PWSZ TARNÓW 2009

PROLOG I HTTP

Paweł Gołąb

Spis treści

1.Wstęp

1.1. Wymagana wiedza

Do prawidłowego zrozumienia tematu potrzebna będzie wiedza z następujących zagadnień:
-znajomość języka Prolog
-podstawy PHP
-podstawy HTML

1.2. Protokół HTTP

HTTP (ang. *Hypertext Transfer Protocol* – protokół przesyłania dokumentów hipertekstowych) to protokół sieci WWW. Obecną definicję HTTP stanowi RFC 2616. Za pomocą protokołu HTTP przesyła się żądania udostępnienia dokumentów WWW i informacje o kliknięciu odnośnika oraz informacje z formularzy. Zadaniem stron WWW jest publikowanie informacji – natomiast protokół HTTP właśnie to umożliwia.

Protokół HTTP jest tak użyteczny, ponieważ udostępnia znormalizowany sposób komunikowania się komputerów ze sobą. Określa on formę żądań klienta dotyczących danych oraz formę odpowiedzi serwera na te żądania. Jest zaliczany do protokołów bezstanowych (ang. *stateless*) z racji tego, że nie zachowuje żadnych informacji o poprzednich transakcjach z klientem (po zakończeniu transakcji wszystko "przepada"). Pozwala to znacznie zmniejszyć obciążenie serwera, jednak jest kłopotliwe w sytuacji, gdy np. trzeba zapamiętać konkretny stan dla użytkownika, który wcześniej łączył się już z serwerem. Najczęstszym rozwiązaniem tego problemu jest wprowadzenie mechanizmu ciasteczek. Inne podejścia to m.in. sesje po stronie serwera, ukryte parametry (gdy aktualna strona zawiera formularz) oraz parametry umieszczone w URL-u (jak np. /index.php?userid=3).

1.3. Po co obsługa http w Prologu?

Główne przyczyny zaimplementowania biblioteki http w Prologu:
-Internet to przyszłość, a Prolog ma być językiem przyszłości
-programy potrzebują dane, a Internet to największy zbiór danych
-Prolog pozwala na szybsze wnioskowanie, przy mniejszym wykorzystaniu zasobów, co jest niezwykle ważne w dużych aplikacjach internetowych

1.4. Przygotowanie środowiska

Zestaw bibliotek http jest częścią integralną środowiska SWI-Prolog, co oznacza, że nie jest wymagana dodatkowa instalacja. Korzystanie z tych bibliotek wymaga jedynie wskazanie paczki z której będziemy korzystać np:

:- use_module(library(http/http_client)).

Oznacza to, że będziemy korzystać z biblioteki http_client.

2.Prolog jako klient

2.1.Podstawowe predykaty

http_get(+*URL, -Reply, +Options*) – wysyła żądanie na podany adres, i odbiera odpowiedz.

Dodatkowymi opcjami mogą być:

connection(*ConnectionType*) – Jeśli ‘close’ (domyślnie) dla każdego żądania tworzone jest nowe połączenie. Jeśli ‘Keep-Alive’ szukane jest otwarte połączenie z żądanym hostem .

http_version(*Major-Minor*) – predykat do określenia wersji protokołu http, której będziemy używać. Domyślenie jest to 1.1.

timeout(+*Timeout*) – określa czas oczekiwania na dane z hosta. Domyślnie czeka w „nieskończoność”.

2.2. Dane z serwera http na wyjście strumienia Prologa

```
:- use_module(library(http/http_client)).            (1)
open(URL):-
      http_get(URL, In, []),                         (2)
      copy_stream_data(In, user_output),             (3)
      close(In).                                (4)
```

1-załanczamy bibliotekę,
2-wysyłamy żądanie na podany adres (URL),
3-kopiujemy dane przysłane z serwera http i wrzucamy je na wyjście
4-zamykamy strumień

3.Prolog jako serwer

3.1. Możliwe zastosowania, informacje wstępne

W czasach, gdy aplikacje internetowe zdobywają coraz większe grono użytkowników ważne jest by nasz kod mógł z powodzeniem być wykonywalny na serwerze http. Środowisko SWI-Prolog zostało wyposażone w narzędzia umożliwiające odbieranie żądania z zewnątrz, przetwarzanie oraz wysyłanie odpowiedzi.
Z pomocą Prologa możemy generować proste strony www jak i bardziej skomplikowane wykorzystujące arkusze stylów CSS oraz język skryptowy JavaScript.

Funkcjonalność serwera powinna być zawarta w jednym pliku Prologa (oczywiście z tego pliku możemy dołączać inne). Są możliwe trzy instalacje servera:

- library(thread_httpd)-wielowątkowy serwer.
- library(xpce_httpd) – serwer sterowany zdarzeniami
- library(inetd_httpd) - server-per-client- inicjalizowanie serwera dla każdego połączenia. Zalecane dla serwerów z niskim start-up'em.

