CGI

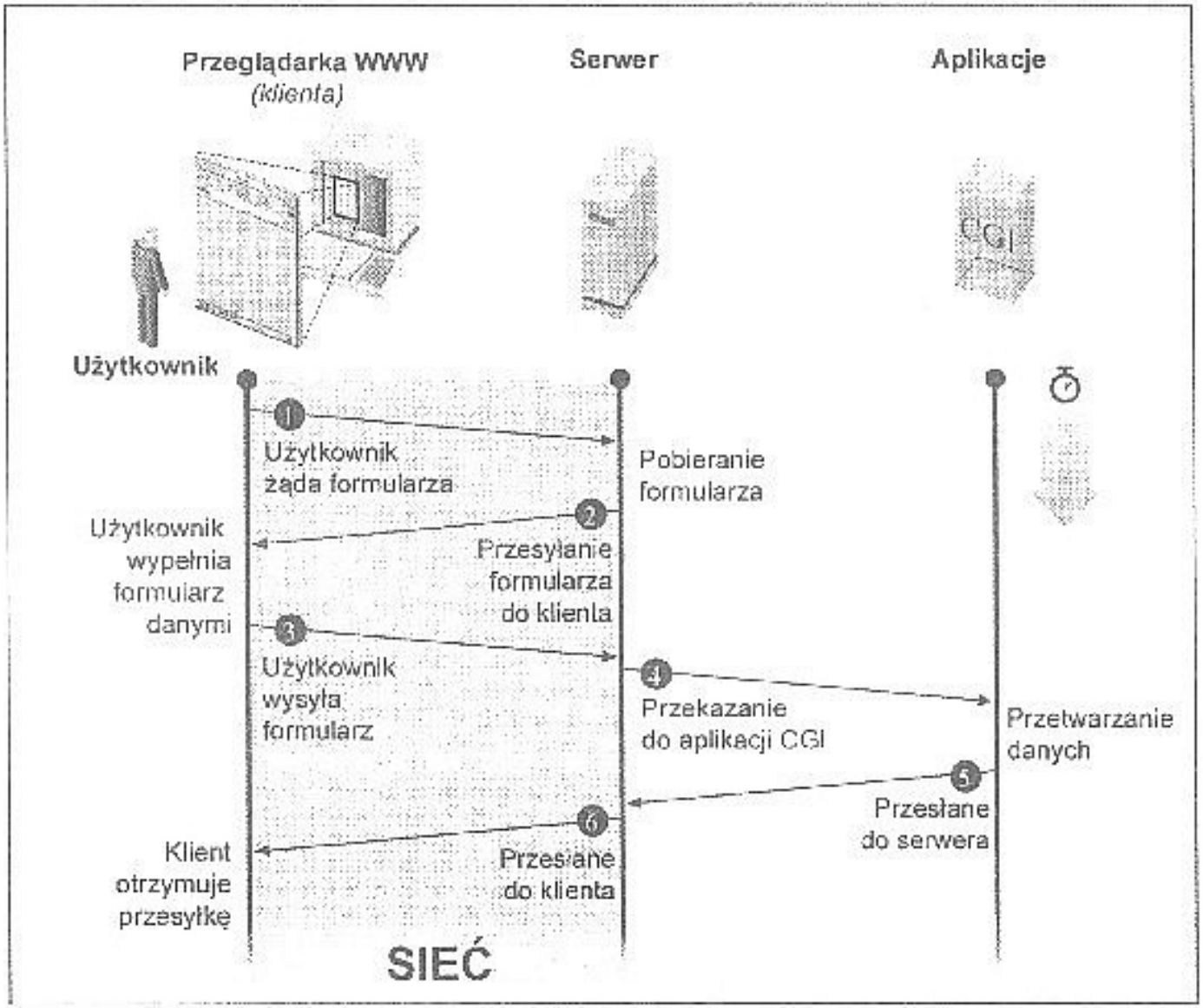
CGI (Common Gateway Interface) - znormalizowany interfejs, umożliwiający komunikację pomiędzy oprogramowaniem serwera WWW a innymi programami znajdującymi się na serwerze.

