Ejemplo de mono

Aguilar López Dulce María
Avila Mora Ivonne Maricela
Covarrubias Flores Esmeralda
Hernández García Héctor Daniel
Leyto Delgado Karina
López Escogido Daniel
Ruíz García Luis Felipe

INSTALACION DE MONO EN UBUNTU

La instalción de Mono es muy sencilla, sobre todo en ubuntu, ya que tenemos la posibilidad de utilizar el gestor de paquetes Synaptic.

Es necesrio instalar los siguientes paquetes:

- mono
- monodoc
- monodevelop
- libapache2-mod-mono ó libapache-mod-mono

EJEMPLO CON C#

Uno de los campos más interesantes de la plataforma .Net es ASP.Net y los servicios web. Mono no iba a ser menos y tiene su versión de ASP.Net, denominada XSP, con soporte para servicios web XML.

XSP provee de un servidor web mínimo escrito en C# en el cual los servicios web pueden ser corridos y probados usando la librería System.Web que facilita Mono. Este servidor es conveniente para probar sitios pequeños.

Para obtenerlo en Ubuntu solo debemos obtener el paquete mono-xsp. Una forma de hacerlo sería:

```
$apt-get install mono-xsp
```

EJEMPLO CON C#

Los servicios web se deben de escribir en ficheros con extensión .asmx para que XSP sea capaz de tratarlos como tal. Un ejemplo simple de servicio web podría ser el siguiente, el cual convierte de grados Farenheit a Celsius.

```
<%@ WebService Language="C#" Class="Converter" %>
using System.Web.Services;

[WebService(Namespace="http://www.monohispano.org/",
Description="ConvertidorF-C.")]

public class Converter{
    [WebMethod(Description="Convierte de grados farenheit a
     celsius.")]
    public double ConvertirFC(double dFahrenheit){
        return ((dFahrenheit - 32) * 5) / 9;
    }
}
```

EJEMPLO CON C#

Este archivo podemos guardarlo como converter.asmx, por ejemplo. Para probarlo solo debemos de ejecutar xsp en el mismo directorio en el que se encuentre nuestro fichero y acceder a través de un navegador.

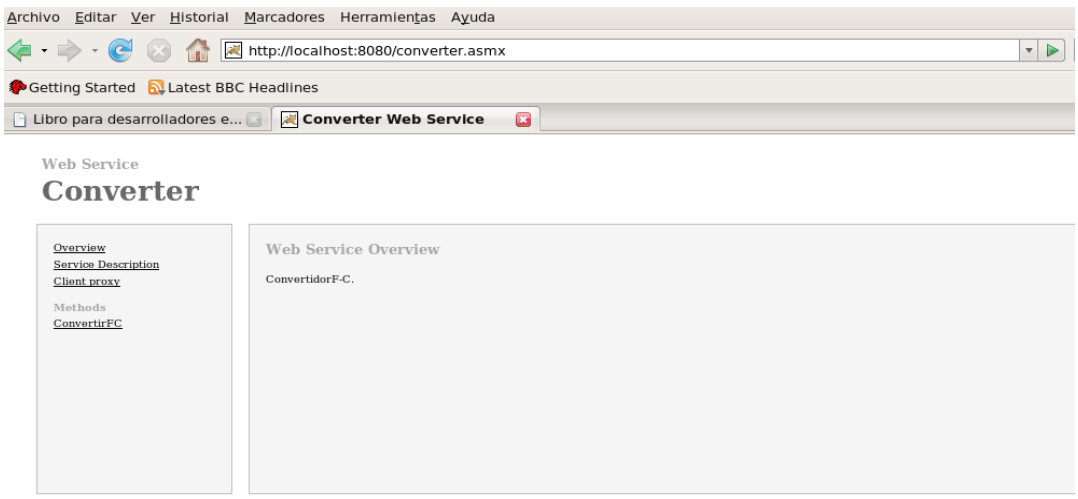
```
$ xsp
xsp
Listening on port: 8080 (non-secure)
Listening on address: 0.0.0.0
Root directory: /tarea11-mono/final
Hit Return to stop the server.
```

Por defecto xsp se lanza en el puerto 8080, así que lo normal será acceder a:

http://localhost:8080/converter.asmx

EJEMPLO CON C#

El xsp ha creado una página con la descripción de los servicios web que hemos escrito, generación de código en linea para usar el proxy, e información de parámetros de los métodos exportados así como los valores que devuelven.



EJEMPLO CON C#

Ahora, con el comando wsdl convertiremos el XML correspondiente a la descripción de los métodos a código compilable.

```
$ wsdl http://localhost:8080/converter.asmx
Writing file 'Converter.cs'
```

El archivo Converter.cs contiene entre otras cosas:

```
[System.Web.Services.WebServiceBinding(Name="ConverterSoap",
Namespace="http://www.monohispano.org/")]
[System.Diagnostics.DebuggerStepThroughAttribute()]
[System.ComponentModel.DesignerCategoryAttribute("code")]
public class Converter : System.Web.Services.Protocols.SoapHttpClientProtocol {
    public Converter() {
        this.Url = "http://localhost:8080/converter.asmx";
    }
...
ParameterStyle=System.Web.Services.Protocols.SoapParameterStyle.Wrapped,
Use=System.Web.Services.Description.SoapBindingUse.Literal)]
    public System.Double ConvertirFC(System.Double dFahrenheit) {
        object[] results = this.Invoke("ConvertirFC", new object[] {
            dFahrenheit});
        return ((System.Double)(results[0]));
    }
...
```

EJEMPLO CON C#

Para compilarlo necesitaremos usar la librería de servicios web:

```
$ mcs  -r:System.Web.Services Converter.cs -target:library
```

-r: Indica que debe ligar las librerías de Web Services que se encuentran en System.
-target: Indica el tipo de archivo que deseamos crear, exe, library, modulo o winexe.