3.2. Podstawowe predykaty

Wszystkie interfejsy serwerów umożliwiają stosowanie predykatu **http_server(:Goal, +Options)** do uruchomienia serwera.
Predykat ten przyjmuje następujące opcje:

port(*?Port*)-numer portu z którego będziemy korzystać podczas połączenia.

workers(*+N*) –definiuje liczbę wątków uruchamianych w jednym procesie.

timeout(*+SecondsOrInfinite*)- określa limit czasu odpowiedzi. Jeśli nieokreślony klient czeka na odpowiedź w „nieskończoność".

keep_alive_timeout(*+SecondsOrInfinite*) -czas, jaki klient chce zarezerwować do następnego zapytania w przypadku połączenia *Keep-Alive*. *Domyślnie 5 sekund*

Predykaty używane do obsługi żądania i odpowiedzi:

http_handler(*+Location, :Closure, +Options*)- dla lokalizacji wskazanej w żądaniu wskazuje predykat odpowiedzialny za obsługe.

http_dispatch(*+Request*) –wysyła żądanie do predykatu zdefiniowanego w http_handler. Predykat ten wywoływany jest z http_serwer.

http_delete_handler(*+Path*) – usuwa wskazanie do ścieżki.

reply_html_page(*+HEAD, +HTML*)- generuje odpowiedź w formie dokumentu HTML(odpowiedzią może być również pliki w formatach XML, JSON).

format(*+HTML*)- generuje odpowiedź serwera

3.3. Przykład działania serwera

```
:- use_module(library('http/thread_httpd')).
      //załączenie bibliotek
:- use_module(library('http/http_dispatch')).
:- use_module(library('http/html_write')).

server(Port) :-
    http_server(http_dispatch, [port(Port)]).        //start serwera na
                                                     wybranym porcie

:- http_handler('/', root, []).                      //mapowanie czyli
:- http_handler('/hello/world', hello_world, []).    przyporządkowanie
:- http_handler('/hello/link1', link1, []).          adresowi odpowiedzi


root(_Request) :-
        reply_html_page([ title('Prezentacja PWSZ 2009')     //sekcja HEAD
                        ],
                        [ p(a(href('hello/world'), hello)),  //sekcja HTML
                    p(a(href('hello/link1'), link_tekst))
                        ]).

hello_world(_Request) :-
        reply_html_page([ title('Zdziwiony?')
                        ],
                        [ h1('Co Ty tu robisz?'),
                          p('Nie masz innych zajęć?!:)')
                        ]).
link1(_Request) :-
        reply_html_page([ title('Kolejna stronka')
                        ],
                        [ h1('Cześć'),
                          p('Trochę tekstu!:)')
                        ]).
```

Opis kodu:
Dla żądania katalogu głównego generowana jest strona z tytułem w head'erze i dwoma linkami do innym ścieżek. Dla tych ścieżek mapowane są predykaty które generują odpowiedź dla linków w katalogu głównym. Wszystkie strony generowane są predykatem `reply_html_page`.

3.4. Formularz i przetwarzanie żądania

Jedną z ważniejszych rzeczy w protokole http jest możliwość przesyłania danych w żądaniu. Dane te mogą pochodzić np z formularz i być przesyłane metodą GET lub POST. W bibliotece `library(http/http_parameters)` został zaimplementowany predykat o nazwie `http_parameters` służący do odbieranu danych przesyłanych tymi metodami.

```
:- use_module(library('http/thread_httpd')).
:- use_module(library('http/http_dispatch')).
:- use_module(library('http/html_write')).
:- use_module(library(http/http_parameters)).

server(Port) :-
        http_server(http_dispatch, [port(Port)]).

:- http_handler('/', root, []).
:- http_handler('/hello/proba', proba, []).
root(_Request) :-
      format('Content-type: text/html~n~n', []),
      format('<html>~n', []),
      format('<form name="form" method="GET" action="/hello/proba">~n'),
      format('Imie<input type="text" name="name"/><br/>~n'),
      format('Wiek<input type="text" name="age"/><br/>~n'),
      format('<input type="submit" name="wyslij" value="Wyślij"/>~n'),
      format('~n</form>~n'),
        format('</html>~n', []).
proba(_Request) :-
      http_parameters(_Request,
                        [ name(Name, [ length >= 2 ]),
                          age(Age, [integer])
                        ]),
      reply_html_page([ title('Prezentacja PWSZ 2009')
                        ],
                        [ p(Age),
                    p(Name),
                    p(a(href('/'), wroc))
                        ]).
```

Opis kodu

Dla odpowiedzi na wywołanie katalogu głównego tworzymy formularz używając predykatu `format`. Obsługa wysłanych danych nastepuje w predykacie „proba”. Za pomocą predykatu `http_parameters` odczytujemy przesłane dane. Musza one spełniać uwzględnione warunki. Odpowiedź tworzymy predykatem `reply_html_page`.