Umożliwia dynamiczne (na żądanie klienta) generowanie dokumentów HTML uzupełniając je np. treścią pobieraną z bazy danych. Skrypt CGI tworzy stronę HTML w locie (z ang. on-the-fly).

Jedna z najstarszych metod i najbardziej stabilna (od 1995 roku).

Obecna wersja: 1.1

Schemat działania



Zastosowanie

- Dynamiczne generowanie dokumentów przed wysłaniem ich do przeglądarki (np. z aktualną datą)
- Tworzenie dokumentów w oparciu o dane znajdujące się w bazie i formatowanie rekordów w celu wyświetlenie ich zawartości na stronie
- Pobieranie i formatowanie danych będących wynikiem działania innego oprogramowania (np. dane pobierane z urządzenia pomiarowego mogą być na bieżąco wysyłane do przeglądarki)
- Generowanie i przetwarzanie ankiet i kwestionariuszy
- Tworzenie dynamicznych ilustracji - takich jak wykresy czy schematy

Cechy

- Stabilny (bez zmian od 1995 roku), dostępny za darmo standard.
- Implementacja CGI nie jest zależna od konkretnej platformy sprzętowej/systemowej, zależy natomiast od konkretnego programu serwera WWW. Większość popularnych serwerów ma zaimplementowany mechanizm CGI, włączając w to serwery działające na systemach Unix (Apache, Sun Java System Web Server, Microsoft IIS).
- Programy CGI można pisać praktycznie w dowolnym języku programowania. Często wykorzystuje się Perla, PHP, C, C++, Visual Basic i AppleScript.

  Język powinien umożliwiać:
  * czytać ze standardowego wejścia (STDIN);
  * pisać na standardowe wyjście (STDOUT);
  * obsługiwać zmienne systemowe (ENVARS);
  * czytać parametry wywołania programu.

Dla wielu z tych języków stworzono biblioteki wspomagające obsługę CGI.

Wady

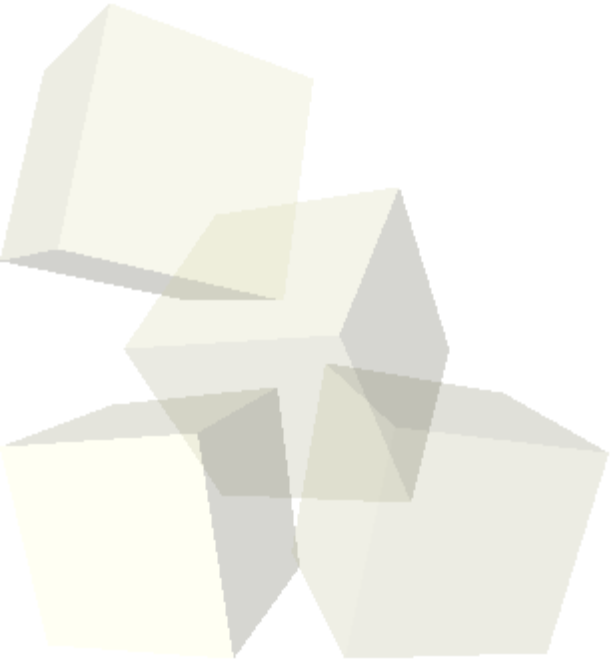
- Przygotowywanie dostosowanych do indywidualnych potrzeb skryptów CGI wymaga zazwyczaj sporych umiejętności programowania.
- Absolutnie niezbędna jest również możliwośc sprawowania przez autora stron kontroli nad serwerem, w tym nad wszelkimi programami współpracującymi ze stronami, np. systemami baz danych
- Obsługa CGI wiąże się zazwyczaj z tworzeniem nowego procesu na każde żądanie. Powoduje to duże obciążenie serwera, zwłaszcza dla języków interpretowanych. Powstały rozwiązania przyśpieszające typu FastCGI lub automatyczne tworzenie tymczasowych wersji kompilowanych.
- Skrypty CGI obciążają serwer, podczas gdy komputery klientów nie są wykorzystywane

