# Dynamische HTML-Seiten für den Omnis WebClient

## Problemstellung

Sie fragen sich vielleicht wozu brauche ich noch dynamische Web-Seiten wenn ich die moderne Omnis WebClient-Technologie verwende? Schließlich kann ich in einem Remote Form mit Subforms das Aussehen für den Anwender in programmgesteuert beliebig dynamisch und sehr flexibel modifizieren.

Auf dem Web-Server benötigt man doch lediglich eine HTML-Seite (Name z.B. **webclient.htm**) mit etwa folgendem Inhalt:

```
<html>
<head>
<title>Test Omnis Web-Client</title>
</head>
<body>
<h1>Test Omnis Studio Web-Client Application</h1>
<!-- Use for Microsoft Internet Explorer -->
<!-- ActiveX object START -->
<object classid="clsid:13510606-30FA-11D2-B383-444553540000"
  width="360" height="240">
  <param name="_Version" value="65536">
  <param name="_ExtentX" value="7161">
  <param name="_ExtentY" value="7373">
  <param name="_StockProps" value="0">
  <param name="OmnisServer" value="192.168.0.8:5912">
  <param name="OmnisLibrary" value="WEBCLIENT">
  <param name="OmnisClass" value="rfTest">
  <param name="WebServerURL" value="http://192.168.0.2">
  <param name="WebServerScript" value="/cgi-bin/nph-omniscgi">
<!-- Use for Netscape, Firefox and others (Windows, Linux and Mac) -->
<!-- Netscape object START-->
<embed type="application/OMNIS-RCC-plugin" name="rcc1" width="360"
   height="240" OmnisServer="192.168.0.8:5912" OmnisLibrary="WEBCLIENT"
   OmnisClass="rfTest" WebServerUrl="http://192.168.0.2"
   WebServerScript="/cgi-bin/nph-omniscgi">
<!-- Netscape object END-->
</object>
<!-- ActiveX object END-->
<p> </p>
</body>
</html>
```

Welchen Vorteil kann es dann noch bringen, eine derartige HTML-Seite dynamisch zu erzeugen?

Es gibt gleich mehrere Gründe diese Seite dynamisch, z.B. mit PHP zu erzeugen:

1. Die Erstellung einer derartigen Web-Seite wird einfacher und der Code wird schlanker.
2. Die Web-Seite lässt sich leichter umkonfigurieren.
3. Es lassen sich dynamische HTTP-Parameter an die Omnis-Applikation übergeben.
4. Wenn der dahinterliegende Omnis Application-Server off-line ist, kann man eine aussagekräftige Fehlermeldung erhalten.

Es geht also nicht darum, die Web-Applikation selbst mit PHP zu erzeugen.

Besonders der Punkt 2. war für uns im technischen Support in Hamburg der Anlass, hier etwas zu entwickeln: Wir testen häufig Web-Applikationen auf den verschiedensten Rechnern und müssen dazu jeweils die HTML-Seite modifizieren. Da der Code für den eingebetteten WebClient für Microsoft's Internet Explorer anders ist als für alle anderen Browser, müssen alle Änderungen (z.B. Höhe, Breite und IP-Adresse und Port des Omnis Servers), immer zweimal in die obige HTML-Seite eingepflegt werden.

## 1. Einfache Erstellung der Web-Seite

Wir haben das Problem mit einem PHP-Skript gelöst, also eigentlich mit zwei PHP-Skripten: Einer recht umfangreichen Klassendefinition in PHP und einem sehr schlanken PHP-Skript das die HTML-Seite ersetzt. Der in der obigen Seite blau markierte Text wird dann einfach durch den folgenden PHP-Code ersetzt:

```
<?php
  $phplib = dirname(__FILE__) . '/../phpsub';
  set_include_path(get_include_path() . PATH_SEPARATOR . $phplib);
  require_once('OmnisWebClass.php');
  $x = new OmnisWebClass('webclient.ini');
  // Insert here Extensions for parameters and / or semaphores
  $x->display_plugin();
?>
```

Außerdem muss die Seite **webclient.htm** in **webclient.php** umbenannt werden. Dann wird der Web-Server den ganzen Code zwischen **<?php** und **?>** durch einen PHP-Parser und –Interpreter schicken und das Ergebnis dieses Prozesses an den Client senden. Der Anwender sieht dann – wenn er sich den Quellcode der Seite anschaut – keinen Unterschied zwischen der statischen Seite **webclient.htm** und der dynamischen Seite **webclient.php**.

Dieses Gerüst gilt für alle Seiten. Es muss nur möglicherweise die zweite Zeile modifiziert werden: Hier ist angenommen, dass der Web-Server auf einem Linux-PC läuft und die Hilfs-Klasse **OmnisWebClass.php** ins Verzeichnis **/srv/www/htdocs/phpsub/** gelegt wurde. Nur diese Zeile muss man modifizieren, wenn man z.B. einen IIS unter Windows als Web-Server verwendet.

