

Utilizando Message Queue con C# en .NET

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

Microsoft Message Queue (MSMQ) es un servicio de Windows que permite la comunicación interproceso entre diferentes computadoras de forma asíncrona, entender cómo funciona puede resultar útil en cualquier momento que los clientes y los servidores no tengan una conexión disponible, ya que los mensajes entre ellos son encolados para su transmisión hasta que ambos estén conectados.

El funcionamiento básico de MSMQ es parecido al de una mensajería entre aplicaciones de software, una aplicación prepara un mensaje y lo escribe hacia una línea de espera o cola de mensajes MSMQ. En este contexto una cola de mensajes es un área de almacenamiento temporal que guarda los mensajes hasta que sean transmitidos. El administrador de colas actúa como un intermediario que transmite el mensaje desde su aplicación origen hasta su destino. El propósito de la cola de mensajes MSMQ es el enrutamiento y garantizar la entrega del mensaje, en caso de que el destinatario del mensaje no esté disponible la cola de mensajes mantendrá el mensaje hasta que pueda ser entregado con éxito.

Este mecanismo de transporte proporciona los siguientes beneficios:

1. **Robustez:** Los mensajes son almacenados en la línea de espera hasta que son procesados. Si se cae la conexión entre un cliente y un server los mensajes se guardan hasta que una conexión este presente.
2. **Mensajería basada en prioridad:** Los mensajes más importantes son los primeros en ser recibidos, después los mensajes que le siguen y así sucesivamente, hasta que los últimos son los de prioridad más baja.
3. **Garantía de entrega:** Cuando se llaman los mensajes a través de la red, MSMQ utiliza una transacción para entregar los mensajes, si se lee un mensaje de la línea de espera y ocurre un error se hace un rollback y el mensaje regresa a la línea de espera.
4. **Seguridad:** La tecnología de MSMQ utiliza la seguridad de Windows para proporcionar el control de acceso, la auditoría, encriptación y autenticación de mensajes, de hecho el ensamblado **System.Messaging** es un wrapper para una API nativa.

Hay tres tipos de líneas de espera en MSMQ:

1. **Públicas:** están registradas en el Active Directory y permiten a todas las maquinas enviar y recibir mensajes a través de la red.
2. **Privadas:** solo se crean en la máquina local y pueden ser únicamente accedidas desde otra máquina si se conocen los nombres exactos de la máquina y de la línea de espera.
3. **Sistema:** son principalmente utilizadas para propósitos administrativos o para aplicaciones que necesitan una línea de espera que guarde copias de los mensajes que son procesados, por ejemplo las aplicaciones de auditoría.

Como ejemplo de la utilización de **MS Message Queue** con C#, hice el siguiente programa que consta de dos clases: **MessageQueueHelper** y **TestMessageQueue**, la primera clase tiene los métodos para crear la cola (CreateQueue), enviar mensaje (SendMessage) y recibir mensaje (GetMessage), la segunda clase es un programa que muestra el uso de los métodos (CreateQueue) y (SendMessage).

Aquí el código de la clase *MessageQueueHelper*

```
using System;
using System.Messaging;

namespace Samples{
    public class MessageQueueHelper{
        static MessageQueue queue = null;
        public static Guid CreateQueue(string queuePath,
            bool isTransactional,
            string label)
        {
            if(MessageQueue.Exists(queuePath))
                queue = new MessageQueue(queuePath);
            else
            {
                queue = MessageQueue.Create(queuePath,isTransactional);
                queue.Label = label;
            }
            return queue.Id;
        }

        public static void SendMessage(string queuePath,
            string message,string subject)
        {
            message += "\nCreated on " + DateTime.Now.ToString("dd/MM/yyyy hh:mm:ss");
            MessageQueueTransaction trans = null;
            if(MessageQueue.Exists(queuePath))
            {
                queue = new MessageQueue(queuePath);
                queue.Formatter = new XmlMessageFormatter();
                if(queue.Transactional)
                {
                    trans = new MessageQueueTransaction();
                    trans.Begin();
                }
                queue.Send(message,subject);

                if(queue.Transactional)
                    trans.Commit();
                Console.WriteLine("Message sent.");
            }else
                Console.WriteLine("The queue doesn't exist");
        }

        public static string GetMessage(string queuePath)
        {
            string message = null;
            Message queueMessage = null;
            if(MessageQueue.Exists(queuePath))
            {
                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();
}
else
    message = "That queue doesn't exist";
return message;
}

}
}

