



DEO

Osnove PHP-a

U OVOM DELU:

1. Upoznavanje PHP-a
2. Promenljive
3. Operatori i izrazi
4. Kontrola toka i funkcije
5. Stringovi i nizovi

Upoznavanje PHP-a

RAZVOJ APLIKACIJA I SAJTOVA ZA WORLD WIDE WEB, ILI ZA UPOTREBU SLIČNU WEBU (INTRANET) PREDSTAVLJA OBLAST U KOJOJ SE RAČUNARI DANAS NAJVIŠE KORISTE. AKO NEŠTO MOŽE DA SE URADI NA DIGITALAN NAČIN, BUDITE SIGURNI DA NEKO NEGDE POKUŠAVA DA TO PRILAGODI DA RADI PREKO WEB PRETRAŽIVAČA. AKO ŽELITE DA SVOJU FLEKSIBILNOST KAO PROGRAMER POSTAVITE NA NOVI NIVO, MORATE DA POZNAJETE RAZLIČITE WEB AKTIVNOSTI, KAO ŠTO SU STATIČKE I DINAMIČKE STRANE, SISTEMI NA STRANI KLIJENTA I SERVERA, ITD.

PHP JE NAPRAVLJEN NA OSNOVU POZNATE STRUKTURE JEZIKA KAO ŠTO SU C, JAVA, ILI PERL. ON POMAŽE DA SE KREIRA DINAMIČKI HTML SADRŽAJ, TAKO ŠTO OBEZBEĐUJE POTREBNE ALATE ZA MANIPULISANJE SADRŽAJEM. PHP JE POSTAO JEDAN OD ISTAKNUTIH ALATA ZA POVEĆANJE SNAGE WEB STRANA, POŠTO SE LAKO KORISTI, A IPAK JE PRILIČNO SNAŽAN. KREIRANJE NEKOLIKO ELEMENTARNIH SKRIPTOVA, TESTIRANJE DVA GLAVNA METODA ZA PREBACIVANJE PODATAKA I UPOZNAVANJE KOMENTARA U PHP KODU ĆE VAM POKAZATI KAKO SE PRISTUPA RAZLIČITIM PHP MOGUĆNOSTIMA.

Razvoj za Web

Termin Web razvoj ukazuje na široke i duge poteze. To je opšti termin kojim se kategorizuju različite aktivnosti. Web razvoj može da znači bilo šta, od postavljanja statičke HTML strane na malom WWW sajtu, do kreiranja masivnih intranet mreža velikih preduzeća koje "pokrivaju" više kontinenata i obavljaju različite ključne operacije. Ove aktivnosti mogu, ipak, da se dalje podele u nekoliko drugih kategorija.

Web aplikacije

Ako želite da se bavite razvojem Web aplikacija, morate prvo da ih upoznate. Šta je aplikacija? Šta treba da uradi?

Aplikacija je bilo koji program koji je napravljen da bi bio pojednostavljen, ili obavljen neki zadatak. Nivo zadatka može da se kreće od vrlo specifičnog do opštijeg. Program koji uzima ocene iz šest predmeta nekog studenta, računa srednju vrednost i na kraju to prikazuje u nekom izveštaju je jednostavna, ali ograničena aplikacija. Sa druge strane, aplikacija koja omogućava komunikaciju sa drugima, kao što je online groupware (ona koja omogućava korisnicima da koordiniraju svoje radne tokove), složenija je i služi za neki opštiji cilj. Iako je domen ovakvih aplikacija mnogo širi od domena programa za izračunavanje srednje ocene, ipak je u oba slučaja reč o aplikacijama.

Šta je specifično za Web aplikacije? Generalno govoreći, to su aplikacije koje koriste sveprisutnost i lakoću komunikacije koju nudi Internet. Strožija definicija, ona koju ćemo koristiti u ovoj knjizi, je da je reč o aplikaciji koja kao klijenta koristi pretraživač. Postoje različite tehnologije na strani klijenta koje možete da koristite u pretraživačima. Generalno govoreći, najbolje i najelegantnije aplikacije su one koje koriste jednostavan i elegantan Hypertext Markup Language (HTML). Primeri za isključivo Web aplikacije su bankarski sistem zasnovan na Webu, aukcije i sajtovi sa vestima.

Statički i dinamički Web sajtovi

Statički sajtovi imaju sadržaj koji se ne menja sve dok autor sam nešto ne promeni. Ovakvi sajtovi mogu dobro da posluže većini ljudi, pošto omogućavaju da se informacije dele. Sa druge strane, statički sajtovi ne mogu da obezbede interakciju sa posetiocima i ne mogu da neki zadatak izvršavaju na programabilan način.

Dinamički sajtovi omogućavaju interakciju sa korisnikom. Iako dinamički sajtovi, kao i statički, koriste HTML za interfejs sa klijentom, oni takođe omogućavaju korisnicima da preduzimaju individualne i prilagodljive akcije, kao što rezervisanje, ili kupovina određenog avionskog leta, ili, čak, i sedišta. Cilj koji leži iza onlajn sistema za izdavanje karata je jasan, lak i upotrebljiv interfejs, koji korisniku pruža različite pogodnosti. Sa sistemom kojem se može globalno pristupati preko Weba, posao kupovine karte postaje decentralizovan i lak za izvršavanje.

HTML je tekstualni markap jezik. U idealnim situacijama on se koristi za definisanje sadržaja i skiciranje njegove strukture, dok se CSS (cascading style sheets) datoteke koriste za pozicioniranje i definisanje stila tog sadržaja. Naravno, zbog kompatibilnosti sa onim što je ranije rađeno, CSS

je opcionim izbor. Usled statičke prirode samog HTML jezika (što znači da je reč o jednostavnom tekstualnom jeziku), on sam je ograničen ako želimo da se sadržaj menja i evoluira.

