CGI დაპროგრამება (ლექციების კურსი)

ლექცია 1

CGI-ით (**Common Gateway Interface Class**) დაპროგრამების ორი სტილი არსებობს: ობიექტებზე ორიენტირებული და ფუნქციებზე ორიენტირებული სტილი. ობიექტებზე ორიენტირებული სტილის შემთხვევაში თქვენ ქმნით ერთ ან მეტ ობიექტს და იყენებთ ობიექტების მეთოდებს, რომ შექმნათ ინტერნეტ გვერდის სხვადასხვა ელემენტები. ყოველი ობიექტი იღებს სტარტს სახელდებული პარამეტრების მქონე სიით, რომელსაც გადასცემს სერვერი თქვენს **CGI** სკრიპტს. თქვენ შეგიძლიათ შეასწოროთ ობიექტები, შეინახოთ ისინი ფაილში ან მონაცემთა ბაზაში და ხელმეორედ შექმნათ ისინი. რადგან ყოველი ობიექტი შეესაბამება **CGI** სკრიპტის მდგომარეობას და რადგანაც ყოველი ობიექტის პარამეტრების სია დამოკიდებულია სხვებისაგან, ეს საშუალებას იძლევა შევინახოთ სკრიპტის მდგომარეობა და შემდეგ მოგვიანებით აღვადგინოთ იგი. მაგალითად, ობიექტებზე ორიენტირებული სტილით შევქმნათ მარტივი **“Hello World” HTML** გვერდი:

```
#!/usr/local/bin/perl
use CGI;      # ჩავტვირთოთ CGI პროგრამები
$q=new CGI;      # შევქმნათ ახალი CGI ობიექტი
print $q - >header,      # შევქმნათ HTTP სათაური
   $q -> start_HTML (' hello world ' ),      # ვიწყებთ HTML-ს
   $q -> h1(' hello world ' ),      # პირველი დონის სათაური
   $q -> end_HTML;       # HTML-ის დამთავრება
```

ფუნქციებზე ორიენტირებული სტილის შემთხვევაში, არსებობს ერთი **CGI** ობიექტი, რომელსაც თქვენ იშვიათად უკავშირდებით პირდაპირ. ამის ნაცვლად, თქვენ იძახებთ ფუნქციებს, რომ მიიღოთ **CGI** პარამეტრები, შექმნათ **HTML** ჭდეები, მართოთ პროცესის ინდიკატორი და ასე შემდეგ. ეს მოგამარაგებთ თქვენ წმინდა დაპროგრამების ინტერფეისით, მაგრამ გზღუდავთ გამოიყენოთ ერთზე მეტი **CGI** ობიექტი ერთდროულად. შემდეგი მაგალითი ბეჭდავს იგივე გვერდს რასაც წინა მაგალითი, მაგრამ იყენებს ფუნქციებზე ორიენტირებულ ინტერფეისს. მთავარი განსხვავება მდგომარეობს იმაში, რომ თქვენ ახლა გჭირდებათ გარკვეული რაოდენობის ფუნქციების შემოტანა სახელთა არეში (ესენი არიან ჩვეულებრივ “**standard**” ფუნქციები) და არ გჭირდებათ შექმნათ **CGI** ობიექტი.

```
#/user/local/bin/perl
 use CGI qw/:standard/;  # ჩავტვირთოთ CGI ფუნქციები.
  print header,       # შევქმნათ HTTP სათაური
  start_HTML(' hello world ' ),    #  HTML-ს
  h1(' hello world ' ),     # პირველი დონის სათაური
```

end_HTML; # HTM-ის დამთავრება

ამ დოკუმენტში მოყვანილი მაგალითები უმთავრესად იყენებენ ობიექტებზე ორიენტირებულ სტილს. იხილეთ **HOW TO IMPORT FUNCTIONS** იმ ინფორმაციის მისაღებად თუ როგორ გამოიყენოთ ფუნქციებზე ორიენტირებული დაპროგრამება **CGI.PM**-ში

ლექცია 2

CGI.PM პროგრამების გამოძახება

CGI.PM პროგრამების უმეტესობას შეიძლება ჰქონდეთ სხვადასხვა არგუმენტები, ზოგჯერ 20-მდე არასავალდებულო არგუმენტი. რომ გავამარტივოთ მათი გამოყენება, ყველა პროგრამა იყენებს არგუმენტების სახელით გამოძახების სტილს, რომელიც გამოიყურება ასე:

```
print $q -> header (- type => ' image/gif ' , - expires=>' 3d ' );
```

ყოველი არგუმენტის სახელს წინ ეწერება (დეფისი). არგუმენტის სახელში დიდი და პატარა ასოების გამოყენებას არა აქვს მნიშვნელობა. მაგალითად, -tipe, -Tipe და -TYPE ერთიდაიგივეა. არც არგუმენტის რიგს აქვს მნიშვნელობა არგუმენტების სიაში. სინამდვილეში, მხოლოდ პირველ არგუმენტს უნდა ჰქონდეს "-" არგუმენტების სიაში. თუ პირველ არგუმენტს აქვს "-" , მაშინ **CGI.PM** გულისხმობს "-" ყველა დანარჩენი არგუმენტებისთვისაც.

თქვენ არ უნდა გამოიყენოთ ქვედა ხაზი თუ თქვენ არ გსურთ იგი. **CGI** ობიექტის შექმნის შემდეგ, გამოიძახეთ **use_named_parameters( )** მეთოდი რაიმე არანულოვანი მნიშვნელობით. ეს მიანიშნებს **CGI.PM**-ს, რომ თქვენ გსურთ გამოიყენოთ მხოლოდ სახელიანი პარამეტრები:

```
$query = new CGI;
$query -> use_named_parameters(1);
$field = $query ->radio_group(' name ' =>' OS ' , ' value ' =>[' Unix ' , ' Windows ' , ' Macintosh ' ], ' default ' =>' Unix ' );
```

ზოგიერთ პროგრამებს საზოგადოდ ვიძახებთ მხოლოდ ერთი არგუმენტით. ასეთ შემთხვევაში თქვენ შეგიძლიათ იგულისხმოთ იგი სახელის გარეშე. **header ( )** არის ერთ-ერთი ასეთი პროგრამა. ამ შემთხვევაში არგუმენტი ტიპი არის **(type);**

```
print $q-> header (' text/HTML ' );
```

სხვა ასეთი პროგრამები მოცემილია ქვემოთ. ზოგჯერ სახელიანი არგუმენტები მოითხოვენ სკალარს, ზოგჯერ მასივზე მითითებას და ზოგჯერ **hash**-ზე მითითებას. ზოგჯერ თქვენ შეგიძლიათ გადასცეთ არგუმენტის რაიმე ტიპი და პროგრამა აიღებს იმას, რაც ყველაზე უფრო მოსალოდნელია. მაგალითად, **param ( )** პროგრამა გამოიყენება, რომ დავაყენოთ **CGI** პარამეტრი ერთ ან მრავალმნიშვნელობიან მნიშვნელობაზე. ასეთი ორი შემთხვევა ნაჩვენებია ქვემოთ:

```
$q -> param(~name =>' veggie ' , -value = > [' tomato ' ,' tomatho ' , ' potato ' ,' potahto ' ];
```

CGI.PM-ში სინამდვილეში პროგრამების დიდი რაოდენობა არ არის ისე როგორც საჭიროა. ესენი არიან „**HTML shortcuts**” პროგრამები, რომლებიც ქმნიან ჭდეებს (**tags**) დინამიურად შექმნილ გვერდებზე გამოყენებისათვის. **HTML** ჭდეებს აქვთ როგორც ატრიბუტები (ატრიბუტი = მნიშვნელობა წყვილი ჭდის შიგნით) ასევე

შინაარსები (**contents**) (ნაწილი, რომელიც მოთავსებულია ჭდის დაწყება და დამთავრება წყვილს შორის). რომ განვასხვაოთ ატრიბუტები და შინაარსები, **CGI.PM** იყენებს შეთანხმებას, რომლის თანახმად პირველ არგუმენტად უნდა გადასცეთ **HTML** ატრიბუტები, როგორც **hash**- ზე მითითებები და შინაარსები; თუ ისინი არსებობენ, როგორც მომდევნო არგუმენტები. ეს გამოიყურება შემდეგნაირად:

კოდი	გენერირებული **HTML**
h1 ( )	< H1 >
h1 (' some ' ,' ciontents ');	< H1 > some contents < / H1 >
h1 ({-align =>left});	< H1 ALIGN = “LEFT”>
h1 ({-align =>left}, ' contents ');	< H1 ALIGN = “LEFT”>contents < / HIZ

HTML ჭდეები უფრო დაწვრილებით აღწერილია ქვემოთ. **CGI.PM**-ში მუშაობის დამწყებთათვის თავსატეხ ამოცანას წარმოადგენს, თუ როგორ განასხვავონ გამოძახების შეთანხმებები, რომლებიც არსებობენ **HTML shortcates**-სა, რომლებიც მოითხოვენ ფიგურულ ფრჩხილებში ჩასმას. **HTML** ჭდის ატრიბუტში და გამოძახების შეთანხმებები სხვა პროგრამაში, რომლებიც ქმნიან ატრიბუტებს ფიგურული ფრჩხილების გარეშე. არ აგერიოთ. მოხერხებულობის მიზნით ფიგურული ფრჩხილები გამოიყენებიან ყველგან სურვილისამებრ, **HTML shortcates**-ის გარდა. თუ თქვენ მოგწონთ, თქვენ შეგიძლით გამოიყენოთ ფიგურული ფრჩხილები, როცა აიღებთ სახელიან არგუმენტებს პროგრამის გამოძახებისას. მაგალითად:

```
print $q->header({-type=>' image/gif ' , -espires=> ' +3d ' });
```

თუ თქვენ იყენებთ **–w** გადამრთველს (**Switch**), ფრთხილად იყავით, რადგან ზოგიერთი **CGI.PM** არგუმენტების სახელები კონფლიქტში მოდიან **Perl**-ის ჩაშენებულ (**built~in**) ფუნქციებთან. ყველაზე ხშირად ეს ხდება **–values** არგუმენტის დროს, როცა გამოყენებულია მრავალმნიშვნელობიან მენიუში, **radio button** კლასტერებში და მის მსგავსებში. რომ ავიცილოთ ეს შემთხვევა, თქვენ გაქვთ რამოდენიმე არჩევანი:

1. გამოიყენოთ ამ არგუმენტისათვის სხვა სახელი **–values**-თვის.
2. გამოიყენოთ მთავრული ასოები. მაგალითად, **-Values**.
3. არგუმენტის სახელი ჩასვით ბრჭყალებში. მაგალითად, ' **–values** ' .

მრავალ პროგრამებს შეუძლიათ გააკეთონ რაიმე სასარგებლო საქმე სახელიანი არგუმენტებით, რაც არაა საყოველთაოდ ცნობილი. მაგალითად, თქვენ შეგეძლოთ შეგექმნათ არასტანდარტული **HTTP** სათაურის ველები სახელიანი არგუმენტების გამოყენებით:

```
print $q -> header (-type =>' text/HTML ' ,
                    - cost=> ' Therse smaskers ' ,
                    - annoyance_leisel => ' high ' ,
                    -complians_to => ' bit bucket ' );
```

ეს მოგვცემს შემდეგ არასტანდარტულ **HTTP** სათაურს:

```
HTTP/1.0 200 Ok
Cost: Three smaskers
Annoyance_level: high
Complaints_to: bit bucket
Contant_type: text/HTML
```

შევნიშნოთ, რომ ქვედა ხაზები გადაითარგმნენ ავტომატურად დეფისებად. **HTML**-ის შემქმნელი პროგრამები ასრულებენ სხვადასხვა ტიპის თარგმანებს. ეს თვისება საშუალებას გვაძლევს გავითვალისწინოთ სწრაფად ცვლადი **HTTP** და **HTML** **“standards”**-ები.

ლექცია 3

ახალი ობიექტების შექმნა

(ობიექტებზე ორიენტირებული სტილი);

```
$query = new CGI;
```

მოხდება შესასვლელის გარჩევა (**POST** და **GET**) მეთოდებიდან) და შეინახება იგი **perl5** ობიექტში სახელით **$quary**.

ახალი ობიექტ-კითხვარის შექმნა შესასვლელი (INPUT) ფაილიდან

```
$quary = new  CGI (INPUTFILE);
```

თუ **new ( )** მეთოდს არგუმენტად მივცემთ ფაილის სახელს (**handle**), მაშინ ეს მეთოდი წაიკითხავს პარამეტრებს ამ ფაილიდან (ან **STDIN**, ან რაიმედან). ეს ფაილი შეიძლება იყოს ერთ-ერთი ფორმით, რომელიც აღწერილია ქვემოთ **debug**-ში. (ახალ სტრიქონზე გადასვლები **nl** შემოსაზღვრული **TAG = Value** წყვილებით იმუშავებენ). ფაილის ეს ტიპი იქმნება **save ( )** მეთოდით (იხილეთ ქვემოთ). მრავალი ჩანაწერები შეიძლება იყოს შენახული და აღდგენილი.

ეს სინტაქსი უშვებს ფაილის სახელზე მითითებას, რომელიც არის ოფიციალური ხერხი ფაილის სახელის გადასაცემად:

```
$quary = new CGI(\*STDIN);
```

თქვენ შეგიძლიათ ინიციალიზაცია გაუკეთოთ **CGI** ობიექტს ფაილის სახელით ან **IO: : File** ობიექტით. თუ თქვენ იყენებთ ფუნქციებზე ორიენტირებულ ინტერფეისს და გსურთ **CGI** მდგომარეობის ინიციალიზაცია ფაილის სახელის საშუალებით, ეს შეგიძლიათ გააკეთოთ **restore_parameters ( )** ფუნქციით. ეს აღადგენს გაჩუმების პრინციპით დაყენებულ **CGI** ობიექტის მდგომარეობას მითითებული სახელიდან.

```
open(IN, “test.in”)|| die;
restore_ parameters(IN);
close (IN);
```

თქვენ შეგიძლიათ ობიექტი-კითხვარის ინიციალიზაცია ასოციატიური მასივის მომთითებლით:

```
$quary = new CGI ({' dinosaur ' => ' banney ' ,
    ' song ' =>' I love you ' ,
    ' friends ' =>[qw/Jessica George Nancy /]});
```

ან პირდაპირ დაფორმატებული **URL** კითხვარი – სტრიქონიდან

```
$quary = new CGI(' dinozaur ' = ' barney ' & color=purple ' );
```

ან წინასწარ არსებული **CGI** ობიექტიდან (ამჟამად ეს დუბლირებას უკეთებს პარამეტრების სიას, მაგრამ არა ობიექტის სპეციფიკურ ველებს):

```
$old_query = new CGI;
$ new_squery = new CGI ($old_query);
```

რომ შექმნათ ცარიელი კითხვარი, ახდენთ ცარიელი სტრიქონის ან ხეშის (hash) ინიციალიზაციას:

```
$ empty_query = new CGI (“ “);
```

ან

```
$empty_query =  new CGI ({ });
```

კითხვარიდან გასაღები სიტყვების სიის გამოტანა

```
@ keywords=$query ->keywords
```

თუ სკრიპტი იყო გამოძახებული როგორც <INDEX> ძებნის შედეგი, გარჩეული გასაღები სიტყვები შეიძლება მიიღოთ keyword ( ) მეთოდით.

სკრიპტისთვის გადაცემული ყველა პარამეტრის სახელების მოტანა

```
@ names = $query -> param
```

თუ სკრიპტი გამოძახებული იყო პარამეტრების სიით (ე.ი. “name1=value1 & name2 = value2 & name3 = value3”), patram ( ) მეთოდი დააბრუნებს პარამეტრების სახელებს სიის სახით. თუ სკრიპტი გამოძახებული იყო როგორც <ISINDEX> სკრიპტი, მაშინ იქნება რაიმე კონკრეტული პარამეტრი სახელით ' keywords ' .

