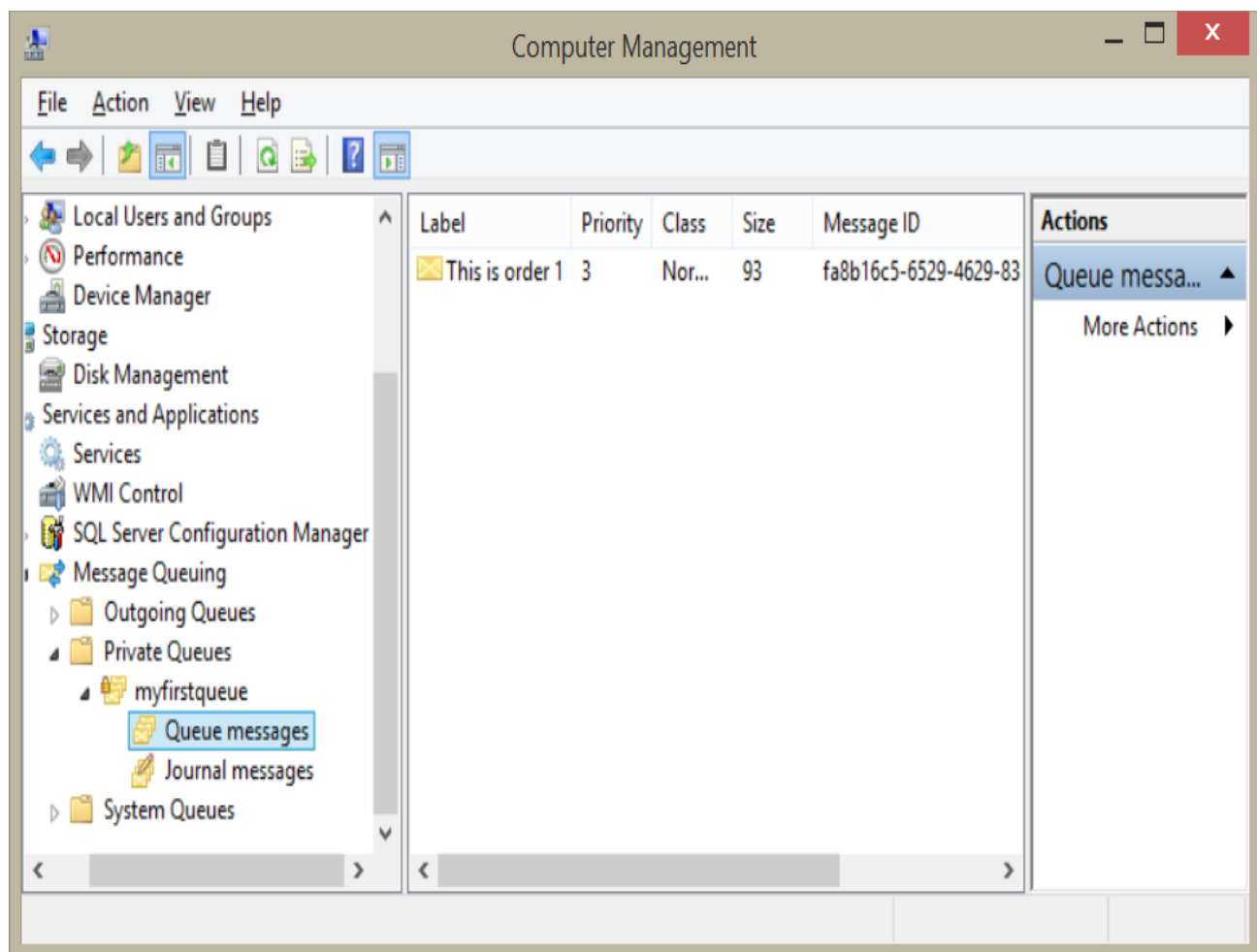


## Utilizando Message Queue con C# en .NET Parte II

por Martín A. O Márquez <xomalli@gmail.com>

Continuando con el tema de **Microsoft Message Queue**, mostraré como recibir un mensaje almacenado en la cola de espera. En la [primera parte de este tutorial](#) enviamos un mensaje a una cola de espera por lo que ahora mostraré un programa que utilice el método *Receive()* de la clase *MessageQueueHelper* que puse de ejemplo en la primera parte de este tutorial para obtener ese mensaje y mostrarlo.