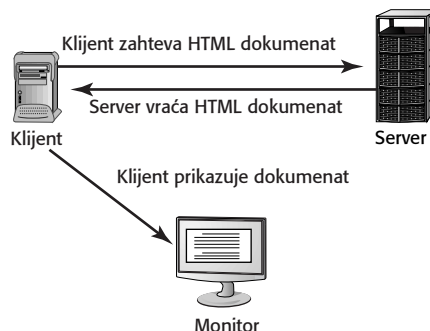
HTML omogućava deobu sadržaja između različitih Web klijenata, ali ima i nekoliko nedostataka. Kada se od Web servera zatraži HTML dokumenat, Web server vraća dokumenat onome ko ga je zatražio. Ne radi ništa više od toga. Ovo je, prema tome, samo način za publikovanje nekog sadržaja, pri čemu nije moguće taj sadržaj kreirati, kontrolisati, organizovati, ili prilagođavati. HTML, onako kako se danas koristi, svoj osnovni cilj vidi u obezbeđivanju vizuelnog kvaliteta sadržaja, a ne u poboljšanju njegove strukture.

Na strani servera, ili na strani klijenta

HTML je tehnologija koja se izvršava na strani klijenta, što znači da se HTML dokumenat u potpunosti obrađuje kod klijenta. Web server nema neko ponašanje koje bi zavisilo od koda koji se nalazi u HTML dokumentu. Posao Web servera je samo da obezbedi datoteke koje se traže. Pretraživač klijenta donosi odluku kako se to prikazuje. HTML nije programski jezik, tako da u njemu nema konstrukcija koje bi na bilo koji način obezbedile obradu podataka.

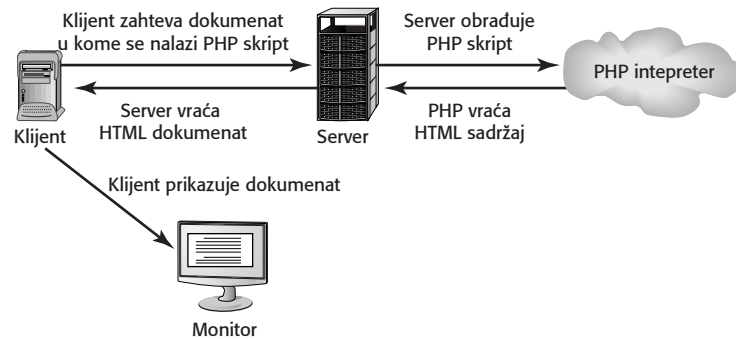
Za razliku od HTML-a, PHP je u potpunosti na strani servera. Kada se PHP skript izvršava, on nema direktnu saradnju sa Web pretraživačem, već se samo završni proizvod PHP skripta, a to je, obično, HTML dokumenat, šalje do Web pretraživača. Ako bi pretraživač primio neobrađeni PHP skript, on bi pokušao da ga prikaže kao običan HTML dokumenat. Pretraživači ne mogu da izvršavaju PHP skripte.

HTML je integralna komponenta u razvoju Web aplikacija. PHP kod se može ubaciti direktno u HTML. Kada klijent sa Web servera zatraži HTML dokumenat, server odgovara slanjem tog dokumenta. Na slici 1.1 smo pokazali kako klijent traži HTML dokumenat i kako server na to odgovara.



SLIKA 1.1 *Zahtev za HTML dokumentom*

Zahtev za PHP skriptom radi drugačije. Pre nego što se dokumenat pošalje do klijenta, obrađuje se pomoću PHP-a i PHP mašina izvršava PHP kod koji postoji u tom dokumentu. Na slici 1.2 je ilustrovan zahtev klijenta za PHP skriptom koji na ovoj ilustraciji vraća obrađeni HTML dokumenat.



SLIKA 1.2 Zahtev za PHP skriptom

Ova dva procesa prikazuju razlike između PHP-a i HTML-a. PHP se izvršava na strani servera i predstavlja potpuni programski jezik. HTML se samo koristi za publikovanje hiperteksta i on se obrađuje na strani klijenta.

Upoznavanje PHP-a

Kad god pravite aplikaciju u nekom programskom jeziku, vrlo je bitno da u potpunosti poznajete mogućnosti i ograničenja okruženja koje se koristi. U ovom odeljku ćemo pokazati kako se statički HTML koristi sa PHP-om. Ovo incijalno objašnjenje predstavlja osnove koje treba da poznaje svako ko čita ovu knjigu. Početak istraživanja razvoja za Web i okruženja koje postoji počecemo pisanjem prvog PHP skripta. Upoznaćete promenljive i osnove PHP-a i dinamičkog sadržaja.

NAPOMENA

Kompletna vodič za instalaciju i konfigurisanje PHP-a možete naći u dodatku A, "Kratak kurs iz PHP-a". ■

Vaš prvi PHP skript

Pre nego što se PHP skript izvrši, instrukcija da treba da se izvrši mora se uputiti serveru. Ovo ćete uraditi tako što ćete blok sa skriptom uokviriti specijalnim oznakama koje serveru saopštavaju da je reč o PHP kodu. Kada server naiđe na oznaku za otvaranje PHP-a, sve što se nalazi između te oznake i oznake za kraj, smatra se PHP-om. PHP oznaka za otvaranje je:

```
<?php
```

dok je oznaka za zatvaranje

```
?>
```

Alternativno, možete da koristite i oznake `<? i ?>`. Preporučuje se da koristite oznaku `<?php` za otvaranje, jer se time pravi kod koji je u skladu sa stilom iz XML-a. Generalno govoreći, prihvatljivi su i jedan i drugi metod.