```

Bien ahora el código de la clase **TestMessageQueue** que prueba los métodos de la clase anterior.

```
using System;
```

```

namespace Samples{
public class TestMessageQueue{
public static void Main(string[] args){
bool isTransactional = false;
Console.WriteLine("\nTest MSMQ\n");
try
{
    Console.Write("\nEnter path > ");
    string path = Console.ReadLine();
    Console.Write("\nIs transactional? (1=true/0=false) > ");
    string sTransactional = Console.ReadLine();
    if(sTransactional.Equals("1"))
        isTransactional = true;
    Console.Write("\nEnter a label > ");
    string label = Console.ReadLine();
    Guid guid = MessageQueueHelper.CreateQueue(path,isTransactional,label);
    Console.WriteLine("\nMessageQueue created with guid {0}\n",guid);
    Console.Write("\nMessage subject > ");
    string subject = Console.ReadLine();
    Console.Write("\nMessage to send > ");
    string message = Console.ReadLine();
    Console.WriteLine();
    MessageQueueHelper.SendMessage(path,message,subject);
}
catch(Exception ex){
    Console.WriteLine(ex.Message);
}
}

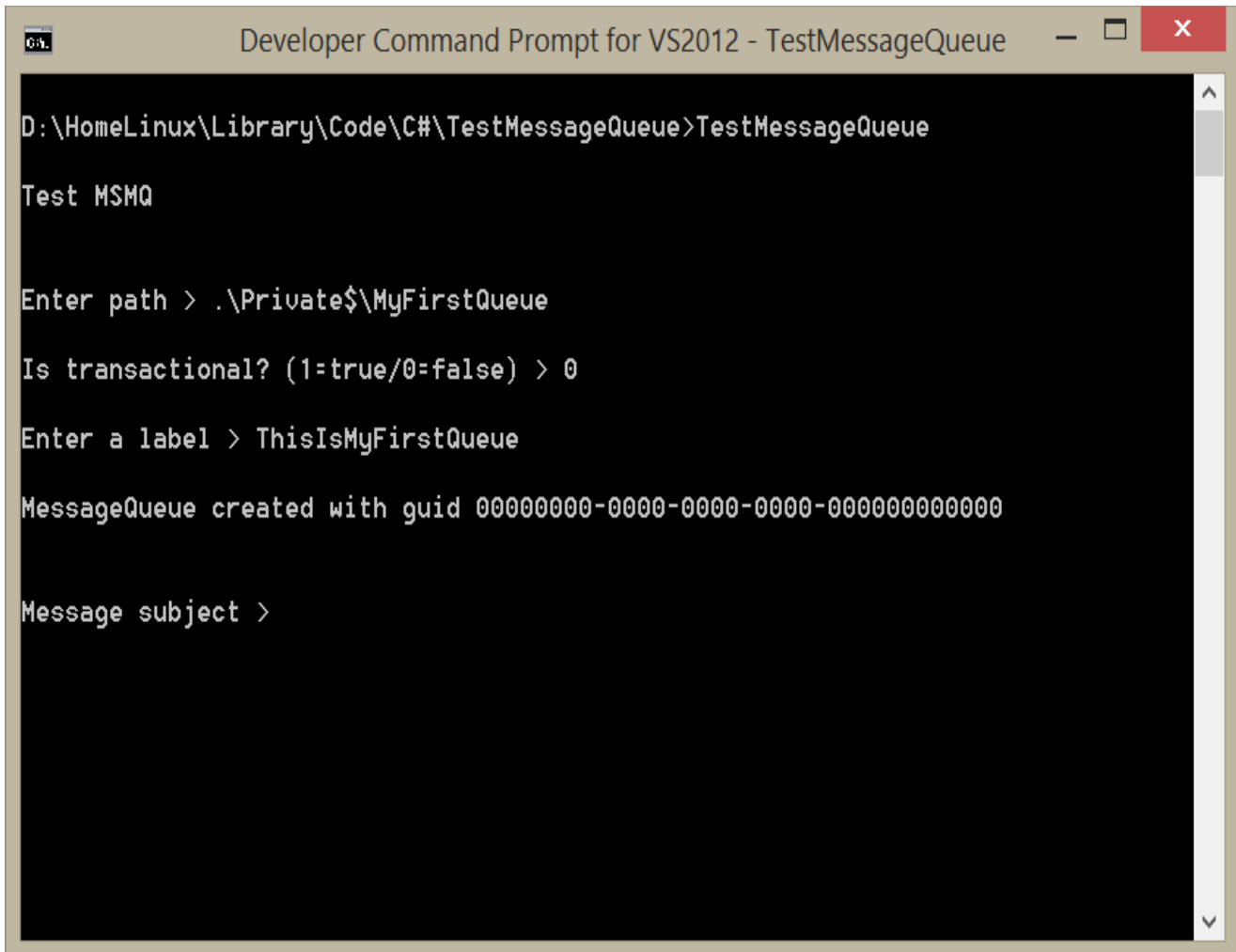
}
}

```

Al ejecutar el programa solicitará los siguientes parámetros:

- *Path*: Es el nombre de la cola de espera.
- *Is transactional*: 1 si la cola es transaccional, 0 si no lo es.
- *Label*: Identificador de la cola.