Bezpeiczeństwo a CGI

- Aby umieścić skrypt na serwerze Apache należy posiadać uprawnienia root'a (domyślna konfiguracja)
- Skrypty mogą umożliwiać dostęp do zasobów, do których użytkownik nie ma (np. użytkownik ***user*** ma inne prawa niż skrypt uruchamiany na przykład z prawami użytkownika ***apache***)
- Błędnie napisane skrypty mogą być skompromitowane i mogą stanowić poważną lukę w zabezpieczeniach systemu
- Zwykle administratorzy nie pozwalają użytkownikom dodawanie własnych skryptów, oferując w zamian dostęp do umieszczonych przez siebie skryptów

Przekazywanie parametrów

Metody: POST oraz GET



GET

Informacje są przekazywane poprzez dołączanie ich do ciągu tekstowego, stanowiącego adres URL.

- Dane użytkownika występują w adresie po znaku zapytania i są zorganizowane w pary nazwa-wartość, połączone znakiem równości.
- Poszczególne pary oddzielone są symbolem &.
- W większości przypadków stosowane nazwy odpowiadają nazwom zmiennych, stosowanym w programach obsługi formularzy.
- Implemenatacja jest bardzo prosta i sprowadza się do dodania odpowiednich łańcuchów znakowych do adresu URL.
- Metoda z założenia pozbawiona jakichkolwiek zabezpieczeń, użytkownik może bez problemu zmienić nazwy i wartości parametrów
- Ilość danych przekazywanych metodą GET jest ograniczona maksymalną długością odnośnika do skryptu (zwykle przeglądarki akceptują max. 1024 znaki)
- Dane są przekazywane w zmiennej **QUERY_STRING**

```
http://jakas.strona.pl/skrypt.cgi?name=zosia&nazwisko=kowalska&wiek=23
```

POST

- Umożliwia przekazywanie dużej liczby zmiennych przy zachowaniu podstawowych zasad poufności. Nazwy zmiennych i ich wartości nie są widoczne, ponieważ są wysyłane jako część nagłówka pliku.
- Minusem tej metody jest to, że strony z wysłanymi danymi metodą POST nie można np. dodać do Ulubionych.
- Dane są przekazywane na standardowe wejście

Żądanie HTTP – POST oraz GET

```
POST /cgi/skrypt_cgi HTTP/1.1
Host: AdresSerweraHTTP
Content-Length: 47
Content-Type:  application/x-www-form-
  urlencoded

naz_imie=Imie+Nazwisko&email=aaa%40bbb.com
```

```
GET /cgi/skrypt_cgi?
  naz_imie=Imie+Nazwisko&email=aaa
  %40bbb.com HTTP/1.1
host: AdresSerweraHTTP
```

Zmienne środowiskowe

CONTENT_LENGTH - Liczba bajtów danych wysłanych z przeglądarki do aplikacji CGI w metodzie POST (dane te są dostępne na standardowym wejściu)

CONTENT_TYPE - Typ danych wysłanych z przeglądarki do aplikacji CGI w metodzie POST

QUERY_STRING - Dane wysłane do aplikacji CGI w metodzie GET

HTTP_ACCEPT - Typy danych MIME, jakie może przesyłać serwer WWW do/z aplikacji CGI

HTTP_USER_AGENT - Nazwa przeglądarki w formacie: aplikacja/wesja_biblioteki/wersja

GATEWAY_INTERFACE - Wersja CGI zainstalowana na serwerze

SCRIPT_NAME - Nazwa programu CGI, który jest uruchomiony

REMOTE_ADDR - Adres IP (liczby) maszyny, na której użytkownik ogląda strony WWW

REMOTE_HOST - Adres (tekstowy) maszyny, na której użytkownik ogląda strony WWW

REMOTE_USER - Nazwa użytkownika w systemie, który ogląda strony WWW

REQUEST_METHOD - Metoda przesyłania danych przez protokół HTTP: GET lub POST

SERVER_NAME - Adres (tekstowy) maszyny, na której jest uruchomiony program CGI

SERVER_SOFTWARE - Nazwa i wersja oprogramowania serwera WWW, na którym jest uruchomiony program CGI