შენიშვნა: როგორც მე-5 ვერსიაში, პარამეტრიანი სახელების მასივი დაბრუნებული იქნება იგივე რიგით, რაც ის რიგი, რომელშიც პარამეტრები არიან განსაზღვრულნი ფორმაში (მხოლოდ ეს არ ეხება spec და ამგვარად არაა გარანტირებული).

ლექცია 4

რაიმე სახელიანი პარამეტრის მნიშვნელობის ან მნიშვნელობების მოტანა:

```
@values = $query  -> param (' foo ' );
```

ან

```
$value = $query -> param (' foo ' );
```

გადასცემს ეს param ( ) მეთოდი რაიმე არგუმენტს სახელიანი პარამერტის მნიშვნელობის მოსატანად. თუ პარამეტრი მრავალმნიშვნელობიანია (მაგალითად, მრავალმნიშვნელობიანი არჩევანი მცურავ (scrolling)) სიაში), თქვენ შეგიძლიათ მიიღოთ რაიმე მასივი. სხვა შემთხვევებში მეთოდი აბრუნებს რაიმე ერთ მნიშვნელობას.

სახელიანი პარამეტრის მნიშვნელობის (მნიშვნელობების) დაყენება:

```
$quary -> param(' foo ' , ' an ' , ' array ' , ' of ' values ' );
```

ეს აყენებს ' foo ' სახელიანი პარამეტრის მნიშვნელობას მნიშვნელობების მასივზე. ეს არის ერთი ხერხი AFTER ველის მნიშვნელობის შესაცვლელად, როცა სკრიპტი იყო გამოძახებული მანამდე

ერთხელ (სხვა ხერხი არის პარამეტრის გადაფარვა, რომელიც დაშვებულია ყველა მეთოდის მიერ, რომლებიც ფორმის ელემენტებს არ ქმნიან).

param ( ) აგრეთვე გამოიცნობს გამოძახების სახელიან პარამეტრულ სტილს, რომელიც აღწერილი იქნება უფრო დეტალურად მოგვიანებით:

```
$query ->parqam(-name =>' foo ' ,-values=>[' an ' , ' array ' , ' of ' , ' values ' ]);
```

ან

```
$query -> param (-name=>' foo ' , -value => ' thevalue ' );
```

სახელიან პარამეტრისათვის დამატებითი მნიშვნელობების მიერთება:

```
$query ->append(-name =>' foo ' , -values =>[' yet ' , ' more ' , ' values ' ]);
```

ეს მიუთითებს სახელიან პარამეტრის რაიმე მნიშვნელობას ან მნიშვნელობათა სიას. ეს მნიშვნელობები მიუერთდება პარამეტრს ბოლოში, თუ იგი არსებობს, სხვა შემთხვევაში პარამეტრი შეიქმნება. შევნიშნოთ, რომ ეს მეთოდი ცნობს მხოლოდ სახელიანი არგუმენტის გამოძახების სინტაქსს.

ყველა პარამეტრების იმპორტირება სახელთა სიაში:

```
$query -> import_names(' R ' );
```

ეს ჰქმნის ცვლადების მიმდევრობას ' R ' სახელთა არეში. მაგალითად, **$R : :foo, @R: foo**. გასაღებ სიტყვათა სიებისათვის ცვლადი **@R : : keywords** გამოჩნდება. თუ სახელთა არე არაა მოცემული, ეს მეთოდი იგულისხმებს **' Q '** –ს. გაფრთხილება: არ შემოიტანოთ რაიმე ' **main** ' –ში; ეს არის უსაფრთხოების დიდი რისკი !!!

ძველ ვერსიებში მეთოდს ერქვა **import ( )**. 2.20 ვერსიაში ეს სახელი ამოგდებული იყო, რომ აგვეცილებინა კონფლიქტი ჩადგმულ **perl** მოდულის **import** ოპერატორთან.

<u>რაიმე პარამეტრის მთლიანად ამოგდება</u>

```
$query -> delete (' foo ' );
```

ეს მთლიანად ამოაგდებს პარამეტრს. იგი ზოგჯერ სასარგებლოა პარამეტრების ხელმეორედ დაყენებისათვის, რომელთა გადაცემა არ გსურთ სკრიპტის გამოძახებებს შორის. თუ თქვენ იყენებთ ფუნქციების გამოძახების ინტერფეისს, გამოიყენეთ “**Delete ( )**” რომ აიცილოთ კონფლიქტი **Perl**-ში ჩადგმულ **delete** ოპერატორთან.

ყველა პარამეტრების ამოგდება

```
$quiery ->delete_all ( );
```

ეს გაასუფთავებს **CGI** ობიექტს მთლიანად. იგი შეიძლება იყოს სასარგებლო იმისათვის, რომ დარწმუნდეთ ყველა გაჩუმება არის აღებული, როცა თქვენ ქმნით შესავსებ ფორმას. გამოიყენეთ **Delete_all ( )** თუ თქვენ იყენებთ ფუნქციების გამოძახების ინტერფეისს.

პარამეტრების სიაზე პირდაპირი მიმართვა:

SQ -> param_fetch(' address ') ->[1}=' 1313 Mosking bt~id Lane;

თუ თქვენ გჭირდებათ პარამეტრების სიაზე მიმართვა ისეთი ხერხით, რომელიც არაა უზრუნველყოფილი ამ მეთოდით, თქვენ უნდა გააკეთოთ პირდაპირი **reference** (მითითება) მასზე **param_fetch ( )** მეთოდით სათანადო სახელით. იგი დააბრუნებს მიმთითებელთა მასივს სახელიან პარამეტრებზე, რომელიც შემდეგ თქვენ შეგიძლიათ გამოიყენოთ როგორც გსურთ. თქვენ შეგიძლიათ, აგრეთვე, გამოიყენოთ რაიმე სახელიანი არგუმენტების სტილი -name არგუმენტის საშუალებით.

ლექცია 5
სკრიპტის მდგომარეობის შენახვა ფაილში:

$query -> save (Filehandle)

ეს ჩაწერს ფორმის მიმდინარე მდგომარეობას ფაილში. თქვენ შეგიძლიათ იგი უკან წაიკითხოთ **new ( )** მეთოდით ფაილის სახელის გათვალისწინებით. შევნიშნოთ, რომ **Filehandle** შეიძლება იყოს ფაილი, არხი ან სხვა რაიმე.

შენახული ფაილის ფორმატია:

```
NAME1=VALUE1
NAME1=VALUE1 '
NAME2=VALUE2
NAME3=VALUE#
```

როგორც სახელი, ასევე მნიშვნელობა არის **UBL escaped**. მნიშვნელობიანი **CGI** პარამეტრები წარმოიდგინებიან, როგორც გამეორებული სახელები. ჩანაწერები განცალკევებულია "=" ნიშნით. თქვენ შეგიძლიათ ჩაწეროთ მრავალი ჩანაწერი და წაიკითხოთ ისინი უკანვე **new**-ს რამოდენიმე გამოძახებით. თქვენ შეგიძლიათ გააკეთოთ ეს ფაილის გახსნით **append** რეჟიმში, რაც საშუალებას მოგცემთ შექმნათ სტუმართა მარტივი წიგნი ან მომხმარებელთა შეკითხვების ენა. განვიხილოთ მრავალი ჩანაწერის შექმნის მოკლე მაგალითი:

```
    USE CGI;
open Z(OUT,">>test.out") || die;
    $ records = 5;
foreach (o ... $ records)  {
        my $q = new CGI;
    $q -> param (~name => ' counter ' , -value =>$_);
    $q -> save (OUT);
     }
     close OUT;
#  ხელმეორედ გახსნა წასაკითხად
 open (IN, "test.out") || die;
while (!eof (IN)) {
  my $q -> new (IN);
```

```
print $q -> param (' counter ' ), "\ n";
  }
```

' save/restore-ში გამოყენებული ფაილის ფორმატი იგივეა, რაც Genome Center ' –ის მონაცემთა გაცვლის ფორმატი "Boulderio" და შეგიძლიათ გამოიყენოთ, აგრეთვე, მონაცემთა ბაზის შექმნა Boulderio უტილიტებით. იხილეთ HTTP:// www.genome.wi.mit edu / genome_software / other / boulder.HTML თუ თქვენ გსურათ გამოიყენოთ ფუნქციებზე ორიენტირებული ინტერფეისი, ამ მეთოდის ექსპორტირებული სახელია save_paremeters ( ).

ლექცია 6
ფუნქციებზე ორიენტირებული ინტერფეისის გამოყენება

ფუნქციებზე ორიენტირებული ინტერფეისი რომ გამოიყენოთ, თქვენ უნდა დააზუსტოთ რომელი CGI.PM პროგრამები ან პროგრამათა სიმრავლეები შემოიტანოთ თქვენი სკრიპტების სახელთა არეში. მცირეა ის ხარჯები, რომლებიც დაკავშირებულია ამ იმპორტირებასთან:

use CGI < მეთოდების სია >;

ჩამოთვლილი მეთოდები იქმნება შემოტანილი მიმდინარე პაკეტში; თქვენ შეგიძლიათ გამოიძახოთ ისინი პირდაპირ თავდაპირველად CGI ობიექტის შექმნის გარეშე. მაგალითი გვიჩვენებს, როგორ შემოვიტანოთ param( ) და header( ) მეთოდები და შემდეგ როგორ გამოვიყენოთ ისინი:

```
use CGI ' param ' , ' header ' ;
print header (' text / plain ' );
$ zipcode = param (' zipcode ' );
```

უფრო ხშირად, თქვენ შემოგაქვთ ზოგადი ფუნქციების სიმრავლეები ჯგუფებზე მითითებით სახელით. ფუნქციების ყველა სიმრავლეები იწყებიან ":"-ით როგორც ": HTML3" (tag-ებისათვის განსაზღვრულ HTML 3 სტანდარტში). აქაა ფუნქციების სიმრავლეების სია, რომელიც თქვენ შეგიძლიათ შემოიტანოთ

: CGI

შემოაქვს ყველა CGI დამუშავების მეთოდები, როგორიცაა param ( ), path_info ( ) და მისთანები.

: form

შემოაქვს ფორმის შევსების მაგენერირებელი მეთოდები, როგორიცაა text field ( ).

: HTML2

შემოაქვს ყველა მეთოდები, რომლებიც გენერაციას უკეთებს HTML 2.0 სტანდარტულ ელემენტებს.

: HTML3

შემოაქვს ყველა მეთოდები, რომლებიც გენერაციას უკეთებენ HTML 3.0 შემოთავაზებულ ელემენტებს (როგორიცაა, < table >, < super > და< sub >).

: netscape

შემოაქვს ყველა მეთოდები, რომლებიც გენერირებას უკეთებენ Netscape დამახასიათებელ HTML გაფართოებებს.

: HTML

შემოაქვს ყველა მეთოდები, რომლებიც გენერირებას უკეთებენ შემოკლებებს (ე. ი. ' HTML2 ' + ' HTML3 ' + ' + ' netscape ') ...

: standard

შემოაქვს "standard" თვისებები, ' HTML2 ' , ' HTML3 ' , ' form ' და ' CGI. '

: all შემოაქვს ყველა შესაძლებელი მეთოდი. სრული სიისათვის იხილეთ CGI.PM კოდი, სადაც ცვლადი % Tags არის განსაზღვრული.

თუ თქვენ შემოიტანთ ფუნქციის სახელს, რომელიც არაა CGI.PM-ის ნაწილი, ეს მოდული დაამუშავებს მას როგორც რაიმე ახალ HTML ჭდეს და გენერაციას გაუკეთებს შესატყვის ქვეპროგრამას. თქვენ შეგიძლიათ შემდეგ გამოიყენოთ იგი როგორც ნებისმიერი სხვა HTML ჭდე. ეს არის გათვალისწინებული სწრაფად განვითარებადი HTML "Standard"-ისათვის. მაგალითად, ვთქვათ Microsoft-ს შემოაქვს ახალი ჭდე სახელით <GRADIENT> (რომელიც წალეკავს მომხმარებლის სამუშაო მაგიდას მბრუნავი გრადიენტული შევსებით, სანამ კომპიუტრი ჩაიტვირთებოდეს). თქვენ არ გჭირდებათ დაელოდოთ CGI.PM –ის ახალ ვერსიას, რომ დაიწყოთ მისი გამოყენება უშუალოდ:

```
use CGI qw / : standard: HTML3 gradient /;
print gradient({-start =>' red ' , -end => ' blue ' });
```

შევნიშნოთ, რომ გამოთვლების სიჩქარის ინტერესებისათვი CGI.PM არ იყენებს ამ სტანდარტს Exporter manpage syntax ჩასატვირთი სიმბოლოების დასაზუსტებლად. მომავალში ეს შეიძლება შეიცვალოს. თუ თქვენ შემოიტანთ ნებისმიერ CGI მდგომარეობის მხარდამჭერ მეთოდიდან ფორმის მაგენერირებელ მეთოდებს, გაჩუმებით შექმნილი CGI ობიექტი იქნება შექმნილი და ინიციალიზებული ავტომატურად. პირველად, თქვენ იყენებთ ამ მეთოდებიდან ნებისმიერს, რომელიც მოითხოვს ერთერთს იყოს წარმოდგენილი. ამას მიეკუთვნება param ( ), textfiled ( ), sumbit ( ) და მათი მსგავსები (თუ თქვენ გჭირდებათ CGI ობიექტზე პირდაპირი მიმართვა, თქვენ შეგიძლიათ იპოვოთ იგი გლობალურ ცვლადში $CGI : : Q). CGI.PM მეთოდების იმპორტირებით თქვენ შეგიძლიათ შექმნათ შესახედავად ელეგანტური სკრიპტები:

```
use CGI qw /: standard/;
print
    header,
    start_HTML (' Simple Script ' ),
    h1(' Simple Script ' ),
    start_form,
    "What's your name? ", textfield (' name ' ), p,
    "What's the combination?",
    cheekbox_group (-name =>' Words ' ,
        -values => [' eenie ' , ' meenie ' , ' minib ' , ' moe ' ],
        -defauts => [' eenie ' , ' moe ' ]), p,
    "What's your favorite color?",
    popul_menu (-name => ' color ' ,
        -values => [' red ' , ' green ' , ' blue ' , ' chartreuse ' ]), p,
```

```
 submit ,
 end_form,
 hr, " \ n";
 if (param){
  print
     "Your name is ", em (param (' name ' )), p,
     "The keywords are:", em(Join(",", param(' words ' ))), p,
     "Your favorite color is", em (param(' color ' )),"in";
}
print  end_HTML;
```

ლექცია 7

PRAGMAS

ფუნქციების სიმრავლეების გარდა, არის **pragmas**-ების გარკვეული რაოდენობა, რომლებიც თქვენ შეგიძლიათ შემოიტანოთ **pragmas**-ების წინ ყოველთვის იწერება დეფისი (-) და ცვლის წესს, რომლითაც **CGI.PM** ფუნქციონირებს სხვადასხვა ხერხით. პროგრამები, ფუნქციების სიმრავლეები და ინდივიდუალური ფუნქციები შეიძლება იყოს შემოტანილი ერთიდაიგივე **USE ( )** სტრიქონზე. მაგალითად, შემდეგ **USE** ინსტრუქციას შემოაქვს ფუნქციების სტანდარტული სიმრავლე და კრძალავს გამართვის რეჟიმს (**pragma no_debug**):

```
USE CGI qw / : standard –no_debug /;
```

pragmas-ების ამჟამინდელი სიაა შემდეგი:

-any

როცა თქვენ იყენებთ **CGI -any**, მაშინ ნებისმიერი მეთოდი, რომელსაც **query** ობიექტი ვერ გამოიცნობს იქნება გაგებული როგორც ახალი **HTML** ჭდე. ეს ნებას გრთავთ თქვენ, შემოიტანოთ **Netscape**-ის ან **Microsoft HTML**-ის შემდგომი გაფართოება. ამით თქვენ შემოგაქვთ ახალი უცნობი ჭდე:

```
USE CGI qw(-any);
$q = new CGI;
print $q -> gradient ({speed =>' fast ' , start      =>' red ' , and =>' blue ' });
```

თუმცა < **cite** > ნებისმიერი რამ < / **cite** > იწვევს მცდარი მეთოდის სახელს, რომელიც უნდა იყოს გაგებული როგორც **HTML** ჭდე, გამოიყენეთ იგი დიდი სიფრთხილით ან სრულიადაც არ გამოიყენოთ.