Fig 1 Ejecución del programa solicitando parámetros para crear la Message Queue.

A screenshot of a Windows command prompt window titled "Developer Command Prompt for VS2012 - TestMessageQueue". The window has a black background with white text. The command prompt shows the following sequence of interactions: the user runs "TestMessageQueue" from the directory "D:\HomeLinux\Library\Code\C#\TestMessageQueue"; the program prompts "Test MSMQ"; the user enters ".\Private\$\MyFirstQueue" for the path; the program asks "Is transactional? (1=true/0=false)" and the user enters "0"; the program asks "Enter a label" and the user enters "ThisIsMyFirstQueue"; the program outputs "MessageQueue created with guid 00000000-0000-0000-0000-000000000000"; and finally, the program prompts "Message subject >".

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

Test MSMQ

Enter path > .\Private$\MyFirstQueue

Is transactional? (1=true/0=false) > 0

Enter a label > ThisIsMyFirstQueue

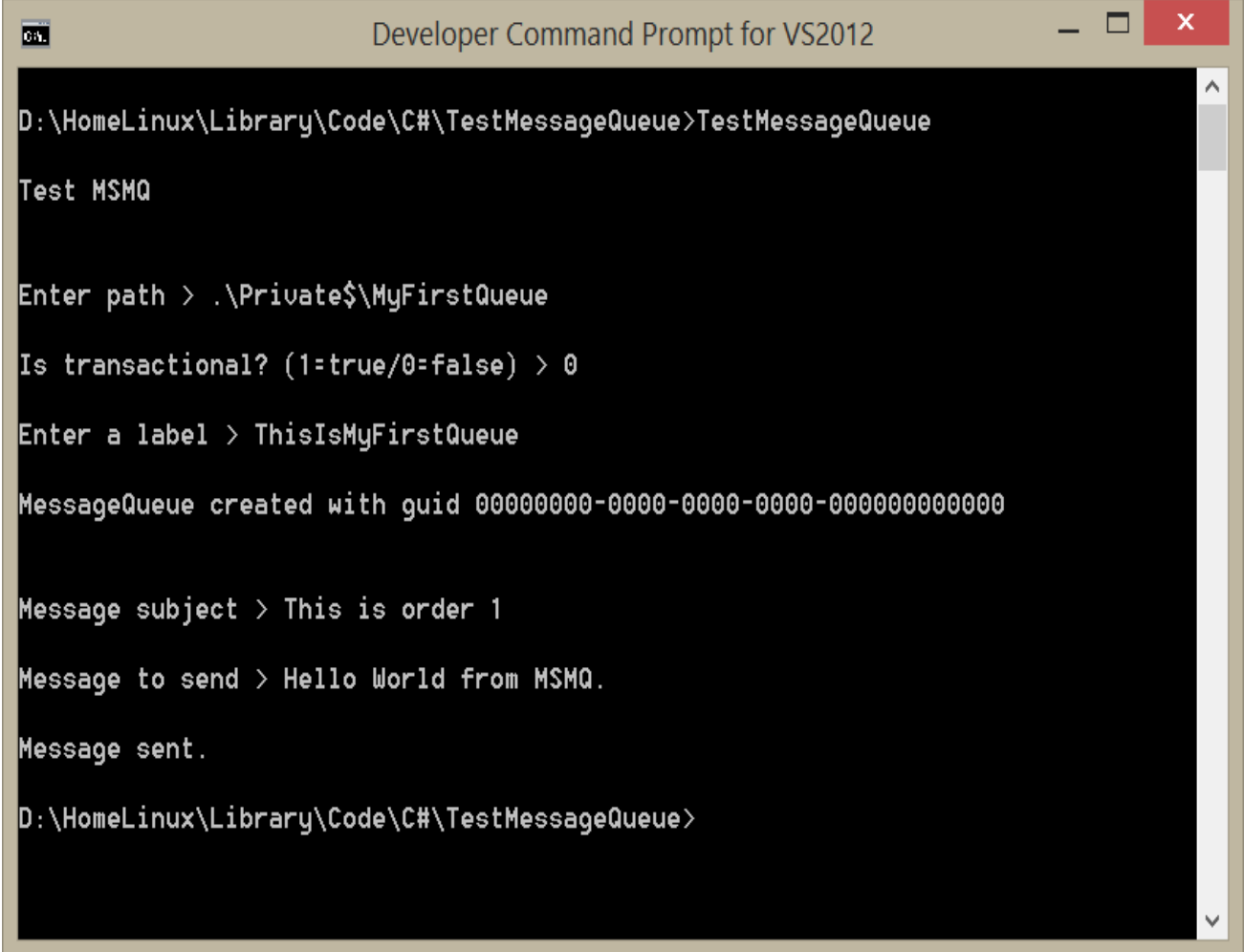
MessageQueue created with guid 00000000-0000-0000-0000-000000000000