Možete da koristite i druge metode za otvaranje i zatvaranje PHP skriptova, ali tada morate da tu opciju omogućite promenom datoteke `php.ini`. Opcija za konfigurisanje `asp_tags` omogućava da za otvaranje i zatvaranje koristite oznake skriptova kao u ASP-u (znači `<% i %>`). U datoteci `php.ini` pronađite red sa tekстом `asp_tags` i promenite od `asp_tags - Off` na `asp_tags = On`. Upotreba formata iz ASP skriptova uglavnom zavisi od ukusa.

NAPOMENA

Preporučujemo da se koristi stil `<?php i ?>`, tako da ćemo njega i koristiti u ovoj knjizi. ■

Ako ste zaista iskusni sa nekim drugim programskim jezikom, posebno sa skript jezikom koji liči na PHP, onda će narednih nekoliko odeljaka za Vas predstavljati samo podsećanje.

Hello, World!

Kakava bi to knjiga o proramiranju bila ako u njoj ne bi bilo programa "Hello, World!". Ovo je uvek prvi program koji se napiše u bilo kom programskom jeziku.

Mi ćemo napisati minimalni PHP skript koji generiše tekst "Hello World!" i šalje ga pretraživaču. Ovaj skript ilustruje oznake za otvaranje i zatvaranje, kao i PHP konstrukciju koja se koristi za generisanje izlaza.

Kreirajte dokumenat po imenu `hello_world.php` i u njega unesite kod iz listinga 1.1. Zapamtite taj dokumenat u korenu (root) fasciklu Vašeg servera i onda ga pronađite u svom pretraživaču.

- Od podrazumevane konfiguracije na Red Hatu korena fascikla je `/var/www/html`.
- Ako ste kao vodič za instalaciju PHP-a i Apache servera koristili dodatak A, korena fascikla za dokumente će biti `/usr/local/apache/htdocs`. Kompletan putanja zajedno sa datotekom bi onda bila `/usr/local/apache/htdocs/hello_world.php`.

SAVET

Da bi Vam bilo lakše da radite, svi skriptovi iz knjige nalaze se na propratnom CD-u. Ove datoteke možete da pogledate preko svog Web pretraživača kada hoćete videti rezultat, a možete i da ih otvorite u nekom programu za obradu teksta da biste videli kako izgleda kod. ■

Listing 1.1: Osnovna PHP Web strana (`hello_world.php`)

```
<html>
<head>
  <title>Hello, World!</title>
</head>
<body>
  <?php print('Hello, World!\n'); ?>
</body>
</html>
```

Možda ćete pogrešno zaključiti da je skript iz ovog listinga izuzetno jednostavan. Prilikom izvršavanja koda dešava se veoma malo. Primetićete opšte entitete iz HTML-a - `html`, `head`, `title` i oznake za telo dokumenta (`body`). U dokumentu, međutim, postoje i dva specijalna PHP entiteta: oznaka za otvaranje skripta `<?>` i oznaka za zatvaranje `?>`. Sve što se nalazi između ove dve oznake Web server šalje do PHP interpretera. PHP obrađuje skript i šalje kompletan izlaz do klijenta. Nakon PHP oznake za zatvaranje, dokumenat se šalje kao običan HTML dokumenat, sve dok se ne nađe na drugu PHP oznaku.

PHP skript "kaže" PHP interpreteru da do pretraživača pošalje tekst "Hello World!", zajedno sa novim redom. Funkcija počinje imenom; u ovom slučaju je to `print`. Nakon toga se za markiranje početka i kraja argumenata funkcije koriste zagrade - u ovom slučaju stringa "Hello, World!\n". Posle funkcije dolazi znak tačka-zarez, što znači da je rež o kraju iskaza. Ako se izostavi tačka-zarez, PHP javlja grešku.

String je samo niz karaktera. String koji se šalje kao argumenat funkcije je sve što se nalazi između jednostrukih navodnika. Tekst `\n` koji se nalazi u okviru stringa šalje se pretraživaču kao jedan karakter (specijalni karakter koji označava novi red), a ne kao dva. U pitanju je escape karakter, odnosno karakter koji predstavlja nešto neuobičajeno, a ne literal. U ovom slučaju ispred slova `n` je stavljena kosa crta. Ovakav karakter (`n` sa kosom crtom ispred njega) predstavlja ASCII karakter za novi red (13), a ne posebne literale kosa crta i `n`.

Ako želite da proverite da li ste pretraživaču zaista poslali oznaku za novi red, pogledajte dokumenat nakon unosa koda. U listingu 1.2 pokazano je kako izgleda kod nakon što ga je PHP obradio i poslao do klijenta.

Listing 1.2: Kod za `hello_world.php` nakon obrade pomoću PHP-a

```
<html>
<head>
  <title>Hello, World!</title>
</head>
<body>
  Hello, World!
</body>
</html>
```

Kada pogledamo ovaj kod, čini se da u njemu nema ničeg specijalnog. PHP oznake se obrađuju i ne šalju do klijenta. Šta se desilo sa `\n`? Šta bi se desilo ako bi se iz stringa bio izbačen karakter za novi red. U tom slučaju bi zadnjih nekoliko linija u dokumentu izgledao ovako:

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

Ovaj iskaz možemo da proverimo ako iz funkcije `print` obrišemo oznaku za novi red, tako da poziv ove funkcije izgleda ovako:

```
print('Hello, World');
```


Nakon što smo obrisali novi red, oznaka body se prikazuje odmah iza teksta Hello. World!.

Ko ste Vi?

Hajde da sada ispitamo malo komplikovaniji program. Napisaćemo dve PHP strane. Jedan skript sadrži HTML formu koja može da se pošalje, dok je drugi aktivna strana, odnosno to je skript kojem se forma šalje. Kada se forma pošalje do druge strane, PHP omogućava da se sa strane za akciju pristupi podacima iz forme. Prvi skript je nazvan who_are_you.php i to je obična HTML strana koja sadrži formu. U datoteku who_are_you.php unesite kod iz listinga 1.3 i zapamtite to na svom Web serveru.