SERVER_PROTOCOL - Nazwa i wersja protokołu do przesyłania danych

SERVER_PORT - Numer portu, z którego korzysta serwer WWW

Zmienne środowiskowe - przykład

```
CONTENT_TYPE = (null)
CONTENT_LENGTH = (null)
QUERY_STRING = imie=Jan&nazwisko=Kowalski&wiek=35
HTTP_ACCEPT = text/html, image/jpeg, image/png, text/*, image/*,
*/*
HTTP_USER_AGENT = Mozilla/5.0 (compatible; Konqueror/3.5; Linux)
                              KHTML/3.5.8 (like Gecko)
GATEWAY_INTERFACE = CGI/1.1
SCRIPT_NAME = /cgi-bin/environment
PATH_INFO = (null)
PATH_TRANSLATED = (null)
REMOTE_ADDR = ::1
REMOTE_HOST = (null)
REMOTE_USER = (null)
REQUEST_METHOD = GET
SERVER_NAME = localhost
SERVER_SOFTWARE = Apache/2.2.6 (Fedora)
SERVER_PROTOCOL = HTTP/1.1
SERVER_PORT = 80
CONTENT_TYP = (null)
CONTENT_TYP = (null)
```

Apache i skrypty CGI

- Konfiguracja

W pliku /etc/httpd/conf/httpd.conf

```
ScriptAlias /cgi-bin/
  "/var/www/cgi-bin/"
<Directory "/var/www/cgi-bin">
    AllowOverride None
    Options None
    Order allow,deny
    Allow from all
</Directory>
```

- Skrypty CGI należy umieszczać w “/var/www/cgi-bin”

Apache + Perl

- W pliku /etc/httpd/conf.d/perl.conf odkomentować

```
Alias /perl /var/www/perl
<Directory /var/www/perl>
    SetHandler perl-script
    PerlResponseHandler
 ModPerl::Registry
    Perl Options +ParseHeaders
    Options +ExecCGI
</Directory>
```

- Utworzyć katalog /var/www/perl
- Zrestartować serwer Apache

Przykłady CGI

Każdy program (bez względu na użyty język programowania) generujący kod postaci

```
Content-type: text/html

<html>
    <head><title>Przyklad skryptu</title></head>
     <body>To jest przyklad skryptu CGI</body>
</html>
```

i umieszczony na serwerze jako skrypt CGI, wygeneruje odpowiednią stronę.

Przykład skryptu CGI w C

```
#include<stdio.h>
#include<stdlib.h>

int main()
{
    printf("%s%c%c\n",
          "ContentType:text/html;charset=iso-8859-1",13,10);
    printf("<HTML>");
    printf("<TITLE>Zmienne środowiskowe</TITLE>");
    printf("<BODY><h1>Hello, jestem skryptem
                CGI</h1></BODY>");
    printf("QUERY_STRING = %s <BR>",getenv("QUERY_STRING") );
    printf("<HTML>");

    return 0;
}
```

Przykład skryptu CGI w Perlu

```
#!/usr/bin/perl
print "Content-type: text/html\n\n";
print "<html>\n<head><title>Skrypt CGI</title></head>\n";
print "<body>\n<h1>Jestem skryptem CGI w Perlu</h1>\n";
print "CONTENT_TYPE = $ENV{'CONTENT_TYPE'}<BR>";
print "CONTENT_LENGTH = $ENV{'CONTENT_LENGTH'}<BR>";
print "QUERY_STRING = $ENV{'QUERY_STRING'}<BR>";
print "</body>\n</html>\n";
```