3.5. Przeglądarka internetowa jako interfejs dla aplikacji

W kolejnym przykładzie pokazany został fakt, że przeglądarka internetowa może pełnić rolę interfejsu dla aplikacji Prologa, dzięki czemu stają się one bardziej przyjazne dla użytkownika.

Program wyliczający procent składany

```
%%% Procent składany
:- use_module(library('http/thread_httpd')).
:- use_module(library('http/http_dispatch')).
:- use_module(library('http/html_write')).
:- use_module(library(http/http_parameters)).

server(Port) :-
        http_server(http_dispatch, [port(Port)]).

:- http_handler('/', root, []).
:- http_handler('/hello/proba', proba, []).
root(_Request) :-
      format('Content-type: text/html~n~n', []),
      format('<html>~n', []),
      format('<form name="form" method="GET" action="/hello/proba">~n'),
      format('Kwota<input type="text" name="kwota"/><br/>~n'),
      format('Procent<input type="text" name="procent"/><br/>~n'),
      format('Lata<input type="text" name="lata"/><br/>~n'),
      format('<input type="submit" name="wyslij" value="Wyślij"/>~n'),
      format('~n</form>~n'),
        format('</html>~n', []).
proba(_Request) :-
      http_parameters(_Request,
                        [ kwota(Kwota, [integer]),       %zczytanie
                          procent(Procent, [integer]),   %parametrów
                          lata(Lata, [integer])
                        ]),
      efekt(Kwota, Procent, Lata, Suma),                 %predykat liczący
      reply_html_page([ title('Prezentacja PWSZ 2009')   %sumę
                        ],
                        [ p('Suma wynosi:'),
                    p(Suma),                             %wypisanie sumy
                    p(a(href('/'), wroc))
                        ]).


efekt(Kwota,_,0,Suma):-
    Suma = Kwota,!.
efekt(Kwota,Procent,Lata,Suma):-
    Lata > 0,
    Lata1 is Lata-1,
    efekt(Kwota,Procent,Lata1,Suma1),
    Suma is Suma1+Suma1*Procent/100.
dubel(Procent,Lata,Suma):-
    generuj(Lata),
    efekt(1,Procent,Lata,Suma),
    Suma>2,!.

generuj(1).
generuj(I):-generuj(I1), I is I1+1.
```

4. Rozwiązania pośrednie

4.1. Sens stosowania Prologa jako serwera http

Decydując się na wykorzystanie Prologa jako serwera naszych aplikacji stajemy przed pytaniem czy ma to sens i na ile będzie nam się to opłacać. Nikt też na pewno nie będzie przerabiał istniejącego już serwisu (opartego o skrypt PHP lub inny) na język Prologa. Dlatego najlepszym rozwiązaniem jest umieszczenie na jednym serwerze obsługi Prologa i innych języków lub wywoływanie żądań na innym serwerze i interpretacja odpowiedzi. Kolejny przykład pokaże taką interakcję Prologa i PHP.

4.2. PHP i Prolog-przykład

Kod PHP

```
<?php
echo "<form name='submit_form' method='post' action='example.php'>";
echo "kwota<input type='text' name='kwota' style='width:300px;' /><br>";
echo "Procent<input type='text' name='procent' style='width:300px;'
/><br>";
echo "Lata<input type='text' name='lata' style='width:300px;' /><br>";
echo "<input type='submit' name='submit' value='Send'/>";
echo "</form>";

if (isset($_POST['submit'])) {

      if (isset($_POST['kwota'])!=false && isset($_POST['procent'])!=false
&& isset($_POST['lata'])!=false){

      $file=implode('',file("http://localhost:5000/hello/proba?kwota=" .
$_POST['kwota'] . "&procent=" . $_POST['procent'] . "&lata=" .
$_POST['lata']));
                  $file=stristr($file, "<p>");
                  $file=stristr($file, ">");
                  $file=stristr($file, "<p>");
                  $i=strpos($file, "<p>", 5);
                  echo "Suma zwrocona przez Prologa wynosi: ".substr($file,
3, $i-1);

            }
      }

?>
```

Kod po stronie Prologa pozostaje taki sam jak w przykładzie z punktu 3.5

6. Bibliografia

Prezentacja ta w głównej mierze bazuje na opracowaniu dostępnym po adresem http://www.swi-prolog.org/packages/http.htm.

Więcej o protokole http można dowiedzieć się pod następującymi adresami:
http://www.w3.org/Protocols/
http://www.jmarshall.com/easy/http/