Listing 1.3: Strana za unos osnovnih podataka (who_are_you.php)

```
<html>
<head>
    <title>Who Are You?</title>
</head>
<body>
<form action="you_are.php">
    Please enter your name:<br />
    I am...
    <?php print('<input type="text" name="person" value="' . $person .
                '"size="15" />');
    ?>
    <input type="submit" value="Go!" size="15" />
</form>
</body>
</html>
```

U ovom listingu postoji nekoliko bitnih detalja. Skript kreira običnu HTML formu, koja se šalje do skripta you_are.php (URL u atributu action za formu). Na formi postoje jedno polje za unos teksta i dugme Submit. Kada korisnik klikne ovo dugme, forma se šalje do strane za akciju.

PHP iskaz jednostavno šalje do izlaza čisti tekst koji postaje HTML polje za unos teksta. Iskaz print je ista funkcija kao i iz listinga 1.1, ali je rezultat listinga 1.3 polje na formi sa svojim atributima.

Vrednost za tekst koji se šalje kao izlaz nastaje spajanjem stringa i promenljivih. Na taj način, nastaje string literal. Operator za spajanje stringova u PHP-u je tačka (.). On interpreteru "govori" da uzme stavke koje su ispred i iza i da ih spoji u jedan zajednički string.

Notacija

```
... value=' ' . $person . ' '...
```

popunjava atribut value podacima iz promenljive \$person, koju smo napravili da bismo smestili podatke koje korisnik treba da unese. Prvo se argumenat funkcije print završava preko jednostrukog znaka navoda. Nakon toga se dodaje vrednost promenljive \$person, odnosno dodaje se na kraj stringa (preko prve tačke). Potom se dodaje i ostatak stringa (preko druge tačke), tako da završavamo sa kompletnim tekstom. Kada PHP od ovoga napravi formu i prebaci je u pretraživač klijenta, promenljiva \$person je prazna, pa se, prema tome, polje popunjava praznim stringom. Kada

korisnik unese tekst i pošalje formu nazad (preko dugmeta Submit), promenljiva `$person` uzima tekst koji je unesen. Ovo ćete videti u narednom odeljku.

Ako poznajete druge programske jezike, primetićete da promenljiva `$person` nije deklarisan, niti inicijalizovana. Jezici sa čvrstim tipovima podataka zahtevaju da promenljiva bude deklarisan i da se pre njene upotrebe definiše njen tip. Ovo nije slučaj sa PHP-om, kao što ćete videti u narednom poglavlju.

NAPOMENA

Sve promenljive u PHP-u moraju da počnu oznakom za dolar (\$). ■

Forme i stringovi sa upitima

Nakon što smo kreirali PHP skript koji generiše HTML formu, moramo da kreiramo i stranu za akciju kojoj se ova forma šalje. HTML forma se šalje do strane `you_are.php`. Napravite PHP skript pod imenom `you_are.php` i u njega ubacite kod iz listinga 1.4.

Listing 1.4: Osnovna strana za akciju (`you_are.php`)

```
<html>
<head>
    <title>You Are! ...</title>
</head>
<body>
<?php
    print('Well, hello ' . $person . ', nice to meet you!');
    print('<br />');
    print('<a href="who_are_you.php?person=' . urlencode($person) . '">
        Back to Who Are You?</a>');
?>
</body>
</html>
```

U ovom listingu se iskaz `print` koristi za generisanje tela HTML dokumenta. Operator za spajanje stringova (.) se koristi za sklapanje novog stringa. Prvi iskaz kao izlaz daje poruku koja pozdravlja korisnika onim što je uneto u polju `$person` na prethodnoj strani. Nakon slanja forme, strana za akciju omogućava da se poljima iz forme pristupi kao PHP promenljivim. Ipak, u listingu 1.4 postoje i neke osobenosti na koje treba obratiti pažnju.

U okviru trećeg `print` iskaza možete da primetite oznaku za vezu sa stranom `who_are_you.php`. Ako izaberete ovaj hiperlink, pretraživač će Vas vratiti nazad na početnu stranu. Susjedni deo iza mesta gde link vodi je polje za unos teksta, koje se popunjava vrednošću unetom na inicijalnoj formi. Kako bi običan hiperlink trebalo da popuni polje sa naše forme?

Odgovor je u URL adresi koja je vezana i koja u sebi sadrži neke dodatne informacije. Relativna URL adresa je `who_are_you.php`, iza koje slede znak pitanja, pa tekst `person=` i na kraju nešto novo što uključuje našu promenljivu `$person`. Ove stavke se spajaju na način koji smo opisali. Ceo hiperlink bi trebalo da izgleda ovako:

```
http://127.0.0.1/who_are_you.php?person=Jeremy+Allen
```

Kada se pošalje forma, pretraživač automatski kodira string upita. String upita je sve što se u URL adresi nalazi iza znaka pitanja (ovde je to `?person=Jeremy+Allen`). Kada Web strana za pretraživanje generiše dugačak URL u kome se nalazi i znak pitanja, onda se preko stringa upita zahtev šalje do mašine za pretraživanje. String upita je način za prosleđivanje parametara do Web strane. Pretraživač organizuje podatke iz forme u obliku parova ime/vrednost. Kada se forma koju smo koristili u datoteci `who_are_you.php` (listing 1.3) pošalje, kreira se jedan par ime/vrednost i dodaje URL adresi koja je zadata atributom action forme. Podaci koji se dodaju počinju znakom pitanja, iza kojeg slede ime, znak jednakosti i, na kraju, vrednost za to ime. Svi naredni parovi ime/vrednost se odvajaju preko znaka `&`; na primer `?person=Jeremy+Allen&role=author`.

