VOLTAR

Tecnologias Client-Server

Entender o que como as diferenças das aplicações client-side e server-side e como funcionam as aplicações server-side.

NESTE TÓPICO

Marcar tópico

O que é linguagem *client-side* e linguagem *server-side*?

Quando estamos programando para *internet*, existem dois tipos de linguagens: as linguagens *client-side* e as linguagens *server-side*.

Mas quais são as diferenças entre elas?

Linguagens *client-side* são scripts processados no próprio cliente, ou seja, no próprio navegador (SANTOS, 2015). Normalmente estes *scripts* tem como objetivo executar validações em controles em formulários, consistências em telas, dentre outras aplicações. O maior problema de se utilizar este tipo de solução é a incompatibilidade de interpretação da linguagem entre os *browsers*, neste caso a recomendação é que a codificação proposta utilize um script *cross-browser*, ou seja, que seja suportada por vários navegadores. Esta aplicação deve ser construída utilizando códigos compatíveis com qualquer navegador e principalmente que atenda as especificações do W3C (W3C, 2015). São páginas estáticas escritas utilizando a linguagem HTML, não possuem nenhuma interação com o usuário, devido as próprias restrições do HTML. Linguagens *client-side* também podem ser chamadas de páginas estáticas. Na figura 1 observa-se um processo *client-side* sendo executado sem qualquer conexão com sistemas computacionais externos.

Figura 1: Exemplo de aplicação client-side. Fonte: Autor.

Figura 1: Exemplo de aplicação client-side. Fonte: Autor.

Fonte: Autor

São exemplos de linguagens *client-side*:

- HTML (xHTML, HTML 4.0; HTML 4.01; HTML 5), (MOZILLA DEVELOPER NETWORK, 2015a)
- CSS (CSS2; CSS3), (MOZILLA DEVELOPER NETWORK, 2015b)
- Javascript (JS 1.0; JS 1.1; ... e JS 1.8), (MOZILLA DEVELOPER NETWORK, 2015c)

Linguagens *server-side* são *scripts* que serão processados no servidor. Toda vez que o usuário solicitar uma página que contenha um código PHP, por exemplo: para acessar uma informação num banco de dados, o servidor interpretará este código e retornará ao cliente somente o HTML, que por sua vez, será interpretado pelo navegador (SANTOS, 2015). Linguagens *server-side* também podem ser chamadas de páginas dinâmicas.

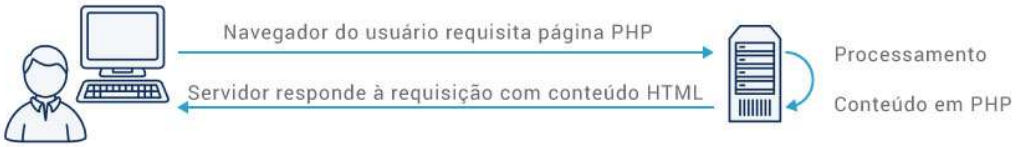

Figura 2: Exemplo de aplicação server-side. Fonte: Autor.

Fonte: Autor

São exemplos de linguagens *server-side*:

- Banco de dados como SQL, MySQL, Oracle...
- PHP (*Hypertext Preprocessor*)
- ASP (*Active Server Pages*)
- ASP.NET (versão atual do ASP)

Agora você deve estar se perguntando como acontece o fluxo de pedido/resposta e as diferenças quando acessamos uma página estática e uma página dinâmica.

Fluxo pedido/resposta quando se acessa uma página estática

Num servidor *web* existe um arquivo chamado *olamundo.html* com o seguinte conteúdo:

```
<html>
  <head>
    <title>Título</title>
  </head>
  <body>
    <p>olá mundo</p>
  </body>
</html>
```

Quando o *browser* iniciar o procedimento de solicitação ao servidor web a sequência de instruções produzidas será esta (W3C, 2015):

```
GET /olamundo.html HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-shockwave
-flash, application/vnd.ms-excel, application/vnd.ms-powerpoint, application/msword,
*/* application/x-pdf */*
Accept-Language: en-gb,pt;q=0.5
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 1.1.432
2; .NET CLR 2.0.50727)
Host: localhost:8080
Connection: Keep-Alive
Cookie: infoview_userCultureKey=useBrowserLocale
```

Ao receber a requisição a resposta gerada pelo servidor *web* será (W3C, 2015):

```
HTTP/1.1 200 OK
Date:Sat, 05 Sep 2015 19:53:40 GMT
Server:Apache/2.0.59 (Unix) PHP/5.2.6 mod_ssl/2.0.59 OpenSSL/0.9.8
X-Powered-By: PHP/5.2.6
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Set-Cookie: PHPSESSID=c38d9f2812d2208aa983d954c28fc798; path=/
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Connection: close
Content-Type: text/html; charset=ISO-8859-1

<html>
   <head>
     <title>Título</title>
   </head>
    <body>
         <p>olá mundo</p>
   </body>
</html>
```

Devemos ainda lembrar que todo o código entre as tags e será interpretado no navegador.