-compile

ეს იწვევს მინიშნებულ ავტომატურად ჩატვირთული მეთოდების კომპილირების წინდაწინ და არა მათ გადადებას შემდეგისათვის. ეს არის სასარგებლო სკრიპტებისთვისაც, რომლებიც გაშვებული არიან დიდი ხნით **Fast CGI**-ში ან **mod perl**-ში და მათთვის, რომლებიც გამიზნული არიან **Malcom Beattie's Perl** კომპილიატორისათვის, გამოიყენეთ იგი მეთოდებთან ან მეთოდების ოჯახებთან ერთად, რომელთა გამოყენებას თქვენ გეგმავთ.

```
USE CGI qw (-compile): standard : HTML3);
```

ან კიდევ

USe CGI qw (-compile : all);

შევნიშნოთ რომ –compile pragma–ს ასეთი გამოიყენება ყოველთვის იწვევს კომპილირებული ფუნქციების შემოტანას სახელთა არეში. თუ თქვენ გსურთ კომპილირება შემოტანის გარეშე, გამოიყენეთ compile ( ) მეთოდი (იხილეთ ქვემოთ).

-nph

ეს ქმნის დაზუსტებულ სათაურს NPH (no parsed header) სკრიპტისათვის. თქვენ შეგიძლიათ ამით უთხრათ სერვერს, რომ სკრიპტი არის NPH. იხილეთ ქვემოთ NPH სკრიპტების თაობაზე დისკუსია.

-autoload

ეს გადაფარავს ავტოჩამტვირთველს ისე, რომ ნებისმიერი ფუნქცია, რომელიც არ იქნება გამოცნობილი გადაეცემა CGI.PM შესაძლო შეფასებისათვის. ეს საშუალებას მოგცემთ გამოიყენოთ CGI.PM ფუნქციები სიმბოლოების ცხრილში მათი ჩამატების გარეშე და იგი ეხება mod_perl-ის მომხმარებლებს, რომელთაც მეხსიერების დახარჯვის ეშინიათ. გაფრთხილება: როცა –autoload ძალაშია თქვენ არ შეგიძლიათ გამოიყენოთ ფუნქციის სახელი ფრჩხილების გარეშე. გამოიყენეთ hr ( ) hr-ის ნაცვლად ან დაუმატეთ subs qw / hr p header / თქვენი სკრიპტის თავში.

-no_debug

ეს გამორთავს ბრძანებების დამუშავების თვისებებს. თუ თქვენ გსურთ გაუშვათ CGI.PM სკრიპტი HTML-ის მისაღებად ბრძანებით და თქვენ არ გსურთ ელოდოთ CGI პარამეტრებს სტანდარტული შეტანიდან ან ბრძანებიდან, მაშინ გამოიყენეთ ეს პროგრამა:

USE CGI qw (-no_debug : standard);

თუ თქვენ გსურთ გამოიყენოთ ბრძანება პარამეტრებით, მაგრამ არა სტანდარტული შეტანით, მაშინ გამოიყენეთ

```
USE CGI qw (-no_debug : standard);
 restore_parameters (Join(' & ' @ ABGV);
```

იხილეთ debug სექცია უფრო მეტი ინფორმაციისათვის.

-private_tempfiles

CGI.PM შეუძლია შეასრულოს ჩატვირთული ფაილი. ჩვეულებრივ იგი აგდებს ჩატვირთულ ფაილს დროებით დირექტორიაში, და მისი შესრულების შემდეგ ამოაგდებს ფაილს. მაგრამ ეს სარისკოა, როგორც აღწერილია ფაილის ჩატვირთვის სექციაში. მეორე CGI სკრიპტის ავტორს შეუძლია შეხედოს ამ მონაცემებს მისი ჩატვირთვისთვისაც საიდუმლო ინფორმაციის შემთხვევაშიც კი. Unix სისტემაში ეს არ მოხდება.

ლექცია 8

დინამიური დოკუმენტების გენერირება

უმეტესობა CGI.PM ფუნქციები ქმნიან დოკუმენტებს შესრულების პროცესში. საზოგადოდ, თქვენ ქმნით ჯერ HTTPდასათაურებას და შემდეგ თვით დოკუმენტს. CGI.PM აქვს ფუნქციები სხვადასხვა ტიპის სათაურების და HTML-ების

შესაქმნელად. GIF სურათების შესაქმნელად, იხილეთ GD.pm მოდული. ყოველი ფუნქცია ქმნის HTML-ის ან HTTP-ს ფრაგმენტს, დაუმატოთ სტრიქონს ან შეინახოთ ფაილში შემდგომი გამოყენების მიზნით.

სტანდარტული HTTP სათაურის შექმნა

ჩვეულებრივ , პირველად რასაც თქვენ აკეთებთ CGI სკპიპტში, არის HTTP სათაურის გამოტანა. ეს ეუბნება ბროუზერს რა ტიპის დოკუმენტს უნდა ელოდოს და აძლევს სხვა არაუცილებელ ინფორმაციას, როგორიცაა გამოყენებული ენა, დოკუმენტის ვადა და დოკუმენტი უნდა იყოს ფარული თუ არა. სათაური შეიძლება იყოს გამოყენებული აგრეთვე სერვერის დასამალავად და ვებ გვერდების ჩვენებისათვის საჭირო თანხის გადასახდელად.

```
print $query -> header;
```

ან

```
print $quary -> header (' image / gif ' );
```

ან

```
prinr $quary -> header(' text / HTML '   , ' 204 No response ' );
```

ან

```
print $quary -> header(' –type =>' image / gif ' ,
        -nph =>1,
        -status =>' 402 Reymant requered ' ,
        -espires =>' +3dzbr ,
        -cookie => $cookie,
        -Cost => ' $ 2.00 ' );
```

header ( ) აბრუნებს Content-type : he ader. თქვენ შეგიძლიათ გაითვალისწინოთ თქვენი საკუთარი MIME ტიპი. თუ თქვენ აირჩევთ მას, სხვა შემთხვევაში გაჩუმებით იგულისხმება text / HTML. არასავალდებულო მეორე პარამეტრი აზუსტებს სტატუსის კოდს და ადამიანის მიერ წასაკითხ შეტყობინებას. მაგალითად, თქვენ შეგიძლიათ დააზუსტოთ 204 “NO response”-შექმნათ სკრიპტი, რომელიც ეუბნება ბროუზერს არაფერი არ მოიმოქმედოს.

უკანასკნელი მაგალითი გვიჩვენებს სახელიანი პარამეტრების გამოყენებას CGI მეთოდებში არგუმენტების გადასაცემად სახელიანი არგუმენტების სტილით. დასაშვები პარამეტრებია: - type, -status, -expires და cookie. ნებისმიერ სხვა პარამეტრს მოეცლება ტირე (hypen) და ისე დაუბრუნდება სათაურის ველს, რაც საშუალებას გაძლევთ დააზუსტო HTTP სათაურები ისე, როგორც თქვენ გსურთ. თავდაპირველი ქვედა ხაზები გადაიყვანება დეფისებში:

```
print $quare -> header (-Content_length => 3002);
```

ბროუზერების უმეტესობა არ მალავენ CGI სკრიპტებიდან გამოტანებს. ყოველთვის როცა ბროუზერი ჩატვირთავს გვერდს სკრიპტი გამოიძახება ხელახლა. თქვენ შეგიძლიათ შეცვალოთ ბროუზერის ასეთი ქცევა –expires პარამეტრით. როცა თქვენ მიუთითებთ აბსოლოუტურ ან ფარდობით ვადას ამ პარამეტრით ზოგიერთი ბროუზერები და proxy სერვერები დამალავენ გამოტანას ვადის გასვლამდე. დასაშვებია შემდეგი ფორმები –expires ველისათვის:

+30 s	30 სეკუნდი მოცემული მომენტიდან
+10m	10 წუთი მოცემული მომენტიდან

+1h	ერთი საათი მოცემული მომენტიდან
-1d	გუშინ
now	ახლავე
+3m	სამ თვეში
+3m	სამ თვეში
+10y	10 წელიწადში

Thursday, 25-Apr-1999 00:40:33 GMT ნაჩვენებ დროსა და თარიღში

-cookie ქმნის სათაურს, რომელიც ეუბნება ბროუზერს გაითვალისწინოს "**mage cookie**" სკრიპტთან დაკავშირების ყველა შემთხვევებში. **Netscape**-ის აქვს სპეციალური ფორმატი, რომელიც შეიცავს საინტერესო ატრიბუტებს, როგორიცაა ჭდის დადება. გამოიყენეთ **cookie ( )** მეთოდი, რომ შექმნათ და იპოვოთ პროცესის ინდიკატორის cookie სესიები. **–nph** პარამეტრს თუ აქვს **true** მნიშვნელობა შექმნის სწორ სათაურებს, იმუშაოთ **NPH** სკრიპტთან. ეს მნიშვნელოვანია, რომ გამოიყენოთ ზოგიერთ სერვერებთან, როგორიცაა **Microsoft Internet Explorer** , რომელიც მოელის, რომ ყველა თავისი სკრიპტები არიან **NPH**-ები.

ლექცია 9
გადამისამართებადი სათაურების გენერირება

```
print $quary -> redirect (' HTTP : || somewhere. else / in / movie / land ' );
```

ზოგჯერ, შეიძლება თქვენ არ გსურთ შექმნათ რაიმე დოკუმენტი თვითონ, არამედ მიმართოთ ბროუზერი სადმე, შეიძლება აირჩიოთ რაიმე **URL** დაფუძნებული დღის დროზე ან მომხმარებლის საცხოვრებელი ადგილის მიხედვით.

redirect ( ) ფუნქცია მიმართავს ბროუზერს სხვა **URL**-ზე თუ თქვენ გამოიყენებთ ასეთ გადამისამართებას, მაშინ თქვენ არ შგეძლებათ გამოიტანოთ სათაური აგრეთვე. როგორც **2.0** ვერსიაში. თქვენ ქმნით როგორც არაოფიციალურ **location : header**-ს ასევე ოფიციალურ **URL : header**-ს. ეს უნდა დააკმაყოფილოს უმეტესობა სერვისებმა და ბროუზერებმა.

ერთი მოსაზრება, რომელიც არ შეიმიძლია არ შემოგთავაზოთ არის ის, რომ ფარდობითმა კავშირებმა შეიძლება არ იმუშაონ სწორად, როცა თქვენ ქმნით გადამისამართებას სხვა დოკუმენტზე თქვენსავე საიტში. ეს გამოწვეულია კარგად მოფიქრებული ოპტიმიზაციით, რომელსაც ზოგიერთი სერვერი იყენებს. ასეთ შემთხვევაში უნდა გამოიყენოთ სრული **URL**. აგრეთვე, თქვენ შეგიძლიათ გამოიყენოთ სახელიანი არგუმენტები:

```
print  $quary -> redirect (-url =>'HTTP :|| somewhere.
    else (in / movie / land, nph =>1);
```

-nph პარამეტრის დაყენება **true**-ზე შექმნის სწორ სათაურს, რომელიც იმუშავებს **NPH** სკრიპტთან. ეს მნიშვნელოვანია **Microsoft internet Explorer**- სთვის, რომელიც მუშაობს მხოლოდ **NPH** სკრიპტებთან.

HTML დოკუმენტის სათაურის შექმნა

```
print $quary -> start_HTML (-title => ' Secrets of the Pyramids;
```

```
 -author =>' fred @ carpicorn.org ' ,
-base => ' true ' ,
-target => ' blank ' ,
-meta => {' keywords ' => ' pharaon secret mummy ' ,
  ' copyright ' =>' copyright 1966 King Tut ' },
-Style => {' src ' => ' / styles / style1.CSS ' },
-BGCOLOR => ' blue ' );
```

HTTP სათაურის შექმნის შემდეგ CGI სკრიპტების უმეტესობა იწყებენ HTML დოკუმენტების გამოტანას. start_HTML ( ) ფუნქცია ქმნის გვერდით ზედა ნაწილს, სხვა არასავალდებულო ინფორმაციის გათვალისწინებით, რომელიც მართავს გვერდის გარეგნობასა და ქცევას. ეს მეთოდი აბრუნებს HTML სათაურს და < BODY > ჭდეს. ყველა პარამეტრი არასავალდებულოა. სახელიანი პარამეტრების ფირმით გამოცნობილი პარამეტრების –title, -auther, -base, -xbase და –target. ნებისმიერ-დამატებითი პარამეტრები, რომლებსაც თქვენ შემოიტანთ, როგორიცაა Netscape-ის არაოფიციალური BGCOLOR ატრიბუტი, დაემატება < BODY > ჭდეს. დამატებითი პარამეტრები უნდა იწყებოდნენ დეფისით.

-xbase არგუმენტი საშუალებას იძლევა გაითვალისწინოთ HRBF < BASE > ჭდისთვია, რომელიც მიმდინარე ადგილისაგან განსხვავებულ ადგილს მიუთითებს, როგორც შემდეგ ჩანაწერშია მოცემული:

-xbase => “HTTP : // home.mcom.com /”;

ყველა ფარდობითი კავშირი განხილული იქნება, როგორც მოცემული tag-ის მიმართ ფარდობითობა.

-target არგუმენტი საშუალებას იძლევა გაითვალისწინოთ გაჩუმებით target ფრეიმი ყველა კავშირისათვის და ამ გვერდის შესავსები ფორმები. იხილეთ Netseape-ის დოკუმენტაცია ფრეიმების შესახებ.

-target => ”answer_window”.

ყველა ფარდობითი კავშირი განიხილება, როგორც ფარდობითი ამ ჭდის მიმართ. თქვენ უმატებთ ნებისმიერ მეტ ინფორმაციის სათაურს –meta არგუმენტის საშუალებით. ეს არგუმენტი გულისხმობს მითითებას ასოციატიურ მასივზე, რომელიც შეიცავს მეტ ინფორმაციის
სახელი / მნიშვნელობა წყვილებს. ისინი წარმოიდგინებიან შემდეგნაირად:

```
< META NAME = “Keywords” CONTENT = “pharaon secret mummy”>
<MET VALUE =”description” CONTENT=”copyright 1996 King Tut” >
```

არ არსებობს არავითარი საშუალება HTTP_EQUIV ტიპისათვის. იმიტომ რომ თქვენ შეგიძლიათ HTTP სათაური დააზუსტოთ პირდაპირ header ( ) მეთოდით. მაგალითად, თუ თქვენ გსურთ გააგზავნოთ

-Refresh : header ეს შეგიძლიათ გააკეთოთ header ( ) მეთოდით:

```
print $q -> header (-Refresh =>' 10; URL=HTTP :www.carpicorn.com ' );
```

-style ჭდე გამოიყენება იმისათვის, რომ ჩაუმატოთ კასკადური ფურცლები თქვენს კოდში. იხილეთ CASCADING stylesheets სექცია.