Kada PHP skript primi kodirane URL parove ime/vrednost, on automatski vrši dešifrovanje (to radi PHP interpreter). Sintaksa URL-a se definiše pomoću nekoliko specijalnih karaktera, koji se moraju kodirati da bi mogli da se predstavljaju literalima za vrednosti tipa karakter u okviru stringa upita. Na primer, znak plus ne može da upotrebi se u URL-u, jer on u stringu upita predstavlja prazninu. Ponekad će se, ipak, desiti da je nekome potrebno da u okviru stringa upita kao deo vrednosti postoji i znak plus.

Da bi se izbegla zabuna sa ovim literalima, mi smo promenljivu `&person` uokvirili još jednom PHP funkcijom, a to je funkcija `urlencode()`. Ovo smo uradili zato što, kad god se string upita dinamički kreira za hiperlink, svi specijalni karakteri moraju biti URL kodirani da bi parovi ime/vrednost mogli da se prebace kako treba.

Slanje podataka: dva metoda

U prvom skriptu koji smo do sada pravili mi smo slali podatke sa Web strane na server i obratno. Ovo je malo podataka, ali su to, ipak, podaci. Sa druge strane, mi i nismo definisali tehniku koju smo koristili. Kada se forma pošalje i kada se parovi ime/vrednost pojave u URL-u, kao u prethodnom odeljku, podaci se prosleđuju pomoću metoda GET. Alternativni način za slanje podataka je metod POST. Hajde da pogledamo nekoliko strana da bismo objasnili primenu svake od ovih.

SAVET

U verziji PHP 4.1 postoje izvesne promene u načinu na koji se upravlja GET i POST zahtevima. Ovde smo dali opšti uvod, a detalje možete naći u Poglavlju 9. ■

UPOZORENJE

Metod GET jednostavno dodaje podatke URL-u koji se šalje do Web servera. Dužina URL-a je ograničena, što znači da on nije dobar "kandidat" za slanje velike količine podataka. Sa druge strane, bilo koju formu koja se šalje pomoću metoda GET može da označi korisnik, pošto se svi podaci forme dodaju URL-u kao string upita. ■

PODEŠAVANJE URL-A

Dužinu URL-a ograničava klijent. Netscape ne nameće nikakva ograničenja u pogledu dužine URL-a, ali IE (Internet Explorer) dozvoljava najviše 2.048 karaktera u delu putanje i najviše 2.083 karaktera u kompletnom URL-u, uključujući i string upita. Macromedia Generator nameće ograničenje od beznačajnih 255 karaktera za dužinu URL-a.

Kada kreirate aplikaciju, morate koristiti najmanji zajednički sadržalac. Ako nameravate da se aplikacija koristi i u Netscapeu i u IE-u, morate da računate sa ograničenjem od 2.048 karaktera. Ovo je samo jedna od okolnosti koje morate da razmotrite kada kreirate aplikaciju.

Metod slanja možete da eksplicitno definišete u kodu. Ovo se radi pomoću atributa `method`. Na primer, u datoteci `who_are_you.php` u listingu 1.3 mi bismo mogli da promenimo oznaku `form` od

```
<form action="you_are.php">
```

na

```
<form action="you_are.php" method = "GET">
```

Sada je u okviru forme definisan način slanja podataka, tako da nema sumnje koji će metod pretraživač da koristi za slanje podataka.

Sledeće što moramo da razmotrimo jesu karakteri koji su rezervisani i koji ne mogu biti deo URL-a, ako se ne kodiraju. U tabeli 1.1 je data lista karaktera koji se najčešće koriste i njihovih kodiranih vrednosti u URL-u. Kada string upita bude sklapan i kada želite da se ubaci i neki literal, morate da tu vrednost kodirate prema kombinaciji koju smo dali u desnoj koloni. PHP funkcija `urlencode()` obavlja ovaj posao, umesto Vas. Obično, URL ne mora da se eksplicitno dekodira, pošto time upravlja pretraživač.

Tabela 1.1: Najčešći karakteri u URL-u i njihove kodirane vrednosti

Karakter	URL kod
Belina (razmak)	%20
"	%22
#	%23
%	%25
&	%26
(%28
)	%29
+	%2B
'	%2C
/	%2F
:	%3A
;	%3B
<	%3C

=	%3D
>	%3E
?	%3F
@	%40
\	%5C
	%7C

Ako ste napravili skript prema listingu 1.4, ovo možete i sami da probate. Usmerite pretraživač na stranu `you_are.php`, zajedno sa putanjom i fasciklom u koju ste ga postavili, i dodajte string upita:

```
http://x.x.x/path/you_are.php?person=Just+Me+%3A%29
```

Trebalo bi da vidite prijateljski pozdrav nakon unosa URL-a koji je adresovan na Just Me :). Unutar vrednosti koju ima par ime/vrednost znak plus predstavlja prazninu i smatra se specijalnim karakterom koji čini deo URL sintakse. Umesto znaka plus, može da se koristi i URL kod `%20`.

NAPOMENA

URL nije ispravan ako nepravilni karakteri nisu kodirani kako treba. Najlakše je da koristite ugrađene PHP funkcije koje će kodirati URL, ali kodiranje URL-a može da bude i problematično. U Poglavlju, "Forme i interakcija sa korisnikom", možete naći više detalja o kodiranju URL-a. ■

Metod GET je relativno jednostavan. U listingu 1.5 se kreira forma koja generiše dovoljno podataka da se vidi kako ovaj metod prosleđuje podatke preko stringa upita. Ispitajte kod u listingu 1.5 i upamtite ga kao `who_are_you_advanced.php`. U ovom kodu smo dali i nekoliko osobina iz PHP-a, a to su komentari i cela PHP oznaka, ugnježdjena između vrednosti atributa.