**Fig 1 Consultando los mensajes almacenados en la cola de mensajes**



Hay que recordar que MSMQ trabaja con un principio **FIFO (First In, First Out)** lo que significa que el primer mensaje en la cola es el mensaje que recuperarás utilizando los métodos *Receive()* o *Peek()* de la clase [MessageQueue](#).

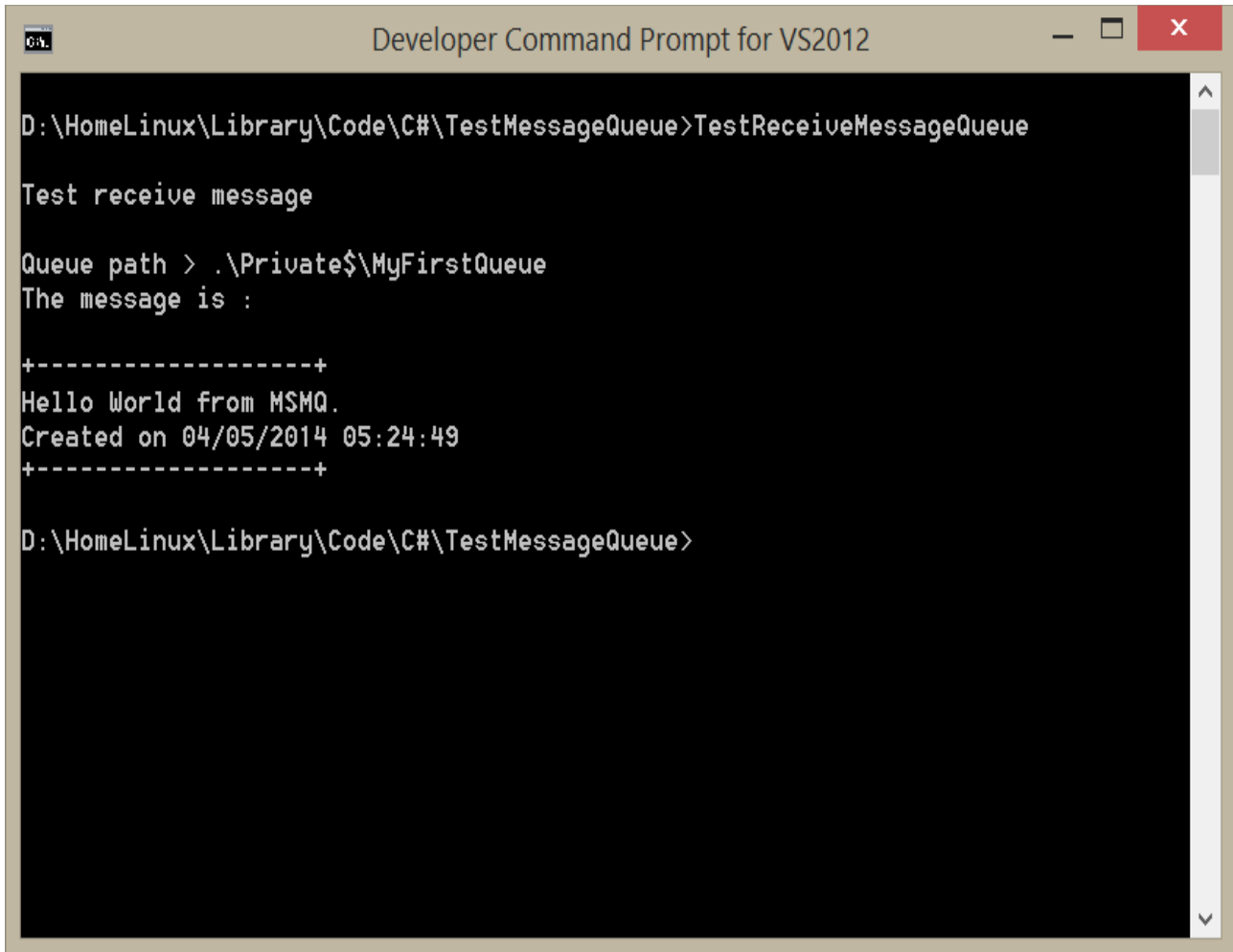
Aquí el código fuente del programa:

```
using System;

namespace Samples{
    public class TestReceiveMessageQueue
    {
        public static void Main(string[] args)
        {
            string message = null;
            try
            {
                Console.WriteLine("\nTest receive message\n");
                Console.Write("Queue path > ");
                string path = Console.ReadLine();
                message = MessageQueueHelper.GetMessage(path);
                Console.WriteLine("The message is : \n");
                Console.WriteLine("+-----+");
                Console.WriteLine("{0}", message);
                Console.WriteLine("+-----+");
            }
            catch(Exception ex)
            {
                Console.WriteLine(ex.Message);
            }
        }
    }
}
```