თქვენ შეგიძლიათ მოათავსოთ სხვა ნებისმიერი HTML ელემენტი <HEAD > სექციაში –head ჭდით. მაგალითად, რომ მოვათავსოთ იშვიათად გამოყენებული < LINK > ელემენტი სათაურის სექციაში გამოიყენეთ:

```
print $q -> start_HTML (-head => Link({-rel =>' next ' ,
  -hef =>' HTTP: | | www. carpicorn.com (s2.HTML ' }));
```

რომ ჩავრთოთ მრავალი **HTML** ელემენტი <HEAD > სექციაში, გადაეცით მიმთითებელთა მასივი:

```
print $q -> start_HTML (-head => [
        Link ({-rel =>' next ' ,
          -href => ' HTTP : // www.carpicorn.com / s2.HTML ' }),
        Link ({-rel => ' previous ' ,
          -href => ' HTTP: //www.carpicorn. com / s1. HTML ' })
          ]
       );
```

Java სკრიპტების გამოყენება:

-script, -noScript, -onLoad, -onMouseOver, -onMouseOut და –onUnload პარამეტრები გამოიყენებიან, რომ დაუმატოთ Netscape JavaScript-ების გამოძახებები თქვენს გვერდებს. –script უნდა მიუთითებდეს ტესტის ბლოკს, რომელიც შეიცავს Javascript ფუნქციების განსაზღვრებებს. ეს ბლოკი მოთავსდება < script > ბლოკის შიგნით **HTML** სათაურში. ეს ბლოკი მოთავსებულია სათაურში იმისათვის, რომ მივცეთ თქვენს გვერდს შანსი ჰქონდეს ყველა Javascript ფუნქცია თავის ადგილას იმ სემთხვევაშიც კი როცა მომხმარებელი დააჭერს stop ღილაკს, სანამ გვერდი ჩაიტვირთებოდეს სრულად. **CGI.PM** ცდილობს დააფორმატოს სკრიპტი ისეთნაირად, რომ Javascript-ის ბროუზერმა ვერ გამორთოს კოდი. სამწუხაროდ ზოგიერთი ბროუზერები ასეთ შემთხვევაში ვერ მუშაობენ სწორად.

-onLoad და –onUnlod პარამეტრები მიუთითებენ Javascript კოდის ფრაგმენტების გამოთვლას გვერდის გახსნისას და დახურვისას. ჩვეულებრივ ეს პარამეტრები გამოიძახებიან –script ველში განსაზღვრული ფუნქციებისათვის:

```
$quary = new CGI;
 print  $quary -> header;
 $JSCRIPT =<< END;
 function riddle_me_this ( ) {
 var  r = prompt (“What wolks on four legs in the morning,”+
       “two legs in the after noun,”+
       “and three legs in the evining?”);
 response (r);
}
 function responce (answer){
   if(answer = = „man“)
  alert („Right you are!“);
   else
    alert („Wrong! Guess again.“);
 }
 END
print quary -> start_HTML (-title =>' The Riddle of the Sphings“,
            -script => $JSCRIPT);
```

გამოიყენეთ –noScript პარამეტრი, რომ გადასცეთ **HTML** ტექსტი, რომელიც გამოიტანება ბროუზერებზე, რომლებსაც არა აქვთ JavaScript (ან ბროუზერებზე, რომლებზეც JavaScript გამორთულია). Netscape 3.0 ცნობს < SCRIPT > ჭდის ზოგიერთ ატრიბუტებს LANGUAGE და SRC-ის ჩათვლით. უკანასკნელი

საინტერესოა, რადგან იგი ნებას გაძლევთ გქონდეთ JavaScript კოდი ფაილში ან CGI სკრიპტში. რომ გამოიყენოთ ეს ატრიბუტები გადეცით HASH მიმთითებელი –script პარამეტრით, რომელიც შეიცავს ერთ ან მეტ –language, -src ან –code-ს:

```
print $q -> start_HTML (-title => ' The Riddle of the Sphinx ' ,
    -script => {-language +> ' JavaScript ' ,
    -scr =>' / JavaScript    / sphinx. Js ' });
print $q -> (-title => ' The Riddle of the Sphinx ' ,
    -script => {-language => ' PERLSCRIPT ' },
    -code =>' print    „“hello world! \ n ' ;“ );
```

ბოლო თვისება საშუალებას გაძლევთ ჩართოთ მრავალი < SCRIPT > სექციები სათაურში. გადეცით სკრიპტ სექციების სია, როგორც მიმთითებლების მასივი. ეს საშუალებას მოგცემთ დააზუსტოთ სხვადასხვა წყარო ფაილები JavaScript-ის სხვადასხვა დიალექტებისათვის. მაგალითად:

```
print $q -& gt; start_HTML (-title =& gt;
  ' The Riddle of the Sphinx ' , -script =& gt; [
    {-language =& gt; ' /JavaScript 1.1 ' ,
    -src = & gt; ' / JavaScript / utilities 11.Js '
    }
    {-language = & gt; ' JavaScript 1.2 ' ,
    -scr = & gt; ' / JavaScript / utilities 12. Js '
    }
    {-language = & gt; ' JavaScript 28.2 ' ,
     -src    = & gt;  ' / JavaScript / utilities 219.Js '
    }
  ]
 );
< / pre >
```

იხილეთ

HTTP:| | home : netscape. com / eng / mozilla / 2.0 / handbook / JavaScript /

დამატებითი ინფორმაციისათვის * JavaScript-ების შესახებ.

ძველი სტილი პოზიტიური პარამეტრებისათვის არის:

Parameters:

დასათაურება

º The author‘s e_mail adress (შექმნის < LINK REV = “MADE” > ჭდეს თუ წარმოდგენილია.

º “True” დროშა თუ გსურთ ჩასვათ < BASE > ჭდე სათაურში. ეს დაგეხმარებათ გადაიყვანოთ ფარდობითი მისამართები აბსოლუტურ მისამართებში, როცა დოკუმენტი გადაგაქვთ სხვაგან, გამოიყენეთ სიფრთხილით.

,5,6 ... ნებისმიერი სხვა პარამეტრები, რომლებიც თქვენ გსურთ ჩართოთ <BODX > ჭდეში. ეს არის მოხერხებული ადგილი Netscape-ს გაფართოების დასასმელად, როგორიცაა ფერები და wallpaper ნიმუშები.

HTML დოკუმენტის დამთავრება:

```
print $query -> end_HTML
```

ეს ამთავრებს HTML დოკუმენტს < / BODY > < / HTML> ჭდეების გამოტანით.

თვითმიმთითებელი URL-ის შექმნა, რომელიც ინარჩუნებს მდგომარეობის ინფორმაციას:

```
$myself = $query -> self_url;
print “<A HREF - $myself > I’m talking to myself. < / A >”;
```

self_url ( ) დააბრუნებს URL-ს, რომელიც, როცა აირჩევა, ხელმეორედ გამოიძახებს ამ სკრიპტს ყველა თავისი მდგომარეობის ინფორმაციით. ეს სასარგებლოა, როცა თქვენ გსურთ გადასცეთ დოკუმენტის შიგნით შიგა ღუზების გამოყენებით, მაგრამ არ გსურთ დაანგრიოთ ფორმის მიმდინარე შინაარსი.

```
$myself = $query -> self_url;
print “< A HREF = $myself # table1> See table 1 </ A >”;
print “< A HREF = $myself # table 2> See table 2 < / A >”;
print “< A HREF = $myself # yourself > See for yourself < / A >”;
```

თუ თქვენ გსურთ გქონდეთ მეტი მართვა მასზე რაც დაბრუნდა, გამოიყენეთ url ( ) მეთოდი.

თქვენ აგრეთვე შეგიძლიათ იპოვოთ შესრულებული კითხვარის სტრიქონი query_string ( )-ით:

```
$the_string = $query -> query_string;
```

სკრიპტის URL-ის მოპოვება

```
$full_url = $query -> url ( );
$full_url = $query -> )-full =>1);  # ალტერნატიული სინტაქსი
$relative_url = $wuery ->url (-relative=>1);
$absolute_url = $query -> url (-absolute = 1);
$url_urth_path = $query -> url (-path_info=> 1);
$url_with_path_and_query = $query ->url (-path_info=> 1, -query=> 1);
```

url ( ) აბრუნებს სკრიპტის URL-ს სხვადასხვა ფორმატებში. უარგუმენტოდ გამოძახებისას აბრუნებს URL-ის სრულ ფორმას მასპინძლის სახელისა და პორტის ნომერის ჩათვლით

```
HTTPi || your. host.com / path / to / script. CGI
```

თქვენ შეგიძლიათ დააზუსტოთ ეს ფორმატი შემდეგი სახელიანი არგუმენტებით: -absolute

თუ ჭეშმარიტია გამოიმუშავებს აბსოლუტურ URL-ს, ე.ი.

```
/ path / to / script. CGI-ს.
```

-relative

გამოიმუშავებს ფარდობით URL-ს. ეს სასარგებლოა როცა გსურთ ხელმეორედ გამოიძახოთ თქვენი სკრიპტი განსხვავებული პარამეტრებით. მაგალითად:

```
script. CGI
```

-fall

გამოიმუშავებს სრულ Url-ს, თითქოს გამოძახებული ყოფილიყოს უარგუმენტებოდ. ეს გადაფარავს –relative და –absolute არგუმენტებს.

-path (-path_info)

აფართოებს დამატებითი path ინფორმაციით URL-ს. იგი შეიძლება იყოს კომბინირებული –full, -absolute ან –relative არგუმენტებთან. –path_info გათვალისწინებულია როგორც სინონიმი.

სტანდარული **HTML** ელემენტების შექმნა:

CGI.PM განსაზღვრავს THML შემოკლების მეთოდებს თითქმის ყველა **HTML** 3 და **HTML** 4 ჭდეებისთვის. **HTML** შემოკლებები დასახელებული არიან ცალკეული **HTML** ელემენტების შემდეგ და აბრუნებენ **HTML** ტექსტის ფრაგმენტს, რომელიც თქვენ შეგიძლიათ გამოიტანოთ ან დაამუშაოთ როგორც თქვენ გსურთ. ყოველი შემოკლება აბრუნებს **HTML** კოდის ფრაგმენტს, რომელიც შეგიძლიათ მიუერთოთ რაიმე სტრიქონს, შეინახოთ ფაილში ან გამოიტანოთ ბროუზერის ფანჯარაში. მაგალითი გვიჩვენებს როგორ გამოიყენოთ **HTML** მეთოდები:

```
$q = new CGI;
  print $q -> blockquote (
    “Many years ago on the island of”,
   $q -> a ({ href=> “HTTP : || crete. org / “; }, “Crete “),
     “there lived a minotaur named “,
     $q -> strong (“Fred”.),
        ),
   $q -> hr;
```

ეს შედეგად გვაძლევს შემდეგ **HTML** კოდს (ახალი სტრიქონები დამატებული აქვს წაკითხვის გამარტივების მიზნით):

```
< blockquote >
Many years ago on the island of
< a HREF = “HTTP! || crete. org / “; > Crete < / a > there lived
 a minotaur named < strong > Fred. < / strong >
< / blockquote >
< hr >
```

თუ თქვენ თვლით რომ, შემოკლებების გამოძახების სინტაქსი მოუხერხებელია, თქვენ შეგიძლიათ შემოიტანოთ ისინი თქვენ სახელთა არეში და გამოიყენოთ ობიექტზე ორიენტირებული სინტაქსით (იხილეთ შემდეგი სექცია):

```
use CGI ‘ : standard “;
  print blockquote (
  “Many years ago on the island of “,
   a ({ href=> “HTTP : || Crete. org / “; } “Crete”),
   “ there lived a minotaur named “,
       strong ( “ Fred “),
        ),
       hr;
```

HTML შემოკლებებისათვის არგუმენტების გათვალისწინება

HTML მეთოდებს შეიძლება ჰქონდეს ნული, ერთი ან მრავალი არგუმენტი. თუ თქვენ არ გამოიყენებთ არგუმენტებს მიიღებთ მხოლოდ ჭდეს:

```
print hr; # < HR >
```

თუ თქვენ გამოიყენებთ ერთ ან მეტ სტრიქონულ არგუმენტებს, მოხდება მათი კონკატენაცია ხარვეზების ნიშნის ჩამატებით და მოთავსდება გახსნილ და დახურულ ჭდეებს შორის:

```
print h1 (“Chapter “,”1”);  # < H1 > Chapter 1 < H1 >
```

თუ პირველი ელემენტია მიმთითებლის ასოციატიური მასივი, მაშინ გასაღებები და მნიშვნელობები გახდებიან **HTML** ჭდეების არგუმენტები:

```
print a ({-href => ‘fred. HTML’, -target =>’ –new’},
        :Open a new frame”);
< AREF = “fred. HTML”, TARGET = “-new” > Open a new frame < / A >
```

თქვენ შეგიძლიათ ჩაწეროთ ატრიბუტები დეფისების გარეშეც:

```
print  img {src => ‘ fred. gif’, align => ’LEFT’};
< IMG. ALIGN = “LEFT” SCR = “ fred. gif “ >
```

ზოგჯერ **HTML** ჭდეს არა აქვს არგუმენტი. მაგალთად, დალაგებული სიები შეიძლება იყოს მინიშნებული როგორც COMPACT. სინტაქსი ასეთ შემთხვევისათვის არის რაიმე არგუმენტი, რომელიც მიუთითებს განუსაზღვრელ სტრიქონს:

```
print ol ({ compact => undef }, li )’one’), li (‘two’), li (‘three));
```

ლექცია 10

HTML შემოკლებების დისტრიბუციული თვისება

HTML შემოკლებები დისტრიბუტულია. თუ თქვენ მისცემთ მათ რაიმე არგუმენტს, რომელიც მიუთითებს რაიმე სიას, tag-ი იქნება დისტრიბუტული სიის ყოველი ელემენტის გასწვრივ. მაგალითად, ქვემოთ მოცემული ხერხი ქმნის დალაგებულ სიას:

```
print ul. (li ({-type =>’disc’}, [‘Sneezy ‘,’Doc’,’Sleepy’,’Happy’]);
        );
```

ეს მაგალითი გვაძლევს შემდეგ **HTML** გამოტანას:

```
< UL >
  < LI TYPE = “disc” > Sneezy < / LI >
  < LI TYPE = “disc” > Doc < / LI >
  < LI TYPE = “disc” > SLEEPY < / LI >
  < LI TYPE = “disc” > Happy < / LI >
< / UL >
```

ეს გამოიყენება ცხრილების შესაქმნელად. მაგალითად:

```
print table ({-border => undef },
  caption (‘When Shood You Eat Your Vegitables?’),
   Tz ({-align => CENTER, -valign => TOP},
     [
      th ([ ‘Vegetable’,’Breakfast’, ‘Lunch’,’Dinner’],
      td ([ ‘Tomatoes’,’no’,’yes’,’yes’]),
      td ([ ‘Broccoli’, ‘no’, ‘no’, ‘yes’]),
      td ([ ‘Onions’, ‘yes’, ‘yes’, ‘yes’])
     ]
```

)
);

HTML შემოკლებები და სიის ინტერპოლაცია

განვიხილოთ

print blockquote (em('Hi'), 'mom!'g);

იგი აბრუნებს სტრიქონს:

< BLOCKQUOTE > <EM> HI< ?IEM > mom!< / BLOCKQUOTE >

მასივის ელემენტებს შორის ადგილის დატოვებას მართავს $" ცვლადი. მისი მნიშვნელობა იქნება ჩამატებული ელემენტებს შორის. როცა გამოგაქვთ სურათების სია, მაშინ სასურველია ცარიელი სტრიქონის ჩამატება სურათებს შორის. ამისათვის უნდა შევცვალოთ $"-ის მნიშვნელობა ცარიელი სტრიქონით.

{ local (S ") = ";
print blockquote (em ('Hi'), 'mom!');
}

ამ შემთხვევაში S" –ის მნიშვნელობა იცვლება ბლოკის შიგნით, გარეთ რჩება იგივე მნიშვნელობა.

არასტანდარტული **HTML** შემოკლებები

ზოგიერთი **HTML** ჭდეები არ იცავენ სტანდარტულ ნიმუშს სხვადასხვა მიზეზების გამო.

comment ( ) იძლევა **HTML** კომენტარს (<!- - comment-- >).

Perl-ის ჩადგმულ ფუნქციებთან კონფლიქტის გამო შემდეგი ფუნქციები იწყებიან დიდი ასოებით:

Select Tr Link Delete

დამატებით start_**HTML** ( ), end_**HTML** ( ), start_form ( ), end_form ( ), start_multipart_form ( ) და ფორმის შესავსები ყველა ჭდე სპეციალურია. იხილე შესატყვისი სექციები.