Listing 1.5: Strana sa naprednim unosom podataka (`who_are_you_advanced.php`)

```
<html>
<head>
    <title>Who Are You Advanced!</title>
</head>
<body>
<form action="you_are_advanced.php" method="GET">
    Please fill in the following fields:<br />
    <?php
        // Assign a value to favorite_language variable
        $favorite_language = "PHP";
    ?>
    <b>First Name:</b>
    <input type="text" name="first"
        value="<?php print($first); ?>" size="15"><br />
    <b>Last Name:</b>
    <input type="text" name="last"
        value="<?php print($last); ?>" size="15"><br />
```

```
<b>Favorite Programming Language:</b>
<input type="text" name="favorite_language"
      value="<?php print($favorite_language); ?>" size="15"><br />
<input type="submit" value="Go!" size="15">
</form>
</body>
</html>
```

U prvom PHP bloku iz ovog listinga nalaze se komentar i dodeljivanje vrednosti promenljivoj. Komentar se koristi za objašnjenje detalja koda, ili za postavljanje neke napomene za budućnost. PHP interpreter u potpunosti zanemaruje komentar, tako da se njemu može naći bilo šta, uključujući i PHP kod. Ovaj komentar je označen preko dve kose crte (postoje različiti stilovi za definisanje komentara, koje ćete upoznati kasnije u ovom poglavlju).

Nakon toga se promenljivoj `$favorite_language` dodeljuje vrednost "PHP". Kada se jednom vrednost sačuva u okviru promenljive, ta promenljiva se može koristiti kao zamena za tu vrednost. Kasnije u kodu mi smo funkciji `print` dali samo tu promenljivu kao argumenat (kao tri atributa). Kao argumente funkcije nismo koristili eksplicitne stringove, ali, pošto su promenljive stringovi, funkcija `print` i dalje štampa ono što treba.

Naša strana za napredni unos podataka treba da ima i akcionu stranu, kojoj ćemo te podatke poslati. Kreirajte datoteku `you_are_advanced.php` na osnovu koda iz listinga 1.6 i zapamtite je. Ako se desi neka greška, proverite da li je kod unesen potpuno isto kao u listingu, ali upamtite da uvek možete da otvorite kopiju datoteke koja se nalazi na CD-u koji dobijate sa ovom knjigom.

Listing 1.6: Napredna strana sa akcijom (you_are_advanced.php)

```
<html>
<head>
  <title>You Are Advanced! ...</title>
</head>
<body>
Hello <b><?php print($first . " " . $last); ?></b>
I am glad to know your favorite programming language
is <?php print($favorite_language); ?>.
<?php
  $query_string = "";
  $query_string .= "?first=" . urlencode($first);
  $query_string .= "&last=" . urlencode($last);
  $query_string .= "&favorite_language=" . urlencode($favorite_language);
?>
<br /><br />
<a href="who_are_you_advanced.php<?php print($query_string) ?>">
  Back To Who Are You Advanced
</a>
</body>
```

```
</html>
```

Prva dva bloka PHP koda jednostavno umeću podatke koje je korisnik uneo u tekst te strane. Naredni blok, onaj koji ima sopstvene redove, radi nešto interesantnije. Prvo se kreira promenljiva `$query_string`, zatim joj se dodeljuje vrednost "", što jednostavno znači da je reč o praznom stringu. Nakon toga smo sklopili string upita, odnosno dodali smo jedan po jedan par ime/vrednost.

Operator `.` je skraćenica za spajanje sa prethodnim sadržajem. Naredne dve linije su ekvivalentne:

```
$query_string .= "?first=" . urlencode($first);  
$query_string = $query_string . "?first=" . urlencode($first);
```

Drugi tip notacije, sa ponavljanjem imena promenljive, može biti malo glomazan, tako da su PHP programeri implementirali nekoliko operatora koji omogućavaju skraćenu notaciju nekih opštih operacija, kao što su sabiranje, oduzimanje, ili množenje. Kao što vidite `.` znači "zalepite sadržaj sa desne strane znaka jednakosti, na kraj postojećeg sadržaja promenljive". Ovo je standardan način za kreiranje stringa upita, a da se ne prave previše dugačke linije.

SAVET

PHP operatori i sintaksa za njihovu upotrebu su detaljnije objašnjeni u Poglavlju 3, "Operatori i izrazi". ■

Sklapanje stringa upita i zapisivanje njegovog sadržaja u promenljivu pružaju jasan izlaz iz URL-a i omogućavaju da se PHP kod mnogo lakše čita i razume. Svaka promenljiva je URL kodirana, tako da se i nepravilni karakteri koje bi korisnik mogao da unese interpretiraju na pravi način, što znači i da je URL adresa ispravna.

Svaki put kada nas hiperlink navede na stranu `who_are_you_advanced.php` jezik (favorite language) se menja vrednošću PHP. Korisnik može da ovo preklopi, tako što će na strani da unese nešto drugo, ali je ovo jedan od načina da se obezbedi podrazumevana vrednost za neki podatak sa forme.

Metod POST

Metod POST i slanje podataka sa forme preko njega rade sasvim drugačije od metoda GET. Najvažniji aspekt metoda POST je da se podaci koji se šalju ne kodiraju u okviru URL-a, već se šalju kao deo HTTP zaglavlja, što je za većinu korisnika nevidljivo. Korisnici ne mogu da vide te podatke i podaci koji se na ovaj način unesu nisu keširani u pretraživaču. Ovo može biti dobro, ili loše, u zavisnosti od toga šta želite. Drugi bitan aspekt metoda POST je da string upita nije ograničen po dužini. Svi ti faktori dovode do toga da je ovaj metod prilično koristan.