Die für die jeweilige Applikation spezifischen Eigenschaften sind in eine separate Datei **webclient.ini** ausgelagert, die sich im gleichen Verzeichnis wie die **webclient.php** befinden muss. Sie hat zum Beispiel den folgenden Inhalt:

```
WebServerURL = http://192.168.0.2
WebServerScript = /cgi-bin/nph-omniscgi
OmnisServer = 192.168.0.8
OmnisPort = 5912
OmnisLibrary = WEBCLIENT
OmnisClass = rfTest
OmnisWidth = 360
OmnisHeight = 240
; enable the next line for a Studio 5.x app server
; OmnisUnicodeServer = 1
```

In der Codezeile

```
$x = new OmnisWebClass('webclient.ini');
```

wird eine Instanz **$x** der Klasse **OmnisWebClass.php** erzeugt und dem Konstruktor wird der Name der .ini-Datei übergeben. Die folgende Codezeile

```
$x->display_plugin();
```

erzeugt dann den Code zur Darstellung des Plugins. Der Aufruf diese Internet-Seite erfolgt dann z.B. durch Eingabe von

```
http://192.168.0.2/test/webclient.php
```

in die Adress-Zeile des Browsers.

## 2. Einfaches Umkonfigurieren

Wenn zum Beispiel das Remote Form eine andere Größe bekommen soll, so muss der entsprechende Parameter in der .ini-Datei lediglich einmal modifiziert werden. Automatisch wird das dann sowohl in den Code für den Internet-Explorer als auch für Netscape, Firefox & Co. eingebaut.

Beim Testen mit der Omnis Entwicklungsumgebung laufen Internet-Browser und Omnis Application-Server auf dem gleichen PC. Dann kann man die Zeile **OmnisServer = ...** auch weglassen. PHP analysiert den HTTP-Request des Browsers und ermittelt u.a. die IP-Adresse des Clients. Wenn in der .ini-Datei keine IP-Adresse für den OmnisServer angegeben ist, setzt PHP die Adresse des Clients ein. Damit können Entwickler zum Testen von unterschiedlichen PCs aus die gleiche Web-Adresse verwenden.

Die Parameter OmnisXXX der .ini-Datei können auch durch cgi-Parameter überschrieben werden. Wenn man im Browser die URI

```
http://192.168.0.2/test/webclient.php?lib=TEST&class=rfTest_2&width=200
```

eingibt, so erzeugt PHP eine Seite, in der die Omnis-Parameter so gesetzt sind, dass das Remote Form rfTest_2 aus der Library TEST mit einer Breite von 200 Pixeln angezeigt wird. Alle noch fehlenden Parameter werden der .ini-Datei entnommen. Dies ist im Detail in der **OmnisWebClass.php** dokumentiert.

## 3. HTTP-Parameter

Bei der Analyse des HTTP-Requests kann PHP aus dem HTTP-Header auch weitere Parameter, z.B. den Typ des Browsers beim Client ermitteln. Dies kann man als zusätzlichen Parameter dem Remote Task in Omnis übergebeben. Siehe Omnis Programming Manual, Kapitel “Deploying Your Omnis Web Application” / “Manually editing your HTML Pages”. In der HTML-Seite würde man dann zusätzlich eingefügen

```
<param name="Param1" value="myParameter=myValue">
```

für den Internet-Explorer bzw.

```
Param1="myParameter=myValue"
```

für Netscape & Co.. Die Klasse **OmnisWebClass.php** ist dafür vorbereitet, man muss lediglich nach der Instanzierung des PHP-Objektes, unmittelbar vor der Zeile

```
$x->display_plugin();
```

