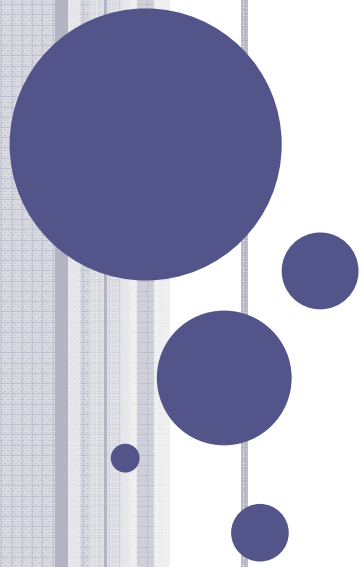
Esther González Rodríguez

28 de mayo de 2008



Perl y Ajax

CGI::Ajax

ÍNDICE

- Webs estáticas: HTML
- Webs dinámicas: DHTML
- CGI's
- AJAX
- CGI::Ajax
  - Métodos
- Ejemplos
- Conclusiones
- Bibliografía

WEBS ESTÁTICAS: HTML

- Realizadas en XHTML o HTML.
- Para cambiar los contenidos hay que acceder al servidor.
- El proceso de actualización es lento.
- No se pueden usar bases de datos, foros, etc.

WEBS DINÁMICAS: DYNAMICHTML

- Técnicas que permiten crear sitios web interactivos usando HTML combinado con otros lenguajes. Así, es posible crear páginas cuyo contenido sea diferente según el usuario que las invoque.

- En el cliente: JavaScript, VBScript, Flash, CSS, DOM.EG1

- En el servidor: ASP, PHP, JSP, CGI.

- Usos (sin compatibilidad entre navegadores):
  - Acceso a bases de datos.
  - Tiendas virtuales.
  - Webs con distintos aspectos según los roles…

Diapositiva 4

EG1 Modelo a través de la cual los programas y scripts pueden acceder y modificar dinámicamente el contenido, estructura y estilo de los documentos HTML y XML.

Su objetivo es facilitar contenido dinámico en las páginas web.

Esther González; 18/05/2008

CGI's (Common Gateway Interface)

- Interfaz entre los servidores estándar HTTP y los "programas dedicados" que se encargan de resolver algún servicio que le delega el servidor.

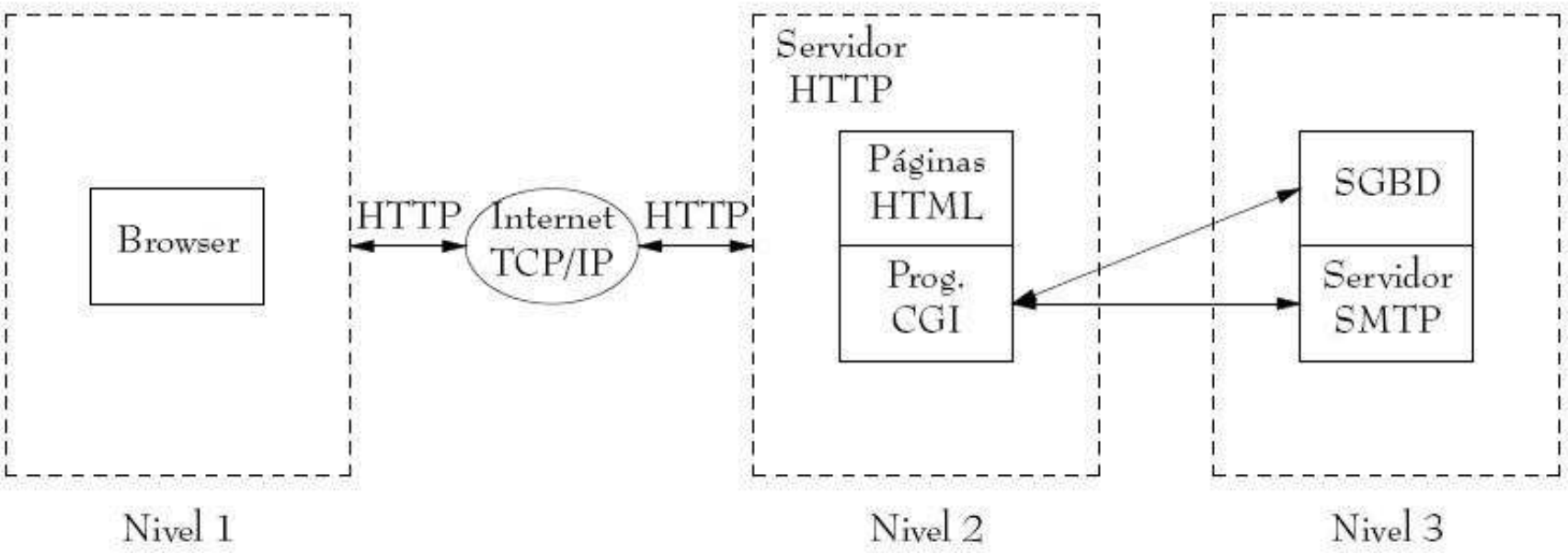
- Es una de las primeras formas que surgió de crear contenido dinámico para las páginas web.

