Autoria de Elton Luís Minetto

19 de July de 2005

Nas últimas semanas algumas tecnologias de desenvolvimento para o ambiente Web tem sido comentados, como Ajax, Ruby on Rails e mais recentemente Django. Resolvi testar alguns deles para entender o motivo de tanta empolgação. O primeiro que testei foi o Ajax.
Pelo que entendi não é uma tecnologia nova mas desde que o Google começou a usá-la no Gmail ganhou os holofotes. Ele usa uma mistura de html, javascript e css possibilitando a criação de aplicações que aparentam estar sendo executadas localmente na máquina do usuário e não em um ambiente Web.

Um exemplo simples seria a apresentação da descrição de um ítem após a digitação do seu código em um campo text de um formulário html, sem aparentar para o usuário que a página realizou uma requisição ao servidor para buscar a informação. Implementei este exemplo para entender melhor. Usei uma biblioteca chamada Sajax que facilita o uso do Ajax gerando grande parte do código Javascript.

Ela está disponível neste link (http://absinth.modernmethod.com/sajax/) e pode ser usada em PHP, Python, Ruby entre outras linguagens.

Após ter feito o download do Sajax e criado a tabela pessoa no MySQL fiz o seguinte código:

```
<xmp>
<?
//faz a inclusão da biblioteca Sajax
require("include/Sajax.php");

// Baseado nos exemplos desenvolvidos por Leonardo Lorieri

/* funcao PHP que recebe o código e faz a pesquisa no banco de dados retornando o nome*/
function mostra_nome($codpes) {
	$con = mysql_connect("localhost","elton","elton");
	mysql_select_db("elton");
	$res = mysql_query("select nompes from pessoa where codpes=$codpes",$con);
	$row = mysql_fetch_object($res);
	$nompes = $row->nompes;
	mysql_close($con);
	return $nompes;
}

$sajax_request_type = "GET"; //forma como os dados serao enviados
sajax_init(); //inicia o SAJAX
sajax_export("mostra_nome"); // lista de funcoes a ser exportadas
sajax_handle_client_request();// serve instancias de clientes

?>
<html>
<head>
<title>Nome </title>
<script>
<?
sajax_show_javascript(); //gera o javascript
?>
function mostra(nome) { //esta funcao retorna o valor para o campo do formulario
	document.teste.nompes.value=nome;
}

function get_nome(c) { //esta funcao chama a funcao PHP exportada pelo Ajax
	cod = c.value;
	//chama a funcao x_mostra_nome que será gerada pelo sajax. o primeiro parametro é o codigo e o
segundo é a
	//funcao JavaScript que tratara o retorno, no caso a mostra
	x_mostra_nome(cod, mostra);
}
</script>
</head>
<body>
<form name="teste">
<input type="text" name="codpes" onchange="get_nome(this)">
<input type="text" name="nompes">
</form>
</body>
</html>
</xmp>
```

Analisando o código fonte visualizado pelo navegador é possível verificar todo o código Javascript gerado pelo Sajax:

```
<xmp>
<html>
<head>
<title>Nome </title>
<script>
// remote scripting library
// (c) copyright 2005 modernmethod, inc
var sajax_debug_mode = false;
var sajax_request_type = "GET";

function sajax_debug(text) {
	if (sajax_debug_mode)
	alert("RSD: " + text)
}
function sajax_init_object() {
	sajax_debug("sajax_init_object() called..")
	var A;
	try {
		A=new ActiveXObject("Msxml2.XMLHTTP");
	} catch (e) {
		try {
			A=new ActiveXObject("Microsoft.XMLHTTP");
		} catch (oc) {
			A=null;
		}
	}
	if(!A && typeof XMLHttpRequest != "undefined")
	A = new XMLHttpRequest();
	if (!A)
	sajax_debug("Could not create connection object.");
	return A;
}

function sajax_do_call(func_name, args) {
	var i, x, n;
	var uri;
	var post_data;

	uri = "/saa/ajax.php";
	if (sajax_request_type == "GET") {
		if (uri.indexOf("?") == -1)
		uri = uri + "?rs=" + escape(func_name);
		else
		uri = uri + "&rs=" + escape(func_name);
		for (i = 0; i < args.length-1; i++)
		uri = uri + "&rsargs[]=" + escape(args[i]);
		uri = uri + "&rsrnd=" + new Date().getTime();
		post_data = null;
	} else {
		post_data = "rs=" + escape(func_name);
		for (i = 0; i < args.length-1; i++)
		post_data = post_data + "&rsargs[]=" + escape(args[i]);
	}

	x = sajax_init_object();
	x.open(sajax_request_type, uri, true);
	if (sajax_request_type == "POST") {
		x.setRequestHeader("Method", "POST " + uri + " HTTP/1.1");
		x.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
	}
	x.onreadystatechange = function() {
		if (x.readyState != 4)
		return;
		sajax_debug("received " + x.responseText);

		var status;
		var data;
		status = x.responseText.charAt(0);
		data = x.responseText.substring(2);
		if (status == "-")
		alert("Error: " + data);
		else
		args[args.length-1](data);
```

```
	}
	x.send(post_data);
	sajax_debug(func_name + " uri = " + uri + "/post = " + post_data);
	sajax_debug(func_name + " waiting..");
	delete x;
}


// wrapper for mostra_nome
function x_mostra_nome() {
	sajax_do_call("mostra_nome",
	x_mostra_nome.arguments);
}

function mostra(nome) { //esta funcao retorna o valor para o campo do formulario
	document.teste.nompes.value=nome;
}

function get_nome(c) { //esta funcao chama a funcao PHP exportada pelo Ajax
	cod = c.value;
	//chama a funcao x_mostra_nome que será gerada pelo sajax. o primeiro parametro é o codigo e o
segundo é a
	//funcao JavaScript que tratara o retorno, no caso a mostra
	x_mostra_nome(cod, mostra);
}
</script>
</head>
<body>
<form name="teste">
<input type="text" name="codpes" onchange="get_nome(this)">
<input type="text" name="nompes">
</form>
</body>
</html>

</xmp>
```

Realmente a utilização do Sajax facilita bastante a programação. Utilizando-se Ajax é possível criar interfaces muito ricas e complexas.
Alguns sites interessantes usando esta linguagem:

http://blog.joshuaeichorn.com/slides/Building_Rich_Web_Applications_With_AJAX/

http://maps.google.com/

http://blog.outer-court.com/chat/
http://gosu.pl/docs/
http://www.ajaxian.com/

Nos proximos artigos irei falar sobre os testes que fiz com o Django, um framework em Python, e com o PHP on Trax, uma implementação em PHP de algumas idéias do Ruby on Rails.