Esto nos habrá generado Converter.dll, que nos permitirá acceder a los métodos de forma remota.

EJEMPLO CON C#

Ahora probaremos el servicio web creado escribiendo un sencillo programa que llame al método para obtener la conversión.

```
using System;
class Test{
    public static void Main(string[] args){
      Converter e = new Converter();
      Console.WriteLine(" Grados farenheit " +
                              Double.Parse(args[0])+ " °F");
      double res = e.ConvertirFC(Double.Parse(args[0]));
      Console.WriteLine(" Resultado de la conversion " +
                            res + " °C");
    }
}
```

EJEMPLO CON C#

Este archivo Test.cs lo compilaremos asi:

```
$ mcs Test.cs -r:Converter.dll
```

Con -r indicamos que debe ligar con la librería Converter.dll y nos generará Test.exe (tendrá extensión exe aunque trabajemos en Linux).

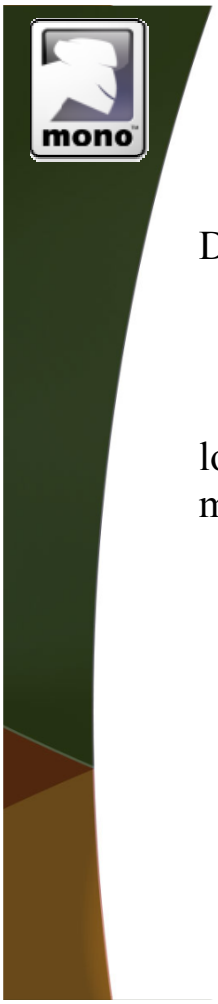
Ejecutamos ahora el .exe pasando por parámetro los grados Farenheit:

```
$ mono Test.exe 150
      Grados farenheit 150 °F
      Resultado de la conversion 65.5555555555556 °C
```

EJEMPLO CON VISUAL BASIC .NET

El código de nuestro servicio web sería el siguiente:

```
Imports System.Web
Imports System.Web.Services
Imports System.Web.Services.Protocols
<System.Web.Services.WebService(Namespace:="http://localhost/vb/")> _
<WebServiceBinding(ConformsTo:=WsiProfiles.BasicProfile1_1)> _
<Global.Microsoft.VisualBasic.CompilerServices.DesignerGenerated()> _
Public Class Service Inherits System.Web.Services.WebService
    <WebMethod(Description:="This method converts a temperature in
        degrees Fahrenheit to a temperature in degrees Celsius.")> _
    Public Function ConvertTemperature(ByVal dFahrenheit As Double)
            As Double
        Return ((dFahrenheit - 32) * 5) / 9
     End Function
End Class
```



EJEMPLO CON VISUAL BASIC .NET

Debemos de compilarlo mediante

```
$mbas -r:System,System.Web,System.Web.Services
            Service.vb -t:library
```

lo cual nos generaría una dll, llamado Service.dll que será el motor de nuestro servicio web.

EJEMPLO CON VISUAL BASIC .NET

Las aplicaciones ASP.NET tienen la característica que necesitan una carpeta llamada bin dentro de la ruta del acceso de la URL, esta carpeta contendrá todas las aplicaciones compiladas, es decir las dlls necesarias para hacer funcionar nuestra aplicación, aunque también es posible que funcionen sin ser compiladas.

Crearemos un archivo llamado index.asmx con el siguiente contenido, sin importar el lenguaje en el cual fue escrito:

```
<%@ WebService Class="Service" %>
```

EJEMPLO CON VISUAL BASIC .NET

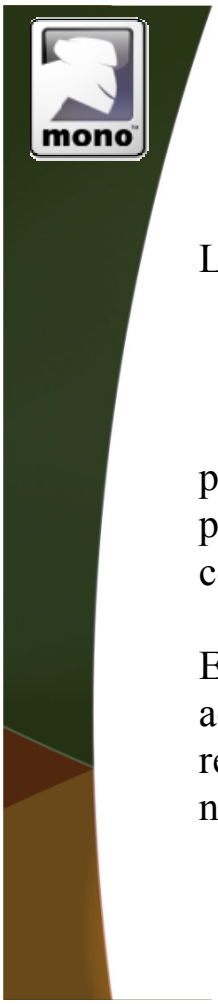
Ahora lo único que falta es arrancar nuestro servidor XSP para hacer funcionar al servicio web, para hacerlo necesitamos ejecutar

```
$xsp --applications /:.
```

de esta forma ya podremos acceder directamente a nuestro servicio web a través de http://localhost:8080/index.asmx

Por lo tanto nuestro arbol de archivos sería algo así:

```
/directorio-donde-esta-el-webservice/
/directorio-donde-esta-el-webservice/bin/
/directorio-donde-esta-el-webservice/bin/Service.dll
/directorio-donde-esta-el-webservice/index.asmx
```



EJEMPLO CON VISUAL BASIC .NET

La ejecución será de la siguiente forma:

```
$ wsdl http://localhost:8080/index.asmx?wsdl -out:
  ServiceProxy.vb -n:ServiceProxy -l:VB
```

para de esta forma no reemplazar el viejo Service.cs, con el parámetro -n se indica un nuevo namespace para que no haya conflicto con el actual.

Este nuevo archivo, ServiceProxy.vb, contiene todos los métodos accesibles del Servicio Web en Visual Basic .NET. Lo único que resta es correr la aplicación con el comando $mono nombre_aplicacion.