Przykład skryptu CGI w Unix/shell

```
#!/bin/sh
    echo Content-type: text/html
    echo
    echo "<html><head><title>Zmienne Srodowiskowe</title>"
    echo "</head><body>"
    echo "<h1>Zmienne Srodowiskowe</h1>"
    echo "CONTENT_TYPE = $CONTENT_TYPE <BR>";
    echo "CONTENT_LENGTH = $CONTENT_LENGTH <BR>";
    echo "QUERY_STRING = $QUERY_STRING <BR>";
    echo "</body></html>"
```

Zegar w CGI

http://www.cs.put.poznan.pl/jkobusinski/cgi.html

```
#!/bin/bash

echo "Content-Type: text/html"
echo "Refresh: 1"
echo
echo "<html><body>"
date
echo "</body></html>"
```

Licznik w CGI

http://www.cs.put.poznan.pl/jkobusinski/cgi.html

```
#!/bin/bash

echo "Content-type: text/html"
echo
touch count.txt
VAL=cat count.txt
if [ -z $VAL ]
then
  VAL=1
fi
echo "<html><body>Do tej pory jest to $VAL wizyta <br>"
VAL=$((VAL+1))

grep $REMOTE_ADDR adresy.txt > /dev/null

if [ ! $? -eq 0 ]
 then
  echo $REMOTE_ADDR >> adresy.txt
  echo $VAL > count.txt
fi

echo "<a href='adresy.txt'>Plik z adresami osób
  odwiedzających</a><br><hr>"
cat adresy.txt
echo "</body></html>"
```

Parametry GET

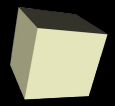
http://www.cs.put.poznan.pl/jkobusinski/cgi.html

```
#!/bin/bash

echo "Content-Type: text/html"
echo

echo "QUERY_STRING = [$QUERY_STRING]"
echo

#Zakończ jeśli nie podano żadnych parametrów
[ -z $QUERY_STRING ] && exit

QUERY=$QUERY_STRING"&"
while true
do
#Ze zmiennej QUERY usuwany jest z końca najdłuższy (%%)
#pasujący do wzorca "&*" ciąg znaków
#np. QUERY="A=1&B=2&C=3" -> ARG="A=1"
  ARG=${QUERY%%&*}

#Zakończ pętlę jeśli nic nie zostało
  [ "$ARG" = "" ] && break;

#Ze zmiennej ARG usuwany jest z początku najkrótszy (#)
#pasujący do wzorca "*=" ciąg znaków
#np. ARG="A=1" -> ARG_VALUE="1"
  ARG_VALUE=${ARG#*=}

#Ze zmiennej ARG usuwany jest z końca najkrótszy (%)
#pasujący do wzorca "*=" ciąg znaków
#np. ARG="A=1" -> ARG_NAME="A"
  ARG_NAME=${ARG%=*}

  echo -e " arg:$ARG_NAME \t value:$ARG_VALUE\n<br/>"
  QUERY=${QUERY#*&}
done
```

