки" в системі захисту, оскільки cookies стають доступними у будь-якому підкаталозі цього каталогу. Обмежити доступ до cookies для всіх сторінок, крім розміщених у конкретному каталозі, наприклад, /web, можна так:

```
setcookie("name", $value, "/web/");
```

Однак і в цьому випадку, наприклад, каталоги /web/index.php, /web1/page.html тощо будуть задовольняти цьому обмеженню. Якщо таке положення також небажане, можна обмежити область видимості cookies до конкретної сторінки:

```
setcookie("name", $value, "/web/index.php");
```

Однак і такий спосіб повною мірою не вирішує проблему, тому що в цьому випадку доступ до інформації, яка міститься в cookie, може отримати, наприклад, скрипт /web/index.php-script/anti_cookie.php. Тому виникає потреба у шифруванні.

4.3. Сесії у PHP

Сесії та cookies призначені для збереження відомостей про користувачів під час переходів між кількома сторінками. Під час використання сесій дані зберігаються у тимчасових файлах на сервері.

Використання сесій та cookies дуже зручне та виправдане у таких додатках як Інтернет-магазини, форуми, дошки оголошень, коли, по-перше, необхідно зберігати інформацію про користувачів протягом кількох сторінок, а по-друге, своєчасно надавати користувачеві нову інформацію.

Протокол HTTP є протоколом "без збереження стану". Це означає, що даний протокол не має вбудованого способу збереження між двома транзакціями. Т. е., коли користувач відкриває спочатку одну сторінку сайту, а потім переходить на іншу сторінку цього ж сайту, то ґрунтуючись лише на засобах, що надаються протоколом HTTP неможливо встановити, що обидва запити належать до одного користувача. Т. о. потрібний спосіб, за допомогою якого було б відстежувати інформацію про юзера протягом одного сеансу зв'язку з Web-сайтів. Одним із таких методів є керування сеансами за допомогою призначених для цього функцій. Для нас важливо те, що сеанс по суті є групою змінних, які, на відміну від звичайних змінних, зберігаються і після завершення виконання PHP-сценарію.

Працюючи з сесіями розрізняють такі этапы:

- відкриття сесії
- реєстрація змінних сесії та їх використання
- закриття сесії

Відкриття сесії

Найпростіший спосіб відкриття сесії полягає у використанні функції session_start, яка викликається на початку PHP-сценарію:

session_start

Синтаксис:

session_start*();*

Ця функція перевіряє, чи існує ідентифікатор сесії, і, якщо ні, створює його. Якщо ідентифікатор поточної сесії вже існує, завантажуються зареєстровані змінні сесії.

Реєстрація змінних сесії

Взагалі, реєстрація змінних сесії здійснюється за допомогою функції session_register, проте, починаючи з версії PHP 4.2.0, практика реєстрування сеансових змінних зазнала деяких змін, що вносить у це питання деяку плутанину. Справа в тому, що функція session_register застосовується лише до глобальних змінних і вимагає, щоб параметр register_globals був увімкнений. Починаючи з цієї версії мови, сеансові змінні завжди реєструються в асоціативних масивах $HTTP_SESSION_VARS та $_SESSION. Т.ч., сесії коректно відкривати присвоєнням значення глобальної змінної:

$_SESSION['username'] = "username";

```
// або
$HTTP_SESSION_VARS['username'] = "username";
```

Надалі під час виконання лабораторних робіт рекомендується використовувати змінну $_SESSION.

Таким чином:

- якщо ви використовуєте функцію session_register, то має бути увімкнено параметр register_globals
- **при відключеному register_globals ініціалізація сеансових змінних відбувається шляхом надання значень елементам асоціативних масивів**

Для коректної роботи програми необхідно перевіряти, чи встановлені змінні сесії.

При використанні асоціативних масивів $HTTP_SESSION_VARS і $_SESSION необхідно безпосередньо перевіряти елементи цих масивів, наприклад, так:

```
if(isset($_SESSION['username']))
```

Закриття сесії

Після завершення роботи із сесією спочатку потрібно розреєструвати всі змінні сесії, а потім викликати функцію session_destroy:

session_destroy

Синтаксис:

```
session_destroy();
```

Способи розреєстрації сеансових змінних різняться залежно від цього, як вони зареєстровані. У разі використання масиву $_SESSION розреєстрація здійснюється викликом функції unset.

unset

Синтаксис:

```
unset($_SESSION["username"]);
```

Приклад простої сесії

Розглянемо приклад простої сесії, що працює із трьома сторінками. Під час відвідування користувачем першої сторінки відкривається сесія та реєструється змінна $username. Відповідний код реалізації:

```
<?
session_start();
$_SESSION['username'] = "alex";
echo'Привіт, '.$_SESSION['username']."<br>";
?>
```

```
 <a href="page2.php">На наступну сторінку</a>
```

Після цього користувач alex натискає на посилання і потрапляє на сторінку page2.php, код якої наведено в лістингу:

```
<?
 session_start();
 echo$_SESSION['username'].' , ти прийшов на іншу сторінку цього сайту!';
 echo("<br>");
?>
 <a href="page3.php">На наступну сторінку</a>
```

При натисканні на посилання користувач потрапляє на сторінку page3.php, при цьому відбувається розреєстрація сеансової змінної та знищення сесії. Відповідний код реалізації наведено у лістингу:

```
<?
 session_start();
 unset($_SESSION['username']); // Розреєстрували змінну
 echo'Привіт, '.$_SESSION['username'];
 /* тепер ім'я користувача вже не виводиться */
 session_destroy();// руйнуємо сесію
?>
```

Після розреєстрації сеансової змінної значення масиву $_SESSION['username'] вже недоступне, і буде виведена лише фраза «Привіт, ».