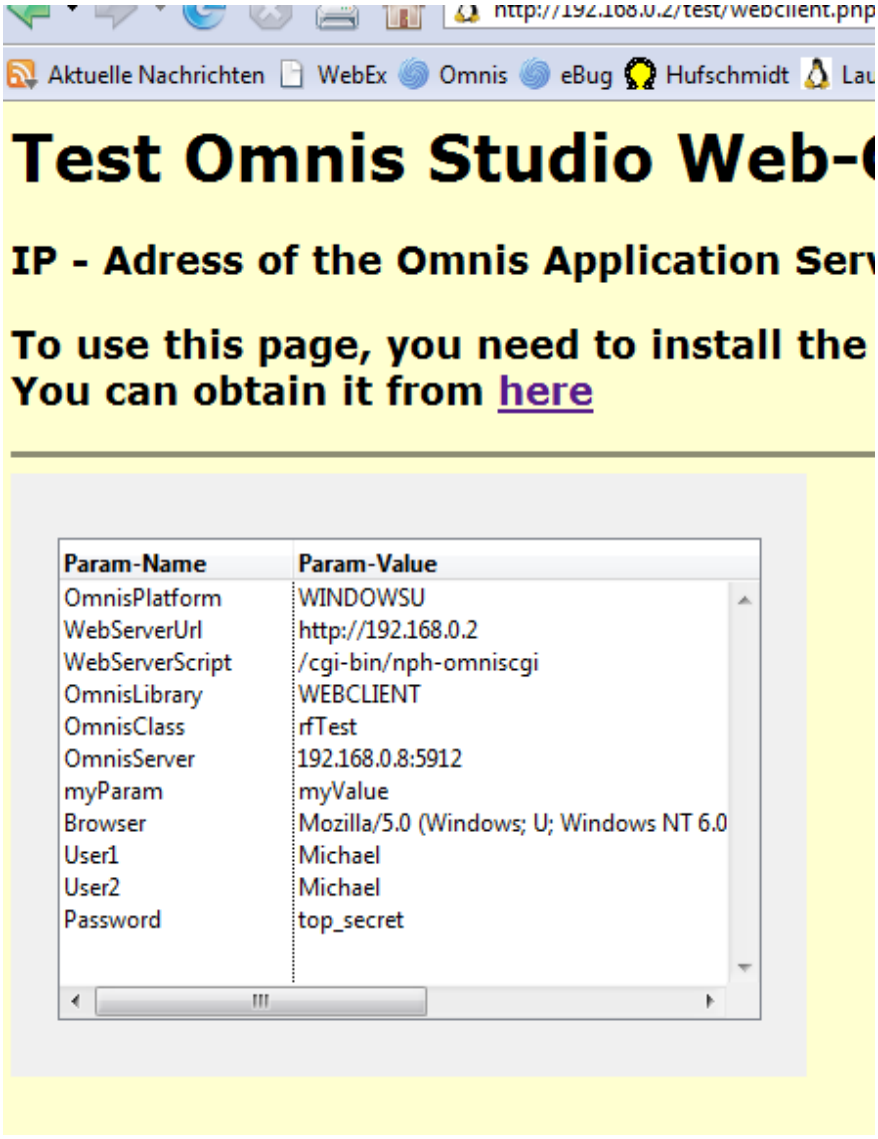
ein entsprechendes Array in der **webclient.php** deklarieren, füllen und an die Instanz übergeben:

```
$params = array ();
$params['myParam'] = 'myValue';
$params['Browser'] = $_SERVER['HTTP_USER_AGENT'];
// insert here further parameters, see below
$x->set_additional_params($params);
```

Wenn der Zugriff auf die Seite mit .htaccess geschützt ist, kann man sogar den User-Namen und das Passwort an Omnis übergeben. Dazu dient der PHP-Code:

```
if (isset($_SERVER['REMOTE_USER']))
  $params['User1'] = $_SERVER['REMOTE_USER'];
if (isset($_SERVER['PHP_AUTH_USER']))
  $params['User2'] = $_SERVER['PHP_AUTH_USER'];
if (isset($_SERVER['PHP_AUTH_PW']))
  $params['Password'] = $_SERVER['PHP_AUTH_PW'];
```

Allerdings funktioniert das nicht, wenn „safe mode restrictions“ für PHP eingestellt sind. Hier ein Beispiel für eine Remote-Form, das aus einem geschützten Verzeichnis aufgerufen wird und die Parameter des Remote Tasks in einer Headed Listbox anzeigt:




## 4. Server Off-Line - Erkennung

Ein Omnis Application Server läuft in der Regel sehr stabil, aber auch der muss mal gewartet werden. Wenn der Server off-line ist und man ruft dann im Browser die Web-Seite mit dem Omnis-Plugin auf, dann friert der Browser für einige Minuten ein und meldet schließlich „Error reading the response from the WEB server“. Die häufigste Ursache dafür ist übrigens – nach Erfahrungen des technischen Supports in Hamburg – dass der Port im Omnis Application-Server nicht richtig konfiguriert wurde.

Mit der Klasse **`OmnisWebClass.php`** und kleinen Ergänzungen in der Omnis Library lässt sich das vermeiden. Voraussetzung ist, dass der Omnis Application-Server in das Dateisystem des Web-Servers schreiben kann. Außerdem müssen wieder einige „safe mode restrictions“ in PHP aufgeweicht werden. Dann muss der Omnis Application-Server unmittelbar nach dem Öffnen der Library eine Datei z.B. mit Namen **`Semaphore_WEBCLIENT.txt`** im dem Verzeichnis erzeugen, in dem die **`webclient.php`** liegt. Beim Schließen der Library muss diese Datei wieder gelöscht werden.