Al ejecutar el programa se verá la siguiente salida:

**Fig 2 Ejecución del programa, recibiendo el mensaje desde la cola de mensajes y mostrando su contenido en la consola.**



```
D:\HomeLinux\Library\Code\C#\TestMessageQueue>TestReceiveMessageQueue

Test receive message

Queue path > .\Private$\MyFirstQueue
The message is :

+-----+
Hello World from MSMQ.
Created on 04/05/2014 05:24:49
+-----+

D:\HomeLinux\Library\Code\C#\TestMessageQueue>
```

Ahora voy a explicar cómo funcionan los métodos de la clase **MessageQueueHelper**. El método *CreateQueue()* recibe tres parámetros que indican las siguientes propiedades:

1. **Path:** Indica la ubicación de la cola de espera.
2. **Transactional:** Especifica si la cola puede aceptar transacciones de mensajes.
3. **Label:** El título que describe de la cola de espera.
- 4.

Para el valor del Path, el “.” Indica que estamos creando la cola de espera en la maquina local y utilizamos el prefijo *Private\$* para indicar que es una cola de espera privada, si omities este prefijo indica que la cola de mensajes es pública.

```
queue = MessageQueue.Create(queuePath,isTransactional);
queue.Label = label;
```

En el método *SendMessage()* utilizamos le asignamos la clase **XmlMessageFormatter** para dar un tipo de formato al mensaje. Los MessageFormatters implementan la interfaz IMessageFormatter, la cual puede ser aplicada a un mensaje o a toda la cola de espera. Es indispensable que utilices el mismo MessageFormatter tanto para enviar o para recibir el mensaje. Aquí una lista de los Formatters:

1. **ActiveXMessageFormatter**: Se utiliza para trabajar con componentes COM.
2. **BinaryMessageFormatter**: Utilizado para serializar o deserializar un mensaje en un formato binario.
3. **XMLMessageFormatter**: El formatter predeterminado, serializa y deserializa en formato XML.

Aquí está el código donde se utiliza:

```
queue = new MessageQueue(queuePath);  
queue.Formatter = new XmlMessageFormatter();
```

Por último en el método *GetMessage()* igual que para enviar el mensaje utilizo una clase MessageFormatter para deserializar el mensaje, también utilizo el método *Receive()* de la clase **MessageQueue** para obtener el primer mensaje de la cola de forma síncrona, como argumento recibe un **TimeSpan** que representa el intervalo de tiempo de espera para recibir el mensaje, este argumento es importante o de lo contrario si no se recibe el programa se bloqueara hasta que no haya recibido el mensaje.

```
queue = new MessageQueue(queuePath);  
queue.Formatter = new XmlMessageFormatter(new Type[]{  
    Type.GetType("System.String")  
});  
queueMessage = queue.Receive(new TimeSpan(0,0,2));  
message = queueMessage.Body.ToString();
```



**[Download el código fuente](#)**

Este documento está protegido bajo la licencia de documentación libre *Free Documentacion License* del Proyecto GNU, para consulta ver el sitio <http://www.gnu.org/licenses/fdl.txt> , toda persona que lo desee está autorizada a usar, copiar y modificar este documento según los puntos establecidos en la «Licencia FDL»