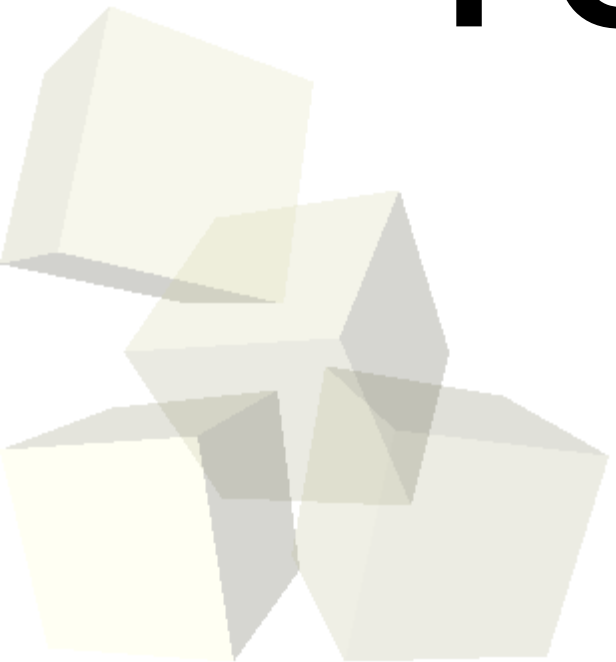
Parametry POST

http://www.cs.put.poznan.pl/jkobusinski/cgi.html

```
#!/bin/bash

echo "Content-Type: text/html"
echo

read QUERY_STRING
echo "QUERY_STRING = [$QUERY_STRING]"
echo

[ -z $QUERY_STRING ] && exit

QUERY=$QUERY_STRING"&"
while true
do
  ARG=${QUERY%%&*}
  [ "$ARG" = "" ] && break;

  ARG_VALUE=${ARG#*=}
  ARG_NAME=${ARG%=*}
  echo -e " arg:$ARG_NAME \t value:$ARG_VALUE\n
  <br/>"

  QUERY=${QUERY#*&}
done
```

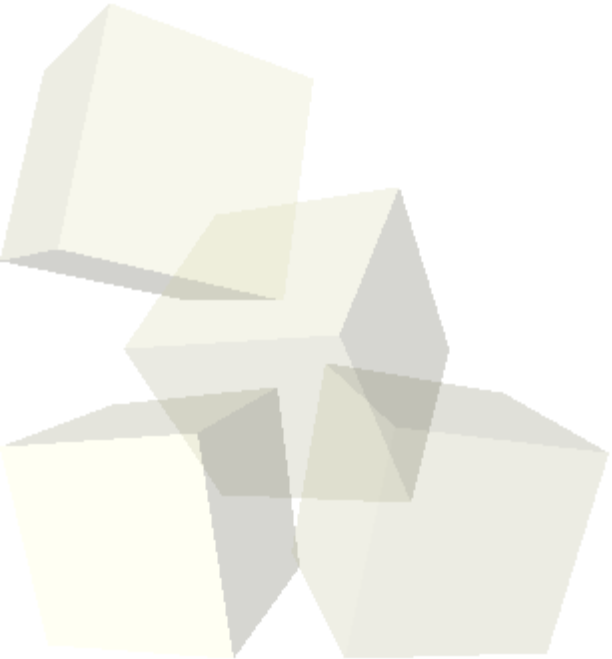


FORMULARZE



Formularze

- Służą użytkownikom do wprowadzania danych
- Przykłady:
  - Ankiety
  - Wysyłanie maili
  - Ogólnie - aplikacje wymagające przesłania danych do serwera WWW



Formularze

```
<FORM ACTION="url_do_CGI/jakas_akcja" METHOD="POST/GET">
   tu wstawiamy elementy formularza
</FORM>
```

ACTION – określa URL skryptu CGI odbierającego żądanie HTTP
METHOD – określa metodę żądania HTTP (GET lub POST). Wartość domyślna – GET.
ENCTYPE – określa sposób kodowania danych dołączonych do formularza. Atrybut ten ma znaczenie tylko dla metody POST (żądania GET nie mają części zasadniczej). Standardowo i domyślnie application/x-www-form-urlencoded. Jedynie do wysyłania plików używany jest multipart/form-data.

UWAGA:
- W dokumencie może występować wiele formularzy.
- Formularzy ***nie wolno*** zagnieżdżać.

```
<FORM ACTION="/cgi-bin//hello_cgi" METHOD=”POST">

</FORM>
```

Znaczniki formularzy

- **Pola tekstowe:**

```
<INPUT TYPE="text/password/hidden" NAME="nazwa"
  VALUE="wartosc" SIZE="rozmiar" MAXLENGTH=”max_rozmiar”>
```