CGI's (2)

- Es independiente del lenguaje de programación del servidor.
- La salida del programa, objetos MIME, es enviada al cliente (en vez del archivo estático).
- Usos habituales: contador, buscador, correo, estadísticas, administración remota del servidor.

CGI's (3)

- Cliente/servidor utilizando CGI:



AJAX (ASYNCHRONOUS JAVASCRIPT AND XML)

- Técnica para crear páginas web más interactivas.
- Se ejecuta en el cliente (navegador) y mantiene comunicación asíncrona con el servidor en segundo plano.
- Permite realizar cambios en partes de una web sin necesidad de recargar la página entera.
- Ventajas → aumenta la interactividad, la velocidad y la usabilidad de las webs.

AJAX (2)

- Es una combinación de 4 tecnologías existentes previamente:
  - **HTML + CSS**: para el diseño.
  - **DOM**: accedido en el usuario por el lenguaje de script para mostrar e interactuar con la información.
  - **XMLHttpRequest**: objeto para intercambiar datos asíncronamente con el servidor.
  - **XML**: formato para la transferencia de vuelta al servidor.

- Ejemplos de uso: Gmail, Google Maps, Flickr.

CGI::Ajax

- Autores:
  - Brian C. Thomas - bct.x42@gmail.com
  - Brent Pedersen - bpederse@gmail.com
- Homepage:
  - http://www.perljax.us/
- Descarga:
  - http://search.cpan.org/~bct/CGI-Ajax-0.701/lib/CGI/Ajax.pm
- Requisitos:
  - Servidor Apache
  - Módulo CGI
  - Módulo Class::Accessor
  - Módulo CGI::Ajax

CGI::AJAX (2)

- Módulo orientado a objetos.
- Mecanismo único para utilizar código Perl en páginas web.
- Permite que una subrutina Perl se invoque de forma asíncrona cuando ocurre:
  - un **evento JavaScript**.
  - una **petición HTTP** (httprequest).
- Generalmente, el usuario no tiene que escribir código JavaScript.
- Excepciones → código asociado a eventos: onClick, onKeyUp, onMouseOver…

MÉTODO: BUILD_HTML()

- **Propósito**:
  - Asocia el objeto CGI ($cgi) con el objeto CGI::Ajax ($pjx).
  - Inserta código JavaScript entre <HEAD>…</HEAD>.
  - Construye la página o la parte a recargar de ella.

- **Argumentos de entrada**:
  - El objeto CGI.
  - Un *coderef* o un *string* que contenga el HTML.

- **Argumentos de salida**:
  - El código HTML.

- **Invocación**:
  - La realiza el propio script.

MÉTODO: SHOW_JAVASCRIPT()

- **Propósito**:
  - Construir el código JavaScript que se incrustará en la página.

- **Argumentos de entrada**:
  - -

- **Argumentos de salida**:
  - Código JavaScript.

- **Invocación**:
  - La realiza el propio script.

MÉTODO: REGISTER()

- **Propósito**:
  - Añade un nombre de función y una referencia de código al hash de referencias global, después que es creado el objeto.

- **Argumentos de entrada**:
  - Nombre de función.
  - Referencia de código.

- **Argumentos de salida**:
  - -

- **Invocación**:
  - La realiza el propio script.

OTROS MÉTODOS

- JSDEBUG()
  - Funciones de debug para la generación del código Ajax.

- DEBUG()
  - Muestra información de debug en el log del servidor.

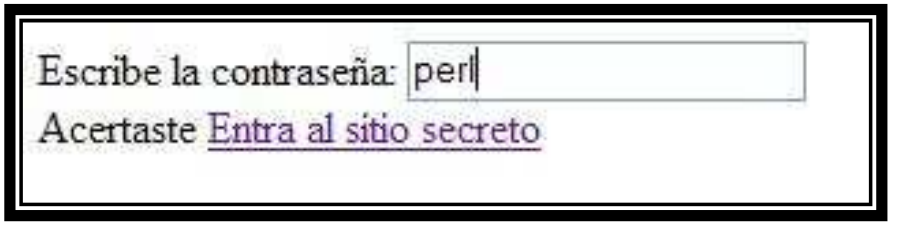
EJEMPLO 1

```
#!c:/Perl/bin/perl.exe
use strict;
use CGI;
use CGI::Ajax;

my $cgi = new CGI;

my $PWD_CORRECTO = "perl";

#Creamos un nuevo objeto AJAX y le asociamos una función
my $pjx = new CGI::Ajax( 'checar_pwd' => \&checar_pwd );

# Desplegamos el HTML
print $pjx->build_html( $cgi, \&mostrar_formulario);
```

EJEMPLO 1 (2)

```
sub checar_pwd{
    #Recibimos los datos envíados
    my $entrada = shift;

    if($entrada eq "") {
            return("Introduce la contraseña secreta");
    } if($entrada eq $PWD_CORRECTO) {
            return("Acertaste <a
            href=\"http://google.com\">Entra al sitio secreto</a>");
    } else{
            return("$entrada es incorrecto");
    }
}
```