მსგავსი ფორმების შექმნა

შევნიშნოთ, რომ ფორმის შემქმნელი მეთოდები აბრუნებენ სტრიქონებს, რომლებიც შიცავენ ჭდეს ან ჭდეებს, რომელიც შექმნის ფორმის მოთხოვნილ ელემენტს. თქვენ ხართ პასუხისმგებელი ჭდეების სწორად განლაგებაზე. მეორე შენიშვნაა, რომ გაჩუმებით აღებული მნიშვნელობები გამოიყენებიან მხოლოდ პირველად სკრიპტის გამოძახებისას. გამოძახებების შემთხვევებში მაფორმირებლის მნიშვნელობები გამოიყენებიან თუნდაც ისინი იყვნენ ცარიელნი.

თუ თქვენ გსურთ რაიმე ველის კონკრეტული მნიშვნელობა თავისი წინა მნიშვნელობით, თქვენ გაქვთ ორი არჩევანი:

(1) გამოიძახეთ param ( ) მეთოდი მის დასაყენებლად.
(2) გამოიყენეთ –override (alias ~force) პარამეტრი. ეს აიძულებს გაჩუმებით აღებულ მნიშვნელობას იყოს გამოყენებული მისი წინა მნიშვნელობის მიუხედავად:

```
print $quary -> textfield (-name =>'field name',
        -default => 'starting value',
        -override => 1,
        -size => 50,
        -maxlenght => 80);
```

თქვენ შეგიძლიათ გამოიყენოთ „< CLICKME >“ როგორც რაიმე ღილაკის ჭდე. ეს არ გამოიწვევს რაიმე ორაზროვნებას. თუ თქვენ გსურთ გამორთოთ ავტომატური აცილება, გამოიძახეთ autoEscape ( ) მეთოდი false მნიშვნელობით **CGI** ობიექტის აგებისთანავე:

```
$query -> = new CGI;
$query -> autoEscape (undef);
```

ISINDEX tag –ის შექმნა

```
print $quary -> isindex (-action => $action);
```

ან

```
print $quary -> isindex ($action);
```

გამოიტანს < ISINDEX < ჭდეს. –action პარამეტრი მიუთითებს სკრიპტის URL-ს შეასრულოს შეკითხვა (query). გაჩუმებით იგულისხმება შეკითხვის შესრულება მიმდინარე სკრიპტით.

ლექცია 11

ფორმის დაწყება და დამთავრება

```
print $quary -> startform (-method => $method,
        -action => $action,
        -encoding => $encoding);
< ... various form stuff ...>
print $query -> endform;
```

ან

```
print $query -> startform ($method, $action, $encoding);
        < ...various form staff ... >
        print $query -> endform;
```

startform ( ) აბრუნებს < FORM > ჭდეს method-ით, action-ით და formencoding-ით, რომელთაც თქვენ მიუთითებთ. გაჩუმებით გვაქვს method : POST action : this script encoding : apploication / x-www-form-urlencoded

endform ( ) აბრუნებს დახურვის < / FDRM > ჭდეს. Startform ( )-ის endcoding method ეუბნება ბროუზერს როგორ შეფუთოს ფორმის სხვადასხვა ველები სანამ გაიგზავნებოდეს ფორმა სერვერზე. შესაძლებელია ორი მნიშვნელობა.

application / x-www-form-urlencoded

CGI.PM კოდირების ტიპის სახელს ინახავს $**CGI** :: URL_ENCODED-ში multipart / form_data

ეს არის კოდირების უახლესი ტიპი. იგი სასურველია ფორმებისათვის, რომლებსაც აქვთ დიდი ველები ან გამიზნულია მონაცემების გარდასაქმნელად. იგი ნებას

რთავს „file upload” თვისებას Netscape 2.0 ფორმებისათვის. **CGI.PM** ამ კოდირების ტიპის სახელი ინახავს
&CGI ::MULTIPART-ში.

ფორმები, რომლებიც იყენებენ კოდირების ამ ტიპს არ არიან იოლად დასამუშავებელი **CGI** სკრიპტებისათვის, თუ არ გამოიყენებენ **CGI.PM**-ს ან სხვა ბიბლიოთეკას, რომელიც გამიზნულია მათ დასამუშავებლად. თავსებადობის მიზნით startform ( ) მეთოდი იყენებს გაჩუმებით კოდირების ძველ ფორმას. თუ თქვენ გსურთ გამოიყენოთ კოდირების უახლესი ფორმა გაჩუმებით, თქვენ შეგიძლიათ გამოიძახოთ start_multipart_form ( ).

Java სკრიპტების გამოძახება: Java სკრიპტებში –name და onSubmit პარამეტრები გამოიყენებიან. –name პარამეტრი აძლევს ფორმას სახელს და იგი გამოიყენება Java script ფუნქციების მიერ მის იდენტიფიცირებისა და მანიპულირებისათვის. –onSubmit მიუთითებს Java script ფუნქციას, რომ იგი უნდა იყოს გამოთვლილი, სანამ ფორმა გადაეცემოდეს სერვერს. თქვენ შეგიძლიათ გამოიყენოთ ეს შესაძლებლობა ფორმის შინაარსის შერსამოწმებლად სისრულეზე. თუ თქვენ აღმოაჩენთ რაიმე შეცდომას, თქვენ შეგიძლიათ აამოქმედოთ საგანგაშო ყუთი ან დააფიქსიროთ რაიმე თვითონვე. თქვენ შეგძლიათ ამოაგდოთ გადაცემა false-ის დაბრუნებით ამ ფუნქციიდან. ჩვეულებრივ Java სკრიპტების ფუნქციები განსაზღვრული არიან **HTML** სათაურის
< SCRIPT > ბლოკში და –onSubmit მიუთითებს ერთ-ერთს ამ ფუნქციების გამოძახებიდან. იხილეთ start_**HTML** ( ).

ტექსტის ველის შექმნა

```
print $query -> textfield (-name =>’field_name’,
          -defult =>’starting value’,
          -size => 50,
         -maxlength
        ან
    print $query -> textfield (‘field_nme’, ‘starting value’, 50, 80);
```

textfield ( ) დააბრუნებს ტექსტის შესატან ველს.

პარამეტრები

პირველი პარამეტრია ველის სახელი (-name).

- მეორე არასავალდებულო პარამეტრია გაჩუმებით ნაგულისხმები სასტარტო მნიშვნელობა ველის შინაარსისათვის (-default).
- მესამე არასავალდებულო პარამეტრია ველის მოცულობა სიმბოლოებში (-size).
- მეოთხე არასავალდებულო პარამეტრია ველში სიმბოლოების მაქსიმალური რაოდენობა (-maxlength).

ყველა ამ მეთოდებით კონკრეტული ველის ინიციალიზაცია ხდება სკრიპტის ბოლო გამოძახებისას წინა შინაარსით. როცა ფორმა განხორციელებულია ტექსტური ველის მნიშვნელობა შეიძლება მივიღოთ შემდეგნაირად:

```
$value = $query -> param (‘foo’);
```

თუ თქვენ გსურთ აღადგინოთ იგი მისი საწყისი მნიშვნელობიდან, მას შემდეგ რაც სკრიპტი გამოძახებული იყო ერთხელ, თქვენ შეგიძლიათ გააკეთოთ იგი ასე:

```
$query -> param ('foo', "I'm taking over this value!");
```

სიახლე 2.15 თუ თქვენ არ გსურთ აიღოთ ველი მისი წინა მნიშვნელობიდან, თქვენ შეგიძლიათ აიძულოთ მიმდინარე მნიშვნელობის მიღება –override (alias-force) პარამეტრის გამოყენებით:

```
print $query-> textfield (-name =>'field_name',
        -default =>'starting value',
        -override =>1,
        -size => 50   ,
        -maxize => 80);
```

Java სკრიპტების გამოყენება: თქვენ შეგიძლიათ გამოიყენოთ –onChange, -onFocus, -onBlur, -onHouseOver, -onMokseOut და –onselect პარამეტრები, რომ დავარეგისტროთ Java სკრიპტის ხდომილებათა დამამუშავებლები. on Change დამამუშავებელი გამოიძახება, როცა მომხმარებელი ცვლის ტექსტური ველის შინაარსს. onFouse და onBlur გამოიძახებიან, როცა ჩასამატებელი წერტილი მოძრაობს ტექსტური ველის შიგნით და გარეთ. onSelect გამოიყენება, როცა მომხმარებელი ცვლის მონიშნულ ტექსტს.

ლექცია 12

ტექსტის დიდი ველის შექმნა

```
print $query -> textarea (-name => 'foo',
        -default =>' starting value',
        -rows => 10,
        -coliemens => 50);
```

ან

```
print $query -> textarea ('foo','starting value', 10, 50);
```

textarea ( ) ჰგავს textfield-ს, მაგრამ იგი საშუალებას გაძლევთ მიუთითოთ ველის საწყისი მნიშვნელობა, რომელიც შეიძლება მოითხოვდეს მრავალ სტრიქონს.
Java სკრიპტირება: -onChange, -onFocus, -onBlur, -onMouseOver, -onMouseOut და onSelect პარამეტრები გამოიყენებიან. იხილეთ textfield ( ).

პასპორტის ველის შექმნა

```
print $query ->password_field (-name => 'secret',
        -value =>'starting value",
        -size => 50,
        -maxlength => 80);
```

ან

```
print  $ query ->password_field ('secret', 'starting,  value', 50, 80);
```

პასპორტის ველის შინაარსი ვარსკვლავებით გამოიტანება ვებ გვერდზე.
Java სკრიპტების გამოყენება: იგივე პარამეტრები გამოიყენება, რაც წინა ველისათვის. იხილეთ textfield ( ).

ფაილის ჩატვირთვის ველის შექნმა

```
print $query -> fillefield (-name =>'uploaded_file',
        -defoult => 'starting value',
        -size => 50,
        -maxlength => 80);
```

ან

```
print $query -> filefield ('uploaded file', 'starting value', 50, 80);
```

filefield ( ) აბრუნებს ჩატვირთული ფაილის ველს Netscape 2.0 ბროუზერებისათვის. იმისათვის, რომ გამოვიყენოთ სრულად მისი უპირატესობები უნდა გამოიყენოთ ფორმისათვის მრავალნაწილიანი კოდირების სქემა. თქვენ შეგიძლიათ გააკეთოთ იგი startform ( )-ის გამოძახებით კოდირების ტიპით $CGI :: MULTIPART ან ახალი start_multipart ( ) მეთოდის გამოძახებით.

პარამეტრები

• პირველი პარამეტრია ველის სახელი (-name).

• მეორე არასავალდებულო პარამეტრია საწყისი მნიშვნელობა ველის შინაარსისათვის უნდა იყოს გამოყენებული, როგორც ფაილის სახელი გაჩუმებით (-default).

• მესამე არასავალდებულო პარამეტრია ველის მოცულობა სიმბოლოებში (-size).

• მეოთხე არასავალდებულო პარამეტრია ველის მაქსიმალური სიგრძე სიმბოლოებში (-maxlength).

როცა ფორმა რეალიზებულია, თქვენ შეგიძლიათ მიიღოთ შეტანილი ფაილის სახელი parm ( ) მეთოდის გამოძახებით.

```
$filename = $query -> param ('unluaded_file');
```

Netscape 2.0-ში დაბრუნებული ფაილის სახელი არის სრული ლოკალური ფაილის სახელი მომხმარებლის შორეულ მანქანაზე. თუ შორეული მომხმარებლის მანქანაა Unix-ი, მაშინ ფაილის სახელი აკმაყოფილებს Unix-ის შეთანხმებებს:

```
/ path / to / the / file
```

MS.DOS-ში WINDOWS-სა და OS/2 მანქანებისათვის გვექნება:

```
C:\ parth \ TO \ THE \ FILE.MSW
```

Macintosh-ის მანქანაზე კი

```
HD 40 : Desktop Folder : Sort Through : Reminders
```

დაბრუნებული ფაილის სახელი არის აგრეთვე ფაილის სახელური. თქვენ შეგიძლიათ წაიკითხოთ ამ ფაილის შინაარსი Perl ფაილის წაკითხვის სტანდარტული გამოძახებით:

```
While (< $filename >) {
   print;
   }
open (OUTFILE, ">> / usr / local / web / users / feed back");
while ($bytesread = read ($filename, $baffer, 1024)) {
   print OUTFILE $buffer;
   }
```

როცა ფაილი ჩაიტვირთება, ბროუზერი აგზავნის რაიმე ინფორმაციას სათაურების ფორმატში. ეს ინფორმაცია ჩვეულებრივ შეიცავს შინაარსის MINE ტიპს. მომავალში

ბროუზერმა შეიძლება გააგზავნოს უფრო მეტი ინფორმაცია (როგორიცაა თარიღი და მოცულობა). ამ ინფორმაციის მისაღებად გამოიძახეთ uploadInfo ( ). იგი აბრუნებს ასოციატიური მასივის მიმთითებელს, რომელიც შეიცავს დოკუმენტის სათაურებს.

```
$filename = $quary -> param ('uploaded_file');
$type = $query -> uploadinfo ($filename) -> {'Content-Type'};
unless ($type eq 'tex / HTML') {
    die " HTML FILES ONLY!";
     }
```

თუ თქვენ იყენებთ კომპიუტერს, რომელიც არჩევს "text" და "benary" მონაცემების ტიპებს, უნდა იყოთ დარწმუნებული როდის და როგორ გამოიყენოთ ისინი. წინააღმდეგ შემთხვევაში, შეიძლება აღმოჩნდეს, რომ ორობითი ფაილი დამახინჯებულია ფაილის ჩატვირთვისათვის.

Java სკრიპტების გამოყენება: -onChange, -onBlur, -onFocuis, -onMouseOver –on MouseOut და –onSelect პარამეტრები გამოიყენება. იხილეთ textfield ( ).

სტეკური (POPUR) მენიუს შექმნა

```
print $quary -> popur_menu ('menu_name',
        ['eenie', 'meenie', 'minib'],
          'meenie');
```

ან

```
% labels = ('eenie'=> 'your first choice',
  'meenie' => 'your second choice',
  'meenie'=> 'your third choice');
print $quary -> popup_menu ('menu_name',
    ['eenie', 'meenie', 'minie']
      'meenie',\ % labels);
```

ან (სახელიანი პარამეტრის სტილი)

```
print $quary-> popun_menu (-name => 'menu_name',
    -values => ['eenie', 'meenie', 'minie'],
    -default => 'meenie',
    -label => \ % labels);
```

popur_menu ( ) ქმნის მენიუს.

1. სავალდებულო არგუმენტია მენიუს სახელი (-name).
2. მეორე სავალდებულო არგუმენტია მასივის მიმთითებელი, რომელიც შეიცავს მენიუს ელემენტების სიას. თქვენ შეგიძლიათ გადასცეთ მეთოდს ანონიმური მასივი, როგორც ნაჩვენებია მაგალითში ან მომთითებელი სახელიან მასივზე, როგორიცაა "\ @ foo".
3. მესამე არასავალდებულო პარამეტრია გაჩუმებით აღებული მენიუს ელემენტი. თუ არაა მითითებული, მაშინ გაჩუმებით აიღება პირველი ელემენტი. წინა არჩევის მნიშვნელობები იქნება შენარჩუნებული კითხვების გასწვრივ.
4. მეოთხე არასავალდებულო პარამეტრი (-labels). გათვალისწინებულია მათთვის, ვისაც სურს მენიუს მნიშვნელობების მონიშვნა.

როცა ფორმა შესრულდება სტეკურ მენიუში არჩეული მნიშვნელობა შეიძლება მივიღოთ

$popur_menu_value = $query -> param (‘menu_name’);-თი

Java სკრიპტების გამოყენება: popur_menu ( ) იყენებს ხდომილებების შემდეგ დამმუშავებლებს: -onChange, -onFocus, -onMouseOver,-onMouseOut და onBlur. იხილეთ textfield ( ) სექცია.