Fluxo pedido/resposta quando se acessa uma página dinâmica

Agora suponhamos que no servidor *web* existe um arquivo chamado *numero.php* com o seguinte conteúdo:

```
<html>
    <head>
        <title>Título</title>
    </head>
    <body>
        <p>
            <?php
                    for ($i=1;$i<=10;$i++){
                        echo $i . " ";
                    }
            ?>
        </p>
    </body>
</html>
```

O procedimento de solicitação ao servidor produzirá a sequência de instruções abaixo (W3C, 2015):

```
GET /numero.php HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-shockwave
-flash, application/vnd.ms-excel, application/vnd.ms-powerpoint, application/msword,
*/* application/x-pdf */*
Accept-Language: en-gb,pt;q=0.5
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 1.1.432
2; .NET CLR 2.0.50727)
Host: localhost:8080
Connection: Keep-Alive
Cookie: infoview_userCultureKey=useBrowserLocale
```

Ao receber a requisição a resposta gerada pelo servidor *web* será (W3C, 2015):

```
HTTP/1.1 200 OK
Date: Tue, 08 Sep 2015 14:11:11 GMT
Server: Apache/2.0.59 (Unix) PHP/5.2.6 mod_ssl/2.0.59 OpenSSL/0.9.8
X-Powered-By: PHP/5.2.6
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Set-Cookie: PHPSESSID=9f90ecc1691fbe28212b4d5d2d240293; path=/
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Connection: close
Content-Type: text/html; charset=ISO-8859-1

<html>
   <head>
         <title>Título</title>
   </head>
   <body>
         <p>1 2 3 4 5 6 7 8 9 10</p>
   </body>
</html>
```

Aqui, vale ressaltar que o código entre será processado no servidor web e retornado ao cliente somente html puro e este será interpretado no navegador-cliente.

Embora os valores possam variar de acordo com o navegador utilizado e com o servidor *web* que responde a este pedido *http*, o conteúdo será praticamente igual:

no pedido

GET /olamundo.html HTTP/1.1 (página estática)

GET /numero.php HTTP/1.1 (página dinâmica)

ou seja, estamos utilizando o protocolo *http* em sua versão 1.1, e os arquivos solicitados *olamundo.html* e *numeros*.php estão armazenados na raiz do servidor.

na resposta

HTTP/1.1 200 OK

Significa que o pedido é válido (200 OK) e os conteúdos seguem abaixo, obtemos a mesma resposta tanto na página estática, quanto na página dinâmica. Como podemos ver, depois de um conjunto de dados aparece finalmente o conteúdo HTML da página que tínhamos solicitado.

Independentemente de estarmos falando de páginas dinâmicas ou páginas estáticas, este será sempre o fluxo que o pedido/resposta entre o servidor e o *browser* irá provocar.

Quiz

Exercício Final

Tecnologias Client-Server

INICIAR

Referências

MOZILLA DEVELOPER NETWORK. **CSS**. Disponível em <https://developer.mozilla.org/pt-BR/docs/Web/CSS>. 2015b. Acessado em 08/09/15, às 10hmin.

MOZILLA DEVELOPER NETWORK. **HTML**. Disponível em <https://developer.mozilla.org/pt-BR/docs/Web/HTML>. 2015a. Acessado em 08/09/15, às 10h.

MOZILLA DEVELOPER NETWORK. **Javascript**. Disponível em <https://developer.mozilla.org/pt-PT/docs/Web/JavaScript>. 2015c. Acessado em 08/09/15, às 11h.

PALMEIRA, Thiago Vinícius Varallo. **Como funcionam as aplicações web**. Disponível em <http://www.devmedia.com.br/como-funcionam-as-aplicacoes-web/25888>, acessado em 08/09/15, às 15h.

SANTOS, Richard. **Client-Side e Server-Side**. Disponível em <https://richardoliveira.wordpress.com/2010/03/22/client-side-e-server-side/>, acessado em 08/09/15, às 09h40min.

W3C. **Header Field Definitions**. Disponível em: <http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html>, acesso em 09/09/2015, às 13h20min.

Avalie este tópico

ANTERIOR
Servidor Web

Índice

PRÓXIMO
Conceitos Básicos da Linguagem PHP

Ajuda?
(https://ava.uninove.br/seu/AVA/topico/topico.php?idCurso=)