EJEMPLO 1 (3)




```
sub mostrar_formulario {
    my $html = <<EOHTML;
            <HTML>
            <BODY>

            Escribe la contraseña:
            <input type="text" name="pwd" id="pwd"
            onKeyUp="checar_pwd( ['pwd'], ['resultado'] );">
            <br>
            <div id="resultado"></div>
            <p>

            </BODY>
            </HTML>
    EOHTML
    return $html;
}
```

EJEMPLO 2

```
#!c:/Perl/bin/perl.exe
use strict;
use warnings;
use CGI;
use CGI::Ajax;

my $cgi  = CGI->new();
my $ajax = CGI::Ajax->new(check_username=>\&check_username);

$ajax->JSDEBUG(1);

print $ajax->build_html( $cgi, \&main )
```

EJEMPLO 2

```
sub check_username {
    my ( $user ) = @_;
    return unless -f '/tmp/users.txt';
    open my $fh, '<', '/tmp/users.txt' or return "open(/tmp/users.txt): $!";

    while (<$fh>) {
            chomp;
            if (lc $_ eq lc $user) {
            return "Username '$user' taken! <img src=\"../Flores.jpg\" width=210
    height=140 border=0 alt=\"Unas flores\">" if($user=~/alu.*/);
            return "Username '$user' taken! <img src=\"../Delfines.jpg\"
    width=210 height=140 border=0 alt=\"Unos delfines\">"
    if($user=~/profe.*/);
            return "Username '$user' taken! <img src=\"../Prohibido.jpg\"
    width=210 height=140 border=0 alt=\"Prohibido\">" if($user=~/.*/);
            }
    } return '';
}
```

EJEMPLO 2 (2)

```
sub save_username {
    my ( $user ) = @_;
    open my $fh, '>>', /tmp/users.txt‘ or die
    "open(>>/tmp/users.txt): $!";
    print $fh "$user\n";
    close $fh;
    return;
}
```

EJEMPLO 2 (3)

```
sub main {
    my $html = <<HTML;
            <html><head>
            <title>Signup!</title>
            <script type="text/javascript" src="binding.js"></script>
            </head><body>
            <h1>Signup!</h1>
            HTML
    if ( my $user = $cgi->param('user') ) {
            my $err = check_username( $user );
            if ( $err ) {
                    $html .= "<p class='problem'>$err</p>";
            } else {
                    save_username( $user );
                    $html .= "<p>Account <em>$user</em> created!</p>\n";
            }
    }
```

EJEMPLO 2 (4)

```
my $url = $cgi->url(-relative => 1);
$html .= <<HTML;
        <form action="$url" method="post">
        <p>Please fill in the details to create a new Account.</p>
        <p>Username: <input type="text" name="user" id="user"/>
        <em id="baduser"></em></p>
        <p>Password: <input type="password" name="pass"
id="pass"/></p>
        <p><input type="submit" name="submit"
value="SIGNUP"/></p>
        </form></body></html>
        HTML
return $html;
} # end main
```

EJEMPLO 2 (5)

Signup!

Please fill in the details to create a new Account.

Username: alu3228

Password: *******

SIGNUP

Signup!

Account *alu3228* created!

Signup!

Username 'alu3228' taken!

Signup!

Username 'profe1234' taken!

Signup!

Username 'maria' taken!

CONCLUSIONES

- Único mecanismo para utilizar Perl y Ajax.
- Fácil de usar, muy orientado al usuario final.
- Aísla al usuario del código JavaScript.
- También es posible escribir código para manejar los eventos JavaScript.
- Es un método rápido de obtener páginas web que ofrecen al usuario una mayor interactividad.

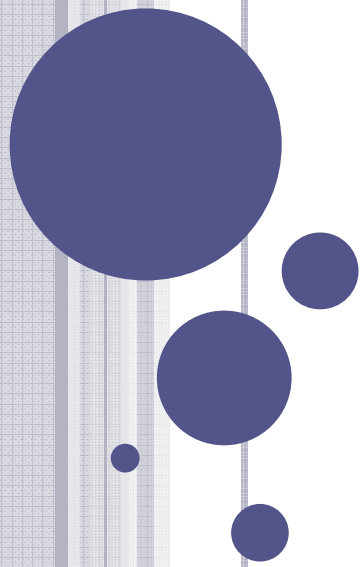
BIBLIOGRAFÍA

- http://www.desarrolloweb.com/articulos/391.php
- http://es.wikipedia.org/wiki/Common_Gateway_Interface
- http://search.cpan.org/
- http://www.perljax.us/
- http://perlenespanol.baboonsoftware.com/tutoriales/cgi/perl_y_ajax.html
- http://www.perl.com/pub/a/2006/03/02/ajax_and_perl.html

Esther González Rodríguez

28 de mayo de 2008

¡Gracias por su atención!



Perl y Ajax

CGI::Ajax