ლექცია 13

მცურავი სიის შექმნა

```
print $wuary -> scrolling_list (‘ list_name’,
        [‘eenie’, ‘meenie’, ‘minie’, ‘moe’],
        [‘eenie’, ‘moe’], 5, ‘true’);
```

ან

```
print $quary – scrolling_list (-name => ‘list_name’,
    -values => [‘eenie’, ‘meenie’, ‘minie’, ‘moe’],
    -default => [‘eenie’, ‘moe’],
    -size > 5,
    -multiple => ‘true’,
    -labels => \ % labels);
```

scrolling_list ჰქმნის მცურავ სიას.

პარამეტრები:

პირველი და მეორე არგუმენტები არიან სიის სახელი და მნიშვნელობები.

მესამე არასავალდებულო არგუმენტია სიის მიმთითებელი, რომელიც შეიცავს გაჩუმებით არჩეულ მენიუს ელემენტებს. როცა გამოტოვებულია ან არაა მითითებული, მაშინ გაჩუმებით არცერთი ელემენტი არ აირჩევა.

მეოთხე არასავალდებულო არგუმენტია სიაში ელემენტების რაოდენობა.

მეხუთე არასავალდებულო არგუმენტი თუ დაყენებულია true-ზე, მაშინ შესაძლებელია მენიუში რამოდენიმე ელემენტის არჩევა. წინააღმდეგ შემთხვევაში, მხოლოდ ერთი ელემენტის არჩევა შეიძლება.

მეექვსე არასავალდებულო არგუმენტია მიმთითებელი ასოციატიურ მასივზე, რომელიც შეიცავს ელემენტების ჭდეებს.

როცა ფორმა შესრულდება, მენიუს არჩეული ელემენტები შეიძლება მივიღოთ:

@ selected = $wuary -> param (‘list_name’);

Java სკრიპტების გამოყენება: scrolling_list /) უშვებს შემდეგ ხდომილებათა დამამუშავებლებს: -onChange, onMouseOver, onFocus, -onMouseOut და –onBlur. იხილეთ textfield ( ) უფრო დეტალური აღწერისათვის.

დაკავშირებული უჯრების ჯგუფის შექმნა

```
print $query-> checkbox_group (-name =>’group_name’,
    -values => [‘eenie‘, ‘meenie‘, ‘minie‘, ‘moe’],
    -default =>[‘eenie’, ‘moe’],
```

-linebreak => ‘true’,
-labels => \ % labels);

ან

print $query -> checkbox_group (‘group_name’,
[‘eenie’, ‘meenie’, ‘minie’, ‘moe’],
[‘eenie’, ‘moe’], ‘true’, \ % labels);

HTML 3 –თან თავსებადი ბროუზერებისათვის მხოლოდ:

print $query -> checkbox_group (-name =>’group_name’,
-values =>[‘eenie’, ‘meenie’, ‘mine’, ‘moe’],
-rows = 2, -columns => 2);

checkbox_group ( ) ჰქმნის დაკავშირებული შესარჩევი უჯრების სიას, რომლებიც ერთიმეორესთან დაკავშირებულია საერთო სახელით.

პარამეტრები:

პირველი და მეორე პარამეტრები არიან შესამოწმებელი უჯრების სახელი და მნიშვნელობები. მეორე არგუმენტი უნდა იყოს მიმთითებელი მასივი. ეს მნიშვნელობები გამოიყენებიან მომხმარებლისათვის წასაკითხ ჭდეებად, რომლებიც შემდეგ გადაეცემა სკრიპტს კითხვარის სტრიქონში. მესამე არასავალდებულო არგუმეტია მიმთითებელი სიაზე, რომელიც შეიცავს მნიშვნელობებს, რომლებიც უნდა იყოს არჩეული გაჩუმებით. თუ ეს არგუმენტი გამოტოვებულია, მაშინ არაფერი არ აირჩევა გაჩუმებით. მეოთხე არასავალდებულო არგუმენტია სტრიქონის გაწყვეტა, რომლის ჭეშმარიტ მნიშვნელობის მიცემა ნიშნავს სტრიქონის გაწყვეტის მოთავსებას შესამოწმებელ უჯრებს შორის ისე, რომ შესამოწმებელი უჯრები გამოიტანება, როგორც ვერტილაკური სია, წინააღმდეგ შემთხვევაში იქნება სტრიქონის გასწვრივ. არასავალდებულო მეხუთე არგუმენტია ასოციატიურ მასივზე მიმთითებელი, რომელიც გამოიტანება როგორც ჭდეები მნიშვნელობებისათვის. HTML თავსებადი ბროუზერი იყენებს არასავალდებულო არგუმენტს –rows და –columns, რომლებიც გადააქცევენ შესარჩევ უჯრათა სიას ცხრილად. რომ მივიღოთ სტრიქონისა და სვეტის სათაურები დაბრუნებულ ცხრილში, თქვენ შეგიძლიათ გამოიყენოთ –row headers და –colheaders პარამეტრები. ორივენი წარმოადგენენ მიმთითებლებს სათაურების მასივზე. როცა ფორმა შესრულდება, ყველა არჩეული უჯრები იქნება დაბრუნებული, როგორც ცია სახელით ‚groip_name’. არჩეული უჯრების მნიშვნელობები შეიძლება მივიღოთ

@turned_0n =$query -> param (‘group_name’); -ით.

checkbox_group ( ) –ით დაბრუნებული მნიშვნელობა არის ელემენტების მასივი. თქვენ შეგიძლიათ გამოიყენოთ ისინი ცხრილების, სიების ან სხვა რაიმეს შიგნით:

@h = $query -> checkbox_group (-name => ‘group_name’,
-values => @values);
&use_in_creative _way ((@h);

Java სკრიპტების გამოყენება: checkbox_group ( ) ცნობს მხოლოდ –onClik პარამეტრს. ეს მიუთითებს Java skript-ის კოდი ან ფუნქციის გამოძახება იყოს შესრულებული ყოველთვის, როცა მომხმარებელი დააჭერს ჯგუფის ერთ-ერთ ღილაკს. თქვენ შეგიძლიათ იპოვოთ დაჭერილი ღილაკი „this” ცვლადის საშუალებით.

შესარჩევი უჯრის შექმნა

```
print $query -> checlbox (-name => 'checkbox_name',
        -checked => 'checked',
        -value => 'ON',
        -label => 'CLICK ME');
print $query -> checkbox ('checkbox_name', 'checked', 'ON', 'CLICK ME');
```

checkbox ( ) გამოიყენება ასარჩევი უჯრის შესაქმნელად.

პარამეტრები:

პირველი პარამეტრია სახელი.

მეორე არასავალდებულო პარამეტრი გვიჩვენებს, რომ უჯრა უნდა იყოს არჩეული გაჩუმებით. სინონიმია –select და –on.

მესამე არასავალდებულო პარამეტრი გვიჩვენებს არჩეული უჯრის მნიშვნელობას. როცა არა გვაქვს სიტყვა "on" იგულისხმება.

მეოთხე არასავალდებულო პარამეტრია უჯრასთან მიერთებული ჭდე. თუ არა გვაქვს ეს პარამეტრი, მაშინ ასარჩევი უჯრის სახელი გამოიყენება. ასარჩევი უჯრის მნიშვნელობა მიიღება

$turned_on = $query -> param ('checkbox_name');-ით.

Java სკრიპტების გამოყენება: checkbox ( ) ცნობს მხოლოდ –on Click პარამეტრს.

რადიო ღილაკების ჯგუფის შექმნა

```
print $query -> radio_group (-name +> 'group_name',
        -values => ['einie', ' meenie', 'menie'],
          -default => 'meenie',
          -lnebreak  => 'true',
          -labels => \ % labels);
```

ან

```
print $query -> radio_group ('group_name', ['eenie', 'meenie', 'minie'],
            'meenie', 'true', \ % labels);
```

HTML 3 –თან თავსებადი ბროუზერებისათვის მხოლოდ:

```
print $query – radio_group (-name=>'group_name', -values =>
['eenie', 'meenie', 'menie', 'moe'], rows =2, -columns => 2);
```

radio_group ( ) ქმნის ლოგიკურად დაკაშირებულ ღილაკებს (ერთი ღილაკის ჩართვა იწვევს სხვა ღილაკების გამორთვას).

პარამეტრები:

პირველი სავალდებულო არგუმენტია ჯგუფის სახელი (-name). მეორე არგუმენტი (-values) არის მნიშვნელობათა სია რადიო ღილაკებისათვის. მნიშვნელობები და ჭდეები ერთიდაიგივეა, რომლებიც გამოჩნდება გამოტანილ გვერდზე. გადეცით მეორე არგუმენტად მიმთითებელთა მასივი ან გამოიყენეთ ანონიმური მასივი ან მიუთითეთ სახელიანი მასივი როგორიცაა „\ @ foo".

მესამე არასავალდებულო პარამეტრია (-default) გაჩუმებით ჩართული ღილაკის სახელი. თუ არაა მოცემული პირველი იქნება არჩეული გაჩუმებით. თქვენ შეგიძლიათ მიუთითოთ არარსებული ღილაკის სახელი "-", რაც უზრუნველყოფს, რომ არცერთი ღილაკი არ იქნება არჩეული გაჩუმებით.

მეოთხე არასავალდებულო პარამეტრია (-labels) მიმთითებელი ასოციატიურ მასივზე, რომელიც არის მნიშვნელობათა ჭდეები გამოტანისას. თუ არაა მოცემული, მაშინ თვით მნიშვნელობები გამოიტანებიან.

HTML –თან თავსებად ბროუზერებში შეიძლება გვქონდეს არასავალდებულო პარამეტრები –rows -columns. ისინი ქმნიან ცხრილს, რომელიც შეიცავს რადიო ღილაკებს. თქვენ თუ მიუთითებთ სვეტების რაოდენობას, საჭირო სტრიქონების რაოდენობას radio_group გამოთვლის თვითონ.

რომ ჩართოთ სტრიქონებისა და სვეტების სათაურები დაბრუნებულ ცხრილში, თქვენ შეგიძლიათ გამოიყენოთ –rowheader და –colheader პარამეტრები. ორივენი იყენებენ მიმთითებლებს მასივზე.

როცა ფორმა გამოიტანება, არჩეული რადიო ღილაკი თქვენ შეიძლება მიიღოთ:

$which_radio-button = $query ->param (‘group_name’); -ით

radio_group ( ) –ის დაბრუნებული მნიშვნელობაა ღილაკების მასივი. თქვენ შეგიძლიათ დაიჭიროთ და გამოიყენოთ ისინი ცხრილებს, სიების შიგნით ან სხვა სახელით:

```
@h = $query -> radio_group (-name => ‘ group_name’, -values => @values;
&use_in_creative_way (@h);
```

ლექცია 14

```
print $query -> submit (-name => ‘button_name’,
    -value => ‘value’);
```

ან

```
print $query _ submit (‘button_name’, ‘value’);
```

submit ( ) ქმნის კითხვარის დამოწმების ღილაკს. ყოველ ფორმას უნდა ჰქონდეს ერთი ასეთი ღილაკი.

პარამეტრები:

პირველი არასავალდებულო არგუმენტია (-name), თქვენ შეგიძლიათ ღილაკს მისცეთ სახელები, თუ თქვენ გაქვთ სხვადასხვა დამოწმების ღილაკები და გსურთ განასხვაოთ ისინი ერთიმეორისაგან.

მეორე არასავალდებულო არგუმენტია (-value). იგი აძლევს ღილაკს მნიშვნელობას, რომელიც გადაეცემა სკრიპტს კითხვარის სტრიქონში. თქვენ შეგძლიათ გაარკვიოთ, რომელი ღილაკი იყო დაჭერილი თუ გამოიყენებთ სხვადასხვა მნიშვნელობებს განსხვავებული ღილაკებისათვის:

$which_one $query -> param (‘button_name’_;

Java სკრიპტების გამოყენება: radio_button ( ) სცნობს –onClick პარამეტრს.

აღდგენილი ღილაკის შექმნა

```
print $query- reset
```

reset ( ) ქმნის “reset” ღილაკს. შევნიშნოთ, რომ იგი აღადგენს ფორმას იმ სახით, როგორიც სკრიპტის ბოლო გამოძახებისას იყო.

გაჩუმების ღილაკის შექმნა

print $query ->default (‘button_label)

default ( ) ქმნის ღილაკს, რომელიც, როცა გამოიძახება, იწვევს ფორმის აღდგენას გაჩუმების მნიშვნელობებით.

ფარული ველის შექმნა

print $query -> hidden (-name =>’hidden_name’,
-default => [value 1’, ‘value 2’...);

hidden ( ) ქმნის ტექსტურ ველს, რომელსაც ვერ დაინახავს მომხმარებელი. იგი სასარგებლოა ცვლადი ინფორმაციის მდგომარეობის გადსაცემად სკრიპტის ერთი გამოძახებიდან მეორეზე.

პარამეტრები:

პირველი სავალდებულო არგუმენტია ველის სახელი (-nme).

მეორე სავალდებულო არგუმენტია მისი მნიშვნელობა (-default). თქვენ შეგიძლიათ გამოიყენოთ ერთი მნიშვნელობა ან სიაზე მითითება.

ველის მნიშვნელობის მიღება შეიძლება შემდეგნაირად

$hidden_value = $query ->param (‘hidden_name’);

თუ თქვენ გინდათ შეცვალოთ ველის მნიშვნელობა, გამოიყენეთ:

$query -> param (‘hidden_name’, ‘new’, ‘values’, ‘here’);

სერათზე დასაწკაპუნებელი ღილაკის შექმნა

print $query -> image_button (-name => ‘button_name’,
-src => ‘/ source / URL’,
-align => ‘MIDDLE’);

ან

print $query -> image_button (‘button_name‘, ‘/ source / URL‘, ‘MIDDLE‘);

image button ( ) ქმნის დასაწკაპუნებელ სურათს. როცა მასზე დავაწკაპუნებთ თქვენ სკრიპტს დაუბრუნდება “button_name. x “ და
“ button_name.y “, სადაც “button_name” არის ველის სახელი, რომელიც თქვენ დაარქვით სურათს.

Java სკრიპტების გამოყენება: cmage_button ( ) სცნობს –0nClick პარამეტრს.

პარამეტრები:

პირველი სავალდებულო არგუმენტია ველის სახელი.

მეორე სავალდებულო არგუმენტია URL

მესამე სავალდებულო არგუმენტია გასწორების ტიპი და შეიძლება იყოს TOP, BOTTOM და MIDDLE.

ღილაკის მნიშვნელობის მიღება შეიძლება $x - $query -> param (‘button_name.x’);
$y = $query -> param (‘button_name.y’);

Javascript მოქმედების ღილაკის შექმნა

```
print $query -> button (-name => 'button_name',
    -value => 'user_visible_label',
    -onClick => "do_some thing ( ) ");
```

ან

```
print $query - > button ('button_name', "do_some thing ( ) ");
```

button ( ) ქმნის ღილაკს, რომელიც თავსებადია Netscape 2.0-ის Java სკრიპტთან. როცა იგი დაჭერილია Java სკრიპტის კოდი მითითებულია –onChick პარამეტრზე და იქნება გამოთვლილი. სხვა ბროუზერებისათვის შეიძლება იგი არ მუშაობდეს.

Netscape პროცესების ინდიკატორები

Netscape ბროუზერები 1.1 და უფრო მაღალი ვერსიები ითვალისწინებენ პროცესების ინდიკატორების გამოყენებას ბროუზერებთან სიანსის დროს. CGI.PM აქვს რამოდენიმე მეთოდი პროცესორების ინდიკატორების მხარდასაჭერად.