5. Контрольні питання

1. Які специфікації регулярних виразів ви знаєте?
2. Опишіть елементи, що застосовуються в шаблонах.
3. Як організуються альтернативи шаблону?
4. Що таке символьний клас?
5. Як і навіщо використовуються квантифікатори?
6. Що таке підшаблони?
7. Які спец. символи ви знаєте?
8. Які бувають перевірки при збігу частини шаблону?
9. Що таке твердження?
10. Як реалізовано механізм cookies?
11. Що таке сесії та як їх використовувати?

6. Варіанти завдань

У кожному варіанті вихідна форма документа повинна містити заголовок з назвою роботи, внизу документа – ПІБ виконавця (можна брати з попередньої роботи).

Сервер запускається/зупиняється файлами run.exe/stop.exe з папки /webservers/etc/ , яка знаходиться у робочій директорії.

Файли ЛР повинні бути розміщені (при запущеному сервері) у папці *z:/home/localhost/www/прізвище_виконавця/*

та будуть відповідно доступні за адресою

http://localhost/прізвище_виконавця/ім'я_скрипту.php

Усі варіанти завдань базуються на завданнях попередньої лабораторної роботи та є їх продовженням

Варіанти:

1. "Онлайн газета".

Здійснити очищення тексту від будь-яких тегів, крім <i>, <b>, , , <h1>, <p>,
, <a>.

Перетворити всі посилання на Інтернет-ресурси (http://....) в правильні html-посилання (<a href=….).

Встановити (і оновлювати) cookie з інформацією про максимальний номер (id) переглянутої статті та додати можливість перегляду лише нових статей.

Організувати зберігання у змінній сесії останнього пошукового запиту та виводити його поруч із формою пошуку.

2. "Міні фотогалерея".

Перевірити ім'я автора за такими правилами: ім'я може містити літери (латинський алфавіт), цифри та знаки «-» та «_», має починатися з великої літери, довжина імені від 6 до 12 символів, ім'я може містити цифри у будь-якій позиції, крім першої та останньої. Якщо ім'я не відповідає шаблону привести його до потрібного вигляду (обрізати за максимальною довжиною або додати знаками «-» до хв. довжини; якщо перший або останній символи є цифрами замінити їх на літеру «a» з відповідним регістром).

Записати в cookie ім'я автора та заповнювати поле для введення автора автоматично (у наступних запитах).

У змінній сесії зберігати ім'я останнього завантаженого файлу та виводити його поряд із формою для завантаження.

3. "Гостьова книга"

Видалити всі теги розмітки HTML з тіла повідомлення.

Перевірити адресу електронної пошти на коректність за допомогою регулярного виразу. У разі неправильної адреси виводити попередження (повідомлення до БД не вносити, а заново виводити у формі для додавання). Після 3 невдалих спроб введення (для рахунку спроб використовувати сесії) встановити значення cookie, за наявності якого блокується форма для додавання (або як варіант - виводиться попередження про перевищення ліміту спроб введення).

4. "Прайс лист"

Встановити та зберігати в cookie останній режим роботи програми (додавання, редагування або перегляд прайс-листа) та повертатися до цього режиму під час виклику програми без параметрів.

Зберігати у змінних сесії найменування останнього доданого та віддаленого товарів.

При введенні найменування товару перевіряти відсутність спеціальних знаків (_, \, /, *, @, #, ^, &), замінювати / і \ на «-», інші на прогалини і виводити попередження.

5. «Список студентів»

Зберігати в cookie останній вибраний режим сортування та повертатися до нього під час скрипту без параметрів.

Вважати, зберігати у змінній сесії та виводити кількість доданих та віддалених студентів.

При введенні перевіряти правильність ПІБ студента (відсутність цифр і спеціальних знаків, перша заголовна буква, відсутність прогалин у імені, прізвища та по батькові), при необхідності призводити до потрібного вигляду та виводити попередження.

6. «Посторінковий висновок записів»

Зберігати в cookie номер сторінки, що переглядається, і переходити до неї при виклику скрипта без параметрів.

Зберігати у змінній сесії номер попередньої перегляданої сторінки та виводити його.

Перетворити текст, виділений тегами на великі літери (використовувати можливість виконання PHP-коду під час заміни), а текст, виділений тегами обрамляти в круглі дужки.

6. Список літератури, що рекомендується

1. Розробка Web-додатків на PHP та MySQL: Пер. з англ. / Лаура Томсон, Люк Веллінг. - 2-ге вид., Випр. - СПб: ТОВ «ДіаСофт», 2003. - 672 с.
2. Аргеріх Л. та ін. Професійне PHP програмування, 2-ге видання. - Пров. з англ. – СПб: Символ-Плюс, 2003. – 1048 с., іл.
3. Кузнєцов І.В., Симдянов І.В., Голишев С.В. PHP 5. Практика розробки веб-сайтів. - СПб.: БХВ-Петербург, 2005. - 960 с.
4. Кузнєцов І.В., Симдянов І.В., Голишев С.В. PHP 5 на прикладах. - СПб.: БХВ-Петербург, 2005. - 576 с.
5. Коггзолл, Джон. PHP5. Повне керівництво: Пер. з англ. - М.: Видавничий дім "Вільямс", 2006. - 752 с.
6. Колісніченко Д.М. Самовчитель PHP 5. СПб.: Наука та техніка, 2004. - 576 с.
7. http://www.php.net
8. http://www.mysql.org