TYPE – określa typ pola tekstowego;
text – zwykłe pole tekstowe;
password – pole do wpisywania haseł;
hidden – pole ukryte (niewidoczne dla użytkownika w oknie przeglądarki, ale widoczne w źródle strony); używane do przekazywania parametrów pomiędzy formularzami; możliwe tylko parametry NAME i VALUE
VALUE – tekst wyswietlany w polu tekstowym (domyślnie pusty łańcuch)
SIZE – rozmiar pola (domyślna wartość 20)
MAXLENGTH – Maksymalna liczba znaków pola (domyślnie bez ograniczeń)
READONLY=”readonly” - Pole tylko do odczytu
DISABLE=”disable” - Pole zablokowane – tylko do odczytu, dane nie są przesyłane

Przykład

```
<html>
  <head>
    <title>Formularz</title>
    <meta http-equiv="Content-type" content="text/html;
         charset=iso-8859-2">
  </head>
<body>
  Formularz do wprowadzania danych <br>
  <form>
    <input type="text" value="Formularz" readonly="readonly">
    </input> - Pole tekstowe tylko do odczytu<br>
    Login:
    <input type="text">
    </input> - Zykłe pole tekstowe <br>
    Hasło:
    <input type="password">
    </input> - Pole do wpisywania haseł <br>
    <input type="hidden">
    </input> - Pole tekstowe ukryte <br>
    Opis: <br>
    <input type="text" size="40">
    </input> - Zykłe pole tekstowe <br>
  </form>
</body>
</html>
```

Wynik

Formularz do wprowadzania danych

Formularz - Pole tekstowe tylko do odczytu

Login: - Zykłe pole tekstowe

Hasło: - Pole do wpisywania haseł

- Pole tekstowe ukryte

Opis:

- Zykłe pole tekstowe

Pola wyboru

- **Pola wyboru:**

```
<INPUT TYPE="checkbox" NAME="nazwa" VALUE="wartosc"
CHECKED=”checked” DISABLED=”disabled”>
```

Wartość “Name” powinna być taka sama, dla wielu pól wyboru odnoszących się do tego samego pytania, np. składniki pizzy.

Otoczenie znacznikami <label> spowoduje możliwośc zaznaczania/odznaczania nawet po kliknięciu na etykietę.

```
<LABEL> <INPUT TYPE="checkbox" NAME="nazwa" VALUE="wartosc"
CHECKED=”checked” DISABLED=”disabled”> </LABEL>
```

Pola wyboru

```
<form>
Składniki pizzy: <br>

<input TYPE="checkbox" NAME="pizza" VALUE="piecz" checked="checked"
disabled="disabled"> Pieczarki</input> <br>
<input TYPE="checkbox" NAME="pizza" VALUE="ser2" checked="checked"> Dodatkowy
ser</input> <br>
<input TYPE="checkbox" NAME="pizza" VALUE="sal"> Salami</input> <br>
<input TYPE="checkbox" NAME="pizza" VALUE="oli"> Oliwki</input> <br>
</form>
```

Składniki pizzy:
- [x] Pieczarki
- [x] Dodatkowy ser
- [ ] Salami
- [ ] Oliwki

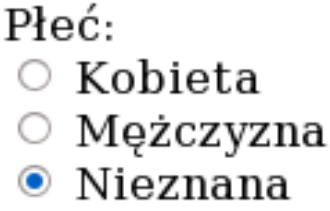
Pola opcji

- **Pola opcji – wybór jednego z wielu**

```
<INPUT TYPE="radio" NAME="nazwa" VALUE="wartosc"
CHECKED=”checked” DISABLED=”disabled”>
```

Pola dotyczące jednej opcji powinny mieć taką samą wartość name

```
<form>
Płeć: <br>
<input TYPE="radio" NAME="plec" VALUE="woman">
Kobieta</input> <br>
<input TYPE="radio" NAME="plec" VALUE="man">
Mężczyzna</input> <br>
<input TYPE="radio" NAME="plec" VALUE="unknown"
checked="checked"> Nieznana</input> <br>
</form>
```