პროცესის ინდიკატორი არის სახელი = მნიშვნელობა წყვილი, რომელიც ჰგავს CGI კითხვარის სახელიან პარამეტრებს. CGI სკრიპტები ჰქმნიან ერთ ან მეტ პროცესების ინდიკატორებს და აგზავნიან მათ ბროუზერებზე HTTP სათაურში. ბროუზერი ითვალისწინებს პროცესების ინდიკატორების სიას, რომელიც ეკუთვნის კონკრეტულ ვებ სერვერს უბრუნებს მათ CGI სკრიპტს სეანსის დროს. გარდა სავალდებულო სახელი = მნიშვნელობა წყვილისა, ყოველ პროცესის ინდიკატორს შეიძლება ჰქონდეს რამოდენიმე არასავალდებულო ატრიბუტები:

1. ვადის გასვლა.

   ეს არის დრო / თარიღი სახის სტრიქონი (CGM ფორმატში), რომელიც უჩვენებს პროცესის ინდიკატორის ვადის გასვლას. cookie შეინახება და დაუბრუნდება თქვენს სკრიპტს სანამ ეს ვადა არ გავა, როცა მომხმარებელი გამოდის Netscape-დან და ხელმეორედ უშვებს მას. თუ ვადა არაა მითითებული, cooke რჩება აქტიური, სანამ მომხმარებელი გამოვა Netscape-დან.
2. This არე არის ნაწილობრივი ან სრული, რომლისთვისაც cookie ვარგისია. ბროუზერი დაუბრუნებს cookie-ს ნებისმიერ მასპინძელს, რომელიც ეთანადება ან ნაწილობრივ არის სახელს. მაგალითად, თუ თქვენ მიუთითებთ არის სახელად ".carpicorn. com "-ს, მაშინ Netscape დაუბრუნებს cookie-ს web სერვერებს, რომლებიც გაშვებული არიან შემდეგ მანქანებიდან რომელიმეზე: " www. carpicorn. com ", "www2 . carpicorn. com ", " feckless. carpicorn. com " და ა.შ.. არის სახელები უნდა შეიცავდნენ ყველაზე მცირე ორ წერტილს, რომ ავიცილოთ მაღალ დონეებთან დამთხვევა, როგორიცაა ".edu". თუ ასე არაა მითითებული, მაშინ ბროუზერი დაუბრუნებს cookie-ს მხოლოდ იმ სერვერებს, საიდანაც cookie მოვიდა.
3. რაიმე გზა, თუ თქვენ მიუთითეთ cookie-ს მზა ატრიბუტი . ბროუზერი ამოწმებს მას თქვენი სკრიპტის URL-თან, სანამ cookie-ს დააბრუნებდეს. მაგალითად, თუ თქვენ მიუთითეთ " / CGI_bin " გზა, მაშინ cookie დაუბრუნდება ნებისმიერს შემდეგ სკრიპტებს: " / CGI_bin / tally.pl ", " / CGI_bin / order.pl " და "/ CGI_bin /codtomer.pl ", მაგრამ არ დაუბრუნდება „ / CGI_private / site adm.pl "-ს. გაჩუმებით გზა უდრის " / "-ს

4. “sequre”. თუ “secure” ატრიბუტი მოცემულია, მაშინ cookie გაეგზავნება მხოლოდ თქვენ სკრიპტს, თუ CGI-ს მოთხოვნა ხდება secure არხზე, როგორიცაა SSL. Netscape cookies ინტერფეისი არის cookie ( ) მეთოდი:

```
$cookie = $query -> cookie (-name => ‘sessinID’, -value => ‘xyzzy’,
        -expres => ‘+1h’, -path =>’ / CGI_bin / database’,
        -domain => ‘. carpicorn. org’, -secure => 1);
    print $query -> header (-cookie => ‘$cookie’);
```

cookie ( ) ქმნის ახალ cookie-ს. მისი პარამეტრებია:

-name

cookie- ს სახელი სავალდებულოა. იგი შეიძლება იყოს ნებისმიერი სტრიქონი. თუმცა Netscape ზღუდავს cookie-ს სახელებს. ისინი არ უნდა შეიცავდეს ხარვეზის ნიშანს. CGI.PM-ში მისი აცილება შეიძლება escape ნიშნით.

-value

cookie-ს მნიშვნელობაა. იგი შეიძლება იყოს სკალარი, მასივზე მითითება, ან ასოციატიურ მასივზე მითითება. მაგალითად, თქვენ შეგიძლიათ შეინახოთ მთელი ასოციატიური მასივი cookie-ში შემდეგნაირად:

```
$cookie = $query -> cookie (-name => ‘family information’,
    -value => \ % childrens_ages);
```

-path

ეს არის არასავალდებულო ნაწილობრივი გზა, რომლისთვისაც cookie ვარგა.

-domain

ეს არის არასავალდებულო ნაწილობრივი არე, რომლისთვისაც cookie ვარგა.

-expires

ეს არის არასავალდებულო ვადა cookie-სთვის. ფორმატი აღწერილია header ( ) მეთოდის სექციაში.

-secure

თუ დაყენებულია tru-ეზე, მაშინ cookie უნდა იყოს ჩადგმული HTTP სათაურში სტრიქონის შიგნით, რომელსაც აბრუნებს header ( ) მეთოდი:

```
print $query -> header (-cookie => ‘$my_cookie’);
```

რომ შექმნათ მრავალი cookie, მიეცით header ( ) მეთოდს მასივზე მითითება:

```
$cookie 1 = 4query -> cookie (-name => ‘riddle_name’,
        -value => “The Sphynx’s Question “);
$cookie 2 = $query -> cookie (-name => ‘ansvers’’,
        -value => \ % ansvers);
print $query -> header (-cookie => [$coocie 1, cookie 2]);
```

რომ მივიღოთ რაიმე cookie, მოითხოვეთ იგი სახელით cookie ( ) მეთოდის გამოძახებით -value პარამეტრის გარეშე:

```
use CGI;
$query = NEW CGI;
% ansvers = $query -> cookie (-name =>’ ansvers ‘);
```

cookie და სახელთა არე (namespace) არიან სხვადასხვა. თუ თქვენ გაქვთ პარამეტრი ‚ansvers’ და cookie ‘ansvers’, მნიშვნელობები მიღებული param(‘ansvers’) და cookie (‘ansvers’) არიან ერთიმეორისაგან დამოუკიდებელნი. თუმცა შეიძლება CGI პარამეტრი მიმართო cookie-საკენ და პირიქით:

```
$c = $q -> cookie (-name = > answers’, value =>[ param (‘answers’)]);
```

იხილეთ cookie.CGI მაგალითი, იმის საილუსტრაციოდ თუ როგორ გამოიყენოთ cookie ეფექტურად.
შენიშვნა: არსებობს ზოგიერთი არადოკუმენტირებელი შეზღუდვა cookie-ს გამოყენების მიმართ. ეს ეხება cookie-ს რაოდენობას და სიგრძეს.

Netscape-ს ფრეიმერებთან მუშაობა

შესაძლებელია CGI.PM სკრიპტმა გამოიტანოს ინფორმაცია ბროუზერების პანელებზე და ფანჯრებში Netscape-ს ფრეიმერების მექანიზმის გამოყენებით.
არსებობს ფრეიმერების გამოყენების სამი საშუალება:

1. <frameset> დოკუმენტის შექმნა
   HTTP სათაურის გამოტანის შემდეგ იმის ნაცვლად, რომ შექმნათ სტანდარტული HTML დოკუმენტი start_HTML ( ) მეთოდის გამოძახებით, შექმენით < FRAMESET > დოკუმენტი, რომელიც განსაზღვრავს ფრეიმერს გვერდზე. მიუთითეთ თქვენ სკრიპტს SCR-ი ყოველი ფრეიმერისათვის. არ არსებობს რაიმე სპეციფიკური საშუალება
   < FRAIMSET > სექციების შესაქმნელად CGI.PM-ში, მაგრამ HTML-ის დაწერა ძალიან იოლია. იხილეთ ფრეიმების დოკუმენტაცია Nestcape-ის Home page –ში:
   HTTP: // home. netscape. com / assist / net_sites / frames. HTML
2. HTTP სათაურში დოკუმენტის დანიშნულების ადგილის განსაზღვრა.
   თქვენ შეგიძლიათ გამოიყენოთ –target პარამეტრი header ( ) მეთოდში:
   print $q -> header (-target => 'Results Winword');
   ეს ეუბნება Netscape-ს ჩატვირთოს თქვენი სკრიპტის გამოსასვლელი „Results Window" სახელიან ფრეიმში. თუ ფრეიმი ასეთი სახელით არ არსებობს, მაშინ Netscape შექმნის ახალ ფანჯარას და ჩატვირთავს თქვენი სკრიპტით შექმნილ დოკუმენტს მასში. არსებობს მთელი რიგი მაგიური სახელებისა, რომლებიც თქვენ შეგიძლიათ გამოიყენოთ –target-ისათვის.
3. დანიშნულების ადგილის მითითება < FORM > ჭდეში. თქვენ შეგიძლიათ მიუთითოთ ფრეიმი, რომელიც უნდა ჩაიტვირთოს < FORM > ჭდეში. ეს გამოიყენება ასე:
   print $q -> start form (-target = > 'Results Winword');
   როცა თქვენი სკრიპტი ხელმეორედ გამოიძახება ფორმის მიერ, მისი გამოსასვლელი ჩაიტვირთება Results Winword" ფრეიმში. თუ ასეთი ფრეიმი არ არსებობს ახალი ფანჯარა შეიქმნება. " frameset. CGI " სკრიპტი, რომელიც მოთავსებულია მაგალითების დირექტორიაში გვიჩვენებს გვერდების შექმნის ერთ-ერთ ხერხს, რომელშიც შესავსები ფორმა და პასუხები მოთავსებული არიან ერთიმეორის გვერდით მყოფ ფრეიმებში.

კასკადური სტილის ფურცლების შექმნის შეზღუდული მხარდაჭერა

CGI.PM-ში გათვალისწინებულია კასკადური სტილის ფურცლების (css) შეზღუდული სახის მხარდაჭერა HTML 3-თვის. რომ ჩადგათ სტილური ფურცელი თქვენს დოკუმენტში, გამოიყენეთ Start_HTML ( ) მეთოდი –style პარამეტრით. ამ პარამეტრის მნიშვნელობა შეიძლება იყოს სკალარი. ასეთ შემთხვევაში იგი ჩაიდგმება < STILE > სექციაში, ან შეიძლება იყოს ხეშ მასივზე მითითება.

უკანასკნელ შემთხვევაში თქვენ უნდა გამოიყენოთ ხეშ მასივი ერთი ან მეტი –src-თი ან –code-ით. –scr მიუთითებს URL-ს სტილური ფურცელი სად შეიძლება იყოს. –code მიუთითებს სკალარი მნიშვნელობა სად შეიძლება იყოს ჩადგმული < STILE > სექციაში. სტილის განსაზღვრებები –code-ში გადაიფარებიან –code-ში ისევე, როგორც –named გადაიფარება –src-ში. აქედან გამომდინარეობს სახელი კასკადირება. თქვენ შეგიძლიათ მიუთითოთ სტილური ფურცლის ტიპი არასავალდებულო –type პარამეტრის დამატებით ხეშზე, რომელიც განსაზღვრულია –style-ით. თუ არაა მითითებული, მაშინ გაჩუმებით სტილი იქნება ,text / css! რომ მიუთითოთ სტილი თქვენი დოკუმენტის ტანში, დაუმატეთ –class პარამეტრი რომელიმე **HTML** ელემენტს:

print h1 ({-class => 'Fancy '}, 'Welcome to the Party '); ან განსაზღვრეთ სტილები –style პარამეტრით:

```
print span ({-style =>'Color : red;'},
        h1 ('Welcome to Hell');
```

თქვენ შეგიძლიათ, აგრეთვე, გამოიყენოთ ახალი span ( ) ელემენტი, რომ გამოიყენოთ რაიმე სტილი ტექსტის სექციაზე:

```
print span ({-style = > 'Color : red;'}
        h1 ('Welcome to Hell'),
    "Where did that hand basket get to?");
```

შევნიშნოთ, რომ თქვენ უნდა შემოიტანოთ " : **HTML**3 " განსაზღვრებები, რომ შეგეძლოთ span ( ) მეთოდის გამოყენება. იხილეთ CSS სპეციფიკაციები **HTTP** : || www . w3 . org / pub / WWW / TR / WD-css-1. **HTML** მისამართით. ქვემოთ მოცემულია CSS-ის გამოყენების მაგალითი:

```
use CGI  qw / : standard : HTML3 /;
# აქ მოცემულია გვერდში ჩადგმული სტილიანი ფურცელი
$new Style = << END;
< ! - -
p. Tip { margin –right : 50 pt;
        margin-left : 50 pt;
        color : red; }
p. Alert { font-size : 30pt;
          font-family : sans-serif;
          color : red; }
            - - >
            END
print header ( );
print start_HTML (-title = > ,CGI with Style',
   -style = > { -src = >'HTTP : || www. carpicorn . com / style /
                                        st1 . CSS',
  -code = >  $new Style} );
  print h1(,CGI with Style'),
       p ( {-class = > ,Tip'},
  "Bether read the cascading style sheet aspect before playing with this !
  "),
  Span ( {-style = > ,color : magenta'},
```

```
"Look mom, no hands ! ",
    p ( ), "Whoo wee ! ");
print en HTML;
```

გამართვა

თუ თქვენ გაუშვებთ სკრიპტს ბრძანებით ან Perl-ის გამმართველით თქვენ შეგიძლიათ გაატაროთ სკრიპტი გასაღები სიტყვების სიით ან შეტანიდან (თქვენ არ უნდა იფიქროთ იმაზე, რომ თქვენ სკრიპტს შეუძლია წაიკითხოს გარემოს ცვლადებიდან). თქვენ შეგიძლიათ გაატაროთ გასაღები სიტყვები შემდეგნაირად:

```
your_skript . pl keyword1 keyword2 keyword3
```

ან

```
your_skript . pl keyword1 + keyword2 + keyword3
```

ან

```
your_script .pl name1 = value1 name2 = value2
```

ან

```
your_script . pl name1 = value1 & name2 = value2
```

ან კიდევ ახალ სტრიქონზე გადასვლით ყოველი პარამეტრისათვის სტანდარტული შეტანიდან.

გამართვის დროს თქვენ შეგიძლიათ გამოიყენოთ ბრჭყალები ან \ სიმბოლო ზოგიერთი აკრძალული სიმბოლოსათვის თქვენს პარამეტრი = მნიშვნელობა წყვილში:

```
your_script . pl "name1 = I am a long value" ' "name2 = two \ words "
```

ყველა Name / value წყვილების გამოტანა

dump ( ) მეთოდი ქმნის სტრიქონს, რომელიც შედგება კითხვარის სახელი / მნიშვნელობა წყვილისაგან, რომელიც კარგად ფორმატირებულია, როგორც სიაში ჩადგმული სია. ეს სასარგებლოა გამართვის დროს:

```
print $query - > dump
```

მივიღებთ შემდეგს:

```
< UL >
< LI > name 1
< UL >
< LI > value 1
< LI > value 2
< / UL >
< LI > name 2
< UL >
< LI > value 1
< / UL >
< / UL >
```

თქვენ შეგიძლიათ გადასცეთ 'true' მნიშვნელობა dump ( )-ს იმისათვის, რომ დაბეჭდოთ შედეგები როგორც სრული ტესტი, რომელიც ჩადგმულია <PRE > სექციაში.