Die Instanz von **`OmnisWebClass.php`** kann dann prüfen, ob die Datei vorhanden ist. Falls nicht, wird nicht der Code für das Omnis Plugin ausgegeben, sondern ein Hinweis „<h3>Omnis App. Server with Library "WEBCLIENT" is off-line!</h3>”, der genaue Text lässt sich auch anders konfigurieren. Um dieses Verhalten einzustellen, muss lediglich in der **`webclient.php`** eine Zeile hinzugefügt werden, die der Instanz den Namen der Semaphore-Datei übermittelt:

```
$x->set_semaphore('Semaphore_WEBCLIENT.txt');
```

## 5. Omnis JavaScript Client

Seit Studio 5.2 kann man JavaScript basierte Remote Forms erzeugen, die im Browser des Clients ausgeführt werden. Eine typische HTML-Seite für einen JS Client wäre etwa:

```
<html>
<head>
<title>Studio 5.2 Test</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<!-- This makes mobile device support work -->
<meta name="viewport" content="width=device-width">
<!-- The style sheets must be present in the directory /css/ -->
<link type="text/css" href="/css/omn_dlg.css" rel="stylesheet"
  media="print,screen" />
<link type="text/css" href="/css/omn_menu.css" rel="stylesheet"
  media="print, screen" />
<link type="text/css" href="/css/smoothness/jquery-ui-1.8.15.custom.css"
  rel="stylesheet" />
<link type="text/css" href="/css/slick.grid.css" rel="stylesheet" />
<link type="text/css" href="/css/slick.columnpicker.css" rel="stylesheet"
  />
<link type="text/css" href="/css/slick.pager.css" rel="stylesheet" />
<!-- Must occur after other stylesheets e.g. jquery-ui-1.8.15.custom.css,
  as it overrides values -->
<link type="text/css" href="/css/omnis.css" rel="stylesheet"/>
<!-- Omnis Studio JavaScript client scripts, must be present in the
  directory /scripts/ -->
<script type="text/javascript" src="/scripts/omjsclnt.js"></script>
<script type="text/javascript" src="/scripts/omjqclnt.js"></script>
</head>
<body onload="jOmnis.onLoad()" onunload="jOmnis.onUnload()"
  style="margin:0;">
<h1>Test JavaScript Client Studio 5.2</h1>
<div id="omnisobject1" WebServerUrl="http://192.168.0.4/omnis_apache"
  OmnisServerAndPort="192.168.0.8:5914" OmnisLibrary="ScriptClient52_Test"
  OmnisClass="rfTest" param1="A1" param2="B1" param3="C1"
  param4="D1"></div>
</body>
</html>
```

Mit dem gleichen .ini-File wie für den Web-Client, kann man die blau markierten Zeilen mit php erzeugen. Eine komplette PHP / HTML-Seite für einen JavaScript Client wäre dann etwa:

```
<html>
<head>
<title>Studio 5.2 Test</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<?php
  $phplib = dirname(__FILE__) . '/../phpsub'
  set_include_path(get_include_path() . PATH_SEPARATOR . $phplib);
  require_once('OmnisWebClass.php');
  $x = new OmnisWebClass('ScriptClient52_Test.ini', 'rfTest');
  $params = array ("A1", "B1", "C1", "D1");
  $x->set_additional_params($params);
  $x->display_header_body_omnis_js('/css','/scripts','margin:0;');
?>
<h1>Test JavaScript Client Studio 5.2</h1>
<?php
  $x->display_client_omnis_js();
?>
</body>
</html>
```

Es lassen sich verschiedene Optionen und Parameter setzen, Details sind in den Kommentarzeilen von **OmnisWebClass.php** ab Zeile 185 beschrieben.

Insbesondere kann man absolute Pfadnamen für die css und scripts Verzeichnisse setzen.

## Programmpaket

Die Klasse **`OmnisWebClass.php`** enthält 650 Programmzeilen, mehr als ein Drittel davon sind aber Kommentare und Erläuterungen zur Anwendung. Wir verzichten darauf, sie hier abzudrucken, für unsere Kunden haben wir alle erforderlichen Dateien in einem .zip-Archiv „**`DynamicWebPages.zip`**“ zusammengestellt. In dem Archiv ist auch diese Dokumentation, die Beispiel-Library **`webclient.lbs`** (für Studio 4.3.1 non-unicode) und die Dateien **`webclient.ini`** und **`webclient.php`** vorhanden. Interessenten können dieses Archiv beim Technischen Support Hamburg unter ge_support@omnis.net anfordern.

File:
**`\\SATURN\Public\SupportFAQs\Techtips\Im_Newsletter_verwendet\DynamischeWebSeiten_de.doc`**

*Michael Hufschmidt*