Znaczniki formularzy - listy

- **Lista rozwijalna:**

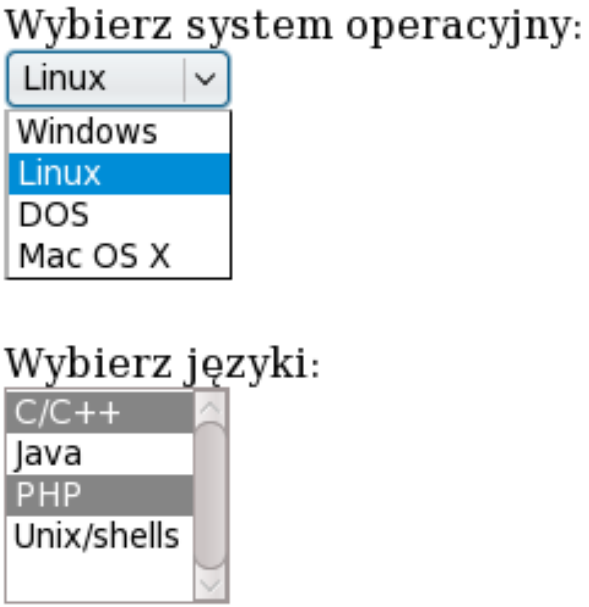
```
<SELECT NAME="nazwa" SIZE=1>
<OPTION SELECTED>Pierwszy</OPTION>
<OPTION>Drugi</OPTION>
</SELECT>
```

- **Lista zwykła (pole listy):**

```
<SELECT NAME="nazwa" SIZE=n MULTIPLE>
<OPTION SELECTED>Pierwszy</OPTION>
<OPTION>Drugi</OPTION>
</SELECT>
```

Listy

```
<form>
Wybierz system operacyjny: <br>
<select name="so" SIZE=1>
<option>Windows</option>
<option selected>Linux</option>
<option>DOS</option>
<option>Mac OS X</option>
</select>
<br><br><br><br><br><br>

Wybierz języki:<br>

<select name="nazwa" size=5 multiple>
<option selected>C/C++</option>
<option>Java</option>
<option selected>PHP</option>
<option>Unix/shells</option>
</select>
</form>
```



Znaczniki formularzy

- **Wielowierszowe pole tekstowe:**

```
<TEXTAREA ROWS=yy COLS=xx NAME="nazwa">
  a tu tekst
</TEXTAREA>
```

```
<form>
Opis: <br>
<textarea rows="10" cols="20"
name="desc">
Wpisz tekst
</textarea>
</form>
```

Opis:

Wpisz tekst

Znaczniki formularzy

- **Selektor plików:**

```
<INPUT TYPE="file" NAME="nazwa"> </input>
```

```
<form>
<input type="file" name="plik_zrodlowy">
</input>
</form>
```

Znaczniki formularzy - przyciski

Przycisk graficzny:

```
<INPUT TYPE="image" SRC="url_obrazka" NAME="nazwa"
  VALUE="wartosc">
```

- **Przycisk zwykły:**

```
<INPUT TYPE="button" NAME="nazwa" VALUE="wartosc">
```

Przycisk wysułający dane:

```
<INPUT TYPE="submit" NAME="nazwa" VALUE="wartosc">
```

Przycisk czyszczący dane:

```
<INPUT TYPE="reset" NAME="nazwa" VALUE="wartosc">
```

Grupowanie pól

- Znacznik ***fieldset*** z opcionalnym znacznikiem tytułu ***legend***.

```
<fieldset>
   Zgrupowane pola formularza
</fieldset>

<fieldset>
<legend align="rodzaj">Tytuł grupy</legend>
   Zgrupowane pola formularza
</fieldset>
```

Login:
Hasło:

Składniki pizzy

- [x] Pieczarki
- [x] Dodatkowy ser
- [ ] Salami
- [ ] Oliwki

Tabulatory

tabindex=”n” - gdzie n dowolna liczba

Służą do określania kolejności przechodzenia pomiędzy polami formularza.

Kolejność od najmnijeszej do największej. Gdy klika pól ma taką samą wartość, aktywacja przebiega według kolejności ich występowania w dokumencie.

Numery nie muszą być kolejne.

Elementy “disabled” są pomijane.

KONIEC