როგორც შემოკლება, 1.56 ვერსიაში, თქვენ შეგიძლიათ მთელი **CGI** ობიექტის ინტერპოლირება რაიმე სტრიქონში და იგი შეიცვლება მშვენიერი **HTML** გამოტანით როგორც ნაჩვენებია ზემოთ:

```
$query = new CGI;
print "< H2 > Current Values < / h2 > $query \ n " ;
```

ლექცია 15

გარემოს ცვლადების მოტანა

ზოგიერთი ყველაზე უფრო სასარგებლო გარემოს ცვლადები შეიძლება შემოტანილ იქნეს ამ ინტერფეისში. ესენი არიან შემდეგი მეთოდები:

accept ( )

აბრუნებს MIME ტიპის სიას, რომელსაც შორეული ბროუზერი ითვალისწინებს. თუ თქვენ გამოიყენებთ ამ მეთოდს არგუმენტით, რომელიც შეესატყვისება MIME ტიპს, როგორც ეს მოცემულია

& query ->accept (' text / **HTML** '), იგი დააბრუნებს მოძრავ წერტილიან მნიშვნელობას, რომელიც შეესატყვისება ბროუზერის არჩევანს ამ ტიპისათვის დაწყებული 0.0 –დან 1.0-მდე. Glob ტიპები (ე.ი. text / *) ბროუზერის გათვალისწინებულ სიაში დამუშავდება სწორად.

raw_cookie ( )

აბრუნებს **HTTP**_cookie ცვლადს. პროცესის ინდიკატორებს cookie-ებს აქვთ სპეციალური ფორმატი და ამ მეთოდის გამოძახება აბრუნებს დაუმუშავებელ ფორმას. იხილეთ cookie ( ).

უპარამეტრებოდ გამოძახებული raw_cookie ( ) აბრუნებს შეფუთულ პროცესის ინდიკატორის cookie სტრუქტურას. თქვენ შეგიძლიათ დაყოთ იგი ცალკეულ პროცესის ინდიკატორის cookie-ბად " ; "-ით. პროცესის ინდიკატორის cookie—ს სახელით გამოძახებული აბრუნებს უ-escape-ო cookie-ს ფორმას. სახელების მისაღებად თქვენ შეგიძლიათ გამოიყენოთ cookie ( ) მეთოდი ან გამოიყენოთ raw_fetch ( ) მეთოდი **CGI** : : Cookie მოდულიდან.

user_agent ( )

აბრუნებს **HTTP**_USER_AGENT ცვლადს. თუ თქვენ გამოიყენებთ არგუმენტს, იგი შეეცდება მოცემული ნიმუში დაამთხვიოს მას:

```
$query -> user_agent (netscape)
```

path_info ( )

აბრუნებს გზის დამატებით ინფორმაციას სკრიპტის URL-დან. ე.ი. / **CGI**-bin / your_script / additional / stuff დააბრუნებს

$query -> path_info ( )-ში "additional / stuff "-ს.

შენიშვნა: Microsoft-ის ინტერნეტ ინფორმაციული სერვერი არ იძლევა გზის დამატებით ინფორმაციას.

patch_translated ( )

იგი აბრუნებს გზის დამატებით ინფორმაციას გადაყვანილს ფიზიკურ გზაში, ე. ი. "/ usr / local etc / **HTTP**d / htdocs / additional / stuft "

unix ოპერატიული სისტემისათვის. Mikrosoft IIS არ იძლევა ასეთ ინფორმაციას.

remote_host ( )

აბრუნებს შორეული მასპინძლის სახელს ან IP მისამართს.

script_name ( ) აბრუნებს სკრიპტის სახელს როგორც ნაწილობრივ URL-s თვითმითითებადი სკრიპტისათვის.

referer ( )

აბრუნებს გვერდის URL-ს, რომელსაც მიმოიხილავდა ბროუზერი თქვენი სკრიპტის მოტანამდე. არ გამოიყენება არცერთ ბროუზერზე.

auth_type ( )

აბრუნებს ნებართვა / შემოწმების მეთოდი გამოიყენება სკრიპტებისათვის თუ არა.

srever_name ( )

აბრუნებს სერვერის სახელს, ჩვეულებრის ეს არის მასპინძელი კომპიუტერის სახელი.

vitrual_host ( )

როცა გამოიყენება ვირტუალური მასპინძლები, მაშინ აბრუნებს მასპინძლის სახელს, რომელთან დაკავშირებასაც ცდილობდა ბროუზერი.

server_softvare ( )

აბრუნებს სერვერის პროგრამული უზრუნველყოფის სახელს და ვერსიის ნომერს.

remote_user ( )

აბრუნებს ნებართვა (შემოწმების სახელს, რომელიც გამოიყენება მომხმარებლის შესამოწმებლად, თუ ეს სკრიპტი დაცულია.

user_name ( )

ცდილობს მოიპოვოს შორეული მომხმარებლის სახელი სხვადასხვა ტექნიკის გამოყენებით. ეს გამოიყენება ძველ ბროუზერში როგორიცაა Mosaic. Netscape არ იძლევა მოხერხებულად მომხმარებლის სახელს.

request_method ( )

აბრუნებს თქვენ სკრიპტთან მიმართვის მეთოდს. ჩვეულებრივ ესაა 'POST', 'GET', ან 'HEAD'.

NPH სკრიპტების გამოყენება

NPH ან ' no-parsed-header ' სკრიპტები გვერდს აუვლიან სერვერს და გადასცემენ HTTP სათაურს პირდაპირ ბროუზერს. ამას აქვს უპირატესობა, როცა ვიღებთ HTTP გაფართოებებს, რომლებიც არ არიან გათვალისწინებული თქვენი სერვერის მიერ, როგორიცაა server push და PICS სათაურები.

სერვერები იყენებენ სხვადასხვა შეთანხმებებს CGI სკრიპტების აღსანიშნავად, როგორც NPH-ები. მრავალი UNIX სერვერები ამოწმებენ სკრიპტის სახელი იწყება თუ არა პრეფიქსით " nph- "

CGI.PM ითვალისწინებს NPH სკრიპტებს სპეციალურ NPH რეჟიმით. როცა ამ რეჟიმშია CGI.PM გამოიტანს აუცილებელ ექსტრა სათაურის ინფორმაციას header ( ) და redirect ( ) მეთოდების გამოყენებისას. CGI.PM ავტომატურად აღმოაჩენს, როცა სკრიპტი გაშვებულია IIS-ში და თვითონ გადავა ამ რეჟიმში. ასე, რომ თქვენ არ გჭირდებათ ამ რეჟიმის დაყენება, თუმცა ამით არაფერი დაშავდება. არსებობს სხვადასხვა ხერხები CGI.PM-ის NPH რაჟიმში გადასაყვანად: use ინსტრუქციაში დაუმატეთ "-nph " პროგრამა შემოსატანი სიმბოლოების სიას:

use CGI qw (:standard –nph)

nph ( ) მეთოდის გამოძახებით:

გამოიძახეთ nph ( ) მეთოდი რაიმე პარამეტრით პროგრამის ნებისმიერ წერტილში CGI.PM გამოყენების შემდეგ.

CGI -> nph (1)

-nph პარამეტრის გამოყენებით header ( ) და redirect ( ) ინსტრუქციებში:

print $q -> header (-nph => 1);

Server Push

CGI.PM ითვალისწინებს სამ მარტივ ფუნქციას იმ ტიპის მრავალნაწილიანი ოკუმენტების შესაქმნელად, რომლებიც რეალიზაციას უკეთებენ server pugh-ს. ეს ფუნქციები შექმნილია Ed Jordan-ის მიერ – მისამართი < ed@fidalgo.net >. რომ შემოიტანოთ ისინი საჭიროა " : push"-ის დაყენება. გირჩევთ, აგრეთვე, დასვათ სკრიპტი NPH რეჟიმში და მიანიჭოთ $I-ს ერთიანი, რომ აიცილოთ ბუფერიზაციის პრობლემები.ქვემოთ

მოყვანილია მარტივი სკრიპტები, რომელიც გვიჩვენებს server push-:

```
# ! / usr / local / bin / perl
 use CGI qw / : push –nph /;
 $   = 1;
print miltypart_init (   boundary => ' - - - - - here we go ! ');
                while (1) {
print multipart_start (~ type => 'texi / plain '),
    "The current time is ", scalar (localtime), " in ",
    multiport_end;
     sleep 1: }
```

ეს სკრიპტი ინიციალიზაციას უკეთებს server push-ს multipart_unit ( ) –ის გამოძახებით. მას შემოაქვს უსასრულო ციკლი, რომელშიც იგი იწყებს ახალ მრავალწლიან სექციას multipart_start ( )-ის გამოძახებით, ბეჭდავს ადგილობრივ დროს და ამთავრებს მრავალწლიან სექციას, შემდეგ იგი მიიძინებს და ისევ იწყებს თავიდან.

multipart_init ( ) multipart_init (-boundary => $boundary);

ინიციალიზაციას უკეთებს მრავალწლიან სისტემას. –boundary არგუმენტი უთითებს MINE საზღვრის სტრიქონს, რომელიც გამოიყენება დოკუმენტის ნაწილების გამოსაყოფად ერთიმეორისაგან. თუ არაა მითითებული CGI.PM თვითონ აირჩევს მიზანშეწონილ საზღვარს.

multypart_start (-type => $type)

იწყებს მრავალწლიანი დოკუმენტის ახალ ნაწილს მითითებული MINE ტიპის გამოყენებით. თუ არაა მითითებული იგულისხმება text / HTML.

multypart_end ( ) ამთავრებს ნაწილს.

მომხმარებლებმა, რომელთაც აინტერესებთ server puch-ის გამოყენება, უნდა იხილონ CGI : : push მოდული.

მომსახურების შემოტევების აკრძალვის აცილება

CGI.PM-ში პოტენციალური პრობლემაა, რომ არაა შეზღუდული გასგზავნი ფორმის მოცულობა. ჰაკერს შეუძლია შეუტიოს თქვენს საიტს იმით, რომ გააგზავნოს მთელი post რაიმე ცვლადში, რომელიც გაიზრდება მოცულობაში, სანამ არ გამოვა მეხსიერების გარეთ. ეს გამოიწვევს სისტემის ავარიულ გამორთვას. ეს

არის მომსახურების შემოტევის აკრძალვის ერთ-ერთი ფორმა. სხვა შესაძლებელი ფორმაა შორეულმა მომხმარებელმა აიძულოს **CGI.PM** მიიღოს ჩასატვირთად დიდი ფაილი და შეინახოს დროებით დირექტორიაში იმ შემთხვევაშიც კი, როცა თქვენი სკრიპტი არ ელოდება რაიმე ფაილის ჩატვირთვას. **CGI.PM** შეუძლია ამოაგდოს ეს ფაილი ავტომატურად, როცა ჩაიტვირთება იგი, მაგრამ მანამდე შორეულ მომხმარებელს შეუძლია გადაავსოს დისკის მეხსიერება და ამით შეუქმნას პრობლემები სხვა პროგრამებს. ამის აცილების საუკეთესო ხერხია შეზღუდული მეხსიერების მოცულობა, მანქანური დრო და დისკის მოცულობა, რომელთა გამოყენება შეეძლება **CGI** სკრიპტს. ზოგიერთ Web სერვერებს თვითონ აქვთ ამის აცილების საშუალებები. სხვა შემთხვევაში თქვენ შეგიძლიათ გამოიყენოთ shell limit ან ulimit ბრძანებები, რომ დააწესოთ **CGI** რესურსების გამოყენების მაქსიმალური საზღვარი.

CGI.PM-ს აგრეთვე აქვს საკუთარი დაცვა მომსახურების შემოტევის აკრძალვისაგან, მაგრამ თქვენ უნდა წინასწარ გაააქტიუროთ იგი. ამისათვის არსებობს ორი გლობალური ცვლადი სახელთა არეში:

```
$ CGI : : POST_MAX
```

ამ ცვლადის მნიშვნელობა არის მაქსიმალური ბაიტების რაოდენობა, რომელიც შეიძლება გამოყენებულ იქნეს POST-ის დროს. **CGI.PM** როგორც კი აღმოაჩენს, რომ POST არის უფრო დიდი, მაშინვე გამოიტანს შეცდომის შეტყობინებას და დაამთავრებს მუშაობას. მისი გამოყენება შეიძლება როგორც ჩვეულებრივი ასევე მრავალწლიანი POST-ის დროს. ჩვეულებრივ ამ ცვლადს ანიჭებენ ერთ მეგაბაიტს.

```
$ CGI : : DISABLF_UPLOADS
```

თუ მას აქვს არანულოვანი მნიშვნელობა, მაშინ სრულად აიკრძალება ფაილის ჩატვირთვები. სხვა შესავსები ფორმები იმუშავებენ ჩვეულებრივად. თქვენ შეგიძლიათ გამოიყენოთ ეს ცვლადები შემდეგი ორი ხერხიდან ერთ-ერთის საშუალებით:

1. სკრიპტიდან სკრიპტზე გადასვლისას:

დააყენეთ ეს ცვლადი სკრიპტის თავში “use“ ინსტრუქციის შემდეგ.

```
use CGI qw / : standard / ;
use CGI : : Carp ‘fatals To Browser‘;
$ CGI : : POST_MAX = 1024 * 100;
$ CGI : : DISABLE_UPLOADS = 1;
```

2. ერთდროულად ყველა სკრიპტისათვის:

გახსენით **CGI.PM**, იპოვეთ $POST_MAX და $DISABLE_UPLOADS და დააყენეთ ისინი სასურველ მნიშვნელობებზე. თქვენ იპოვით მათ ფაილის თავში ქვეპროგრამაში სახელით initialize_globals ( ). როგორც კი იქნება მცდელობა გააგზავნონ POST უფრო დიდი ვიდრე $ POST_MAX-ის მნიშვნელობაა, იგი გამოიწვევს ფატალურ შეცდომას. თქვენ შეგიძლიათ გამოიყენოთ **CGI** : : Carp, რომ გამოიტანოთ ბროუზერის ფანჯარაში შეცდომის შეტყობინება, როგორც ეს ზემოთა მაგალითშია ნაჩვენები. სხვა შემთხვევაში შორეული მომხმარებელი დაინახავს მხოლოდ “ Internal Server” შეცდომის შეტყობინებას. იხილეთ Garp სახელმძღვანელო.

CGI-LIB.PL-თან თავსებადობა

არსებობს “Read Parse” ქვეპროგრამა იმისათვის, რომ იოლად შემოვიტანოთ პროგრამები CGI-lib.pl-დან:

use CGI; CGI : : ReadParse print “The value of the antique is $in {antique} . \ n “;

CGI.PM – ის ReadParse ( ) ქვეპროგრამა შექმნის ცვლადს სახელით % in, რომელსაც შეგიძლიათ მიმართოთ, რომ მოიპოვოთ კითხვარის ცვლადები. ReadPars-ის მსგავსად თქვენ შეგიძლიათ შექმნათ თქვენი საკუთარი ცვლადი. ReadParse –ის იშვიათად გამოყენებადი თვისებები, როგორიცაა @in და $in-ის შექმნა შეიძლება არ იყოს გათვალისწინებული. როცა გამოიყენებთ ReadParse-ს, შემდეგ შეგიძლიათ მიიღოთ თვით კითხვარის ობიექტი შემდეგნაირად:

```
$ = $in {CGI};
print $q ->textfield (-name =>‘wow‘,
   -value => ‘does this really work ?‘);
```

ეს საშუალებას მოგცემთ გაუშვათ CGI.PM-ის უფრო საინტერესო თვისებები თქვენი ძველი სკრიპტების ხელმეორედ გადაწერის გარეშე.

მარტივ ფორმაზე დაფუძნებული სკრიპტის
სრული მაგალითი

```
# ! / usr / local / bin /perl
use CGI;
        ,
        ,
        ,
        ,
        ,
        ,
        ,
```

შეცდომები

ეს მოდული გაიზარდა ფართოდ და მონოლითურად. აქედან გამომდინარე იგი აკეთებს ბევრ რამეს, როგორიცაა: URL-ების დამუშავება, CGI შეტანის გარჩევა, HTML-ის დაწერა და ა. შ., რაც არის აგრეთვე გაკეთებული LWP მოდულში. იგი უნდა იყოს ამოგდებული CGI : : * მოდულების სასარგებლოდ, მაგრამ როგორღაც მე მაინც ვაგრძელებ მათზე მუშაობას. შევნიშნოთ, რომ ეს კოდი დამახინჯებულია იმისათვის, რომ ავიცილოთ შეცდომის ყალბი შეტყობინებები, როცა პროგრამები გაშვებულია

-w გადამრთველით.

იხილეთ აგრეთვე

Carp, URL, Request, MiniSvr, Base, Form, Apache, Switch, Push, Fast.