Otvorite datoteku `who_are_you_advanced.php` (listing 1.5) u nekom programu za obradu teksta i pronađite liniju koja definiše formu:

```
<form action="you_are_advanced.php" method="GET">
```

Promenite ovu liniju na

```
<form action="you_are_advanced.php" method="POST">
```

Nakon što promenite i upamtite ovaj red, možete da posmatrate ponašanje metoda POST. Otvorite datoteku u pretraživaču i u polje za unos teksta, zajedno sa omiljenim programskim jezikom, unesite ime i prezime. Kada kliknete submit, pogledajte URL. U njemu nema parova ime/vrednost, ali na akcionoj strani i dalje primete svoje promenljive. Rad pomoću metoda POST je prilično jednostavan i omogućava prebacivanje datoteka preko izvesnih elemenata forme.

Izbor između metoda GET i POST

Prilikom odlučivanja o načinu transfera podataka najvažnije je da imate na umu:

- količinu podataka koji se šalju (ovo se definiše preko dužine stringa upita za metod GET i veličine HTTP zaglavlja za metod POST)
- vidljivost podataka

Metod GET ima ograničenu dužinu putanje i stringa upita. Metod POST može da prosledi neograničenu količinu podataka i parova ime/vrednost, ali je URL i dalje ograničen kod nekih klijenata.

NAPOMENA

Internet Explorer ograničava putanju u URL-u i za POST i za GET metod na dužinu od 2.048 karaktera. IE je jedan od najpopularnijih pretraživača, tako da bi prilikom programiranja aplikacija trebalo da tu činjenicu uvek imate na umu. ■

Metod GT može da izazove i neke neželjene efekte. Pogledajte stranu koja iz polja na formi traži neke osetljive podatke. Ako se polje iz forme šalje pomoću metoda GET, osetljivi podaci se dodaju u URL. Ako neko pogleda podatke u URL-u, ili, verovatnije i manje očigledno, ako pogleda istoriju URL adresa koje je pretraživač posetio, mogao bi da tom korisniku donese "glavobolje".

Kada se šalju osetljivi podaci, ili kada se šalje puno podataka, najbolje je koristiti metod POST. Metod GET dolazi u obzir samo kod malih količina podataka čija sigurnost nije bitna. Sa druge strane, čak i slanje podataka pomoću metoda POST, ali preko kanala za komunikaciju koji nije siguran, kao što je tipična HTTP veza, koja nije šifrovana pomoću SSL-a, nije sigurno.

SAVET

Detaljniju obradu metoda GET i POST i načina kako da vidite podatke koji se šalju pomoću metoda POST možete naći u Poglavlju 9. ■

Komentari u kodu

Komentari su način da u kodu ostavite male napomene. Koriste se u svim programskim jezicima; čak i HTML ima mogućnost postavljanja komentara. U stvarnosti ništa nije tako jednostavno kao što izgleda. Iako objašnjavanje razloga zašto se kod komentariše i kako se to radi može da duže potraje, ipak ćemo u ovom odeljku pokušati da ukažemo na neke pravce koji se koriste za komentare.

Jedno mišljenje o komentarima je da oni jednostavno ne bi ni trebalo da postoje. Kod bi trebalo da bude tako dobro napisan da sam sebe dokumentuje, tako da su komentari izlišni. Ovakvim razmišljanjima ukazuje se da komentari samo ohrabruju loše programiranje i vode ka kodu koji ne može da se čita i lako razume. Komentari mogu da u kod dodaju složenost i, ako se pogrešno napišu, mogu, čak, i da dovedu u zabunu.

Ovakva linija je, međutim, ekstremna i nije baš mnogo zastupljena. Iako zasnovana na "klimavoj" logici, ipak tu ima i neke istine. Jedna istina je da loš i pogrešan komentar može da bude mnogo gori nego da uopšte nema komentara. Sa druge strane, dobri komentari su od neprocenjive vrednosti kasnije, kada taj kod treba održavati, adaptirati, ili ponovo koristiti. Komentari mogu, takođe, biti sasvim pogodni za privremeno "sakrivanje" delova koda, za koje ne želimo da rade dok smo u fazi razvoja, ili testiranja.

Vrlo je važno da imate konzistentan stil pisanja komentara. PHP podržava tri stila za pisanje komentara:

```
# Ovo je komentar, kao komentar u skriptu iz ljuske
// Ovo je komentar, kao u C++-u, ili Javi
/* Ovo je komentar koji dolazi iz C-a */
```

Stilovi komentara `//`, ili `#` sakrivaju samo jednu liniju, dok stil iz C-a `/* */` mogu da obuhvataju više linija. Različiti programeri vole različite stilove. Ovako veliki broj mogućnosti može da dovede u zabunu, ali je jedna od najboljih osobina PHP-a da on u sebi spaja osobine nekoliko drugih jezika. Ovakvi stilovi za pisanje komentara su korisni pri prilagođavanju programera koji poznaju neki drugi programski jezik. Na kraju da kažemo da treba da koristite onaj način koji se Vama dopada, ali to koristite konzistentno.

Evo jednog primera stila za komentare koji je teško održavati (za isticanje karaktera koji definišu komentare koristili smo masna slova):

```

/*****\
* world_dominator.php                      *
* Author - Jeremy Allen                     *
*                                           *
* Routines that describe how to take over the world using PHP. *
* Take extreme caution when using these routines.             *
*                                           *
\*****/

```

PHP interpreter prepoznaje sekvencu karaktera `/* i */`. Sve što se nalazi između ta dva karaktera biće sakriveno od interpretera. Iako ovakav stil za komentare može izgledati vrlo koristan, on postaje "noćna mora" kada ga treba održavati. Šta učiniti ako u ovaj komentar želite da dodate nekoliko reči, ili da promenite ime autora? Ovo može biti prilično teško, pošto završni karakter na kraju svake linije mora biti propisno napisan, sa razmakom, ili tab karakterom, tako da se sve ovo poravnava kako treba. U praksi ovakav stil može da ometa rad i nema neku realnu namenu.

Razmotrite sledeći stil za pisanje komentara:

```
/*
    world_dominator.php
```

```
Author - Jeremy Allen
Routines that describe how to take over the world using PHP.
Take extreme caution when using these routines.
*/
```

Reč je o stilu koji nije tako veličanstven kao u prvom primeru, ali se ovaj komentar mnogo lakše menja i održava. Kada održavate kod čiji je "životni vek" više od nekoliko dana, ovo morate imati na umu. Svi ovi sitni aspekti programiranja koje upotrebite u svojim programima mogu kasnije da dovedu velike uštede u vremenu.

U listingu 1.7 smo pokazali kako izgledaju dobar stil za pisanje komentara i efikasan stil za pisanje komentara. Naravno, u stvarnom životu lakše bi bilo da koristite komentare iz HTML-a `<!-- -->`. Ovde nam je bila namera da pokažemo kako rade komentari iz PHP-a i da oni mogu da postoje zajedno sa komentarima iz HTML-a.

Listing 1.7: Komentari u kodu, ili "kratke napomene" za programe

```
<?php
/*
    PHP comment
    File:      who_are_you_advanced.php
    Author:    Jeremy Allen
    Description: Who Are You Advanced contains multiple form
                fields that require personal data from a user.
*/
?>
<html>
<head>
    <title>Who Are You Advanced!</title>
</head>
<body>
<form action="you_are_advanced.php" method="POST">
    Please fill in the following fields:<br />
    <?php
        // PHP comment: Strong-arm user into believing PHP is the best
        $favorite_language = "PHP";
    ?>
<!-- HTML comment: Use three input boxes to get user data -->
    <b>First Name:</b>
    <input type="text" name="first"
        value="<?php print($first); ?>" size="15"><br />
    <b>Last Name:</b>
    <input type="text" name="last"
        value="<?php print($last); ?>" size="15"><br />
    <b>Favorite Programming Language:</b>
    <input type="text" name="favorite_language"
        value="<?php print($favorite_language); ?>" size="15"><br />
```

```
<input type="submit" value="Go!" size="15">
</form>
</body>
</html>
```

UPOZORENJE

Komentari u kodu su "dobra stvar". Ipak, generalno govoreći, komentari su za programere, tako da do korisnika ne treba da ih šalžete. Mogli biste da koristite HTML komentare prilikom debugovanja, ali nemojte ih nikada slati do klijenta. Uvek je dobro da se programska logika sakrije od korisnika, ako ne želite da delite svoj kod sa ostatkom sveta. ■

Možete li da otkrijete suvišan komentar u prethodnom kodu? Reč je o napomeni "strong-arm". Komentar koji jednostavno ponavlja šta radi jedna linija koda beskoristan je i u suštini čini kod nejasnim. Pogledajte sledeći komentar:

```
//Do klijenta salje tekst "Hello, World"
print('Hello, World');
```

Ovaj komentar je totalno beskoristan. Sada pogledajte sledeći blok koda.

```
<?php
    // Assemble query string, encoding and gluing the form data together
    $query_string = "";
    $query_string .= "?first=" . urlencode($first);
    $query_string .= "&last=" . urlencode($last);
    $query_string .= "&favorite_language=" . urlencode($favorite_language);
?>
```

Ovaj komentar radi nešto više nego da samo ponovi ono što se već vidi iz koda. Kod je dobro grupisan, pa nema sumnje da sav kod obavlja jedan isti zadatak, tako da se to lako čita i razume. Promenljive koje se koriste imaju dobra imena, što može dalje da pomogne u razumevanju koda. Pored svega ovog, komentar jasno sumira šta se dešava u narednih nekoliko linija koda. Ako ovaj kod čita neko ko nije "familijaran" sa operatorom `.=`, komentar će mu dati generalnu ideju o tome šta se tu dešava. Ovo je vrlo korisno.

Ovde treba da imate meru. Nemojte sad odmah da izbrišete sve komentare koji se odnose na jednu liniju koda. Na kraju krajeva, skoro svaki komentar se lakše razume nego sam kod, sa izuzetkom "smešnih stvari", kao što je

```
$i++; //brojac $i se povecava za jedan
```

Osim u izuzetnim situacijama, za programera je uvek lakše da čita tekst u nekom prirodnom jeziku, nego da čita izvorni kod. Još važnije: komentari privlače pažnju programera, što pomaže prilikom otkrivanja bagova u kodu. Sam kod to ne može da uradi (iz samog koda ne možete reći šta je programer očekivao da on radi). Komentari su sjajni i kada idete kroz datoteke da biste pronašli neki deo koda. Ne morate da čitate ceo blok, ili da sklapate neki izraz da biste saznali šta je predviđeno da se uradi. Tu je komentar koji to objašnjava.

Sada sledi nekoliko završnih napomena o komentarima. Nemojte preterivati i praviti nepotrebne komentare. Da parafraziramo Kernighana i Plaugera (autore knjige "Elementi stila programiranja): "Nemojte da dokumentujete loš kod, već ga ponovo napišite". Ovo je zaista tačno. Kada deo koda izgleda neobično zbijeno i težak za razumevanje, dodavanje komentara možda i neće "raščistiti stvari". To je kao da u kod koji se teško razume dodajete nove poteškoće. Razmislite o tome da kod ponovo napišete. Vreme koje se provede u tome da se dobije kod koji je čitljiv i koji se lako održava kasnije će se isplatiti.

Ako morate da narušite dobar programerski stil, idite dalje i kratko, tačno i koncizno objasnite šta ste to uradili. Svet nije crno-beo. Nadamo se da će Vam ove napomene, kao i stil za komentare koji smo koristili u ovoj knjizi, poslužiti da shvatite koliko komentari mogu biti efikasni.