Message subject >
```

Al crearse la cola de mensajes el programa regresará el identificador: **00000000-0000-0000-0000-000000000000** por tratarse de una cola de mensajes privadas (*private MSMQ*).

Ahora el programa solicitará el título y el cuerpo del mensaje, después lo enviará hacia la cola de mensajes creada.

Fig 2 Ejecución del programa solicitando el mensaje a enviar.



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

Test MSMQ

Enter path > .\Private$\MyFirstQueue

Is transactional? (1=true/0=false) > 0

Enter a label > ThisIsMyFirstQueue

MessageQueue created with guid 00000000-0000-0000-0000-000000000000

Message subject > This is order 1

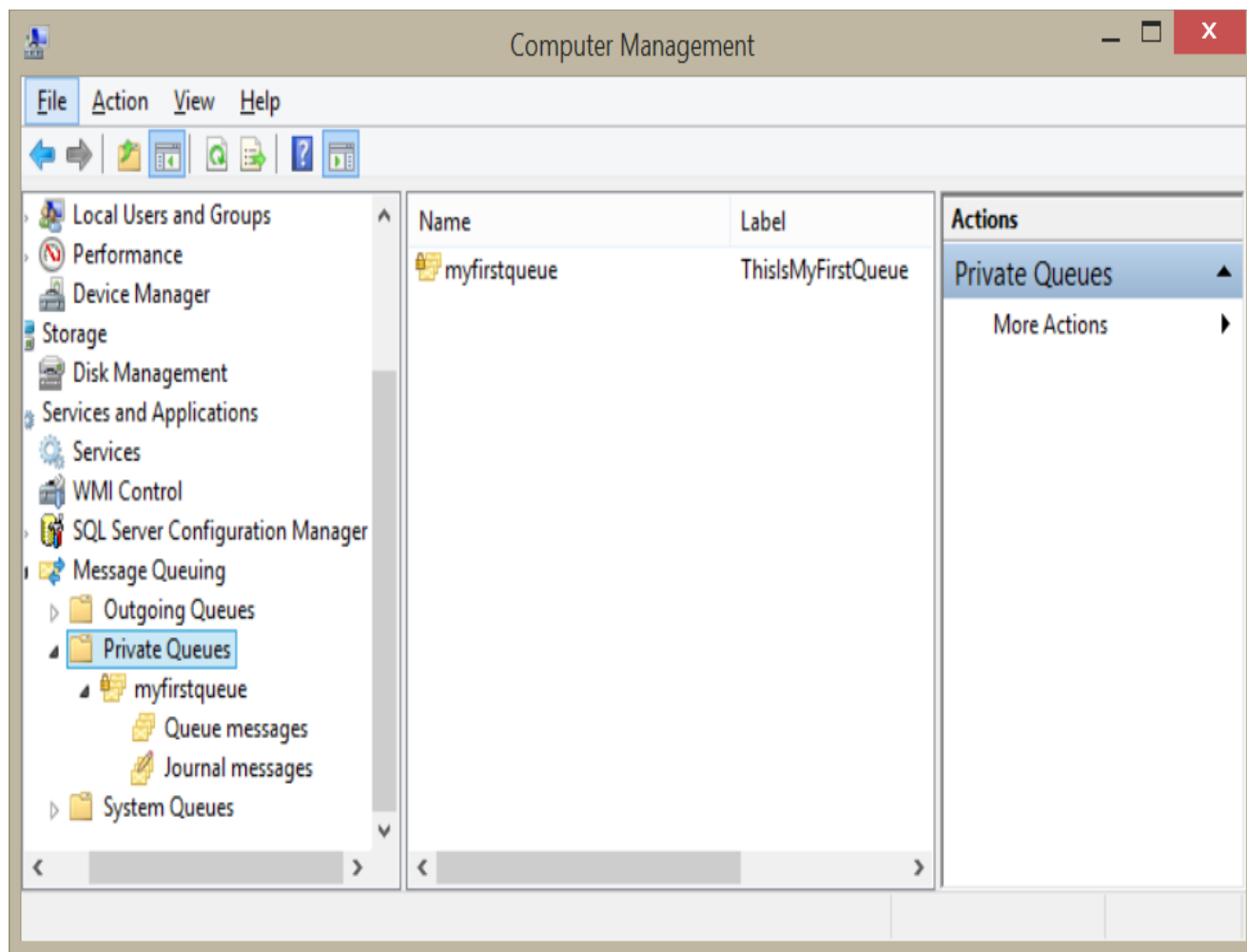
Message to send > Hello World from MSMQ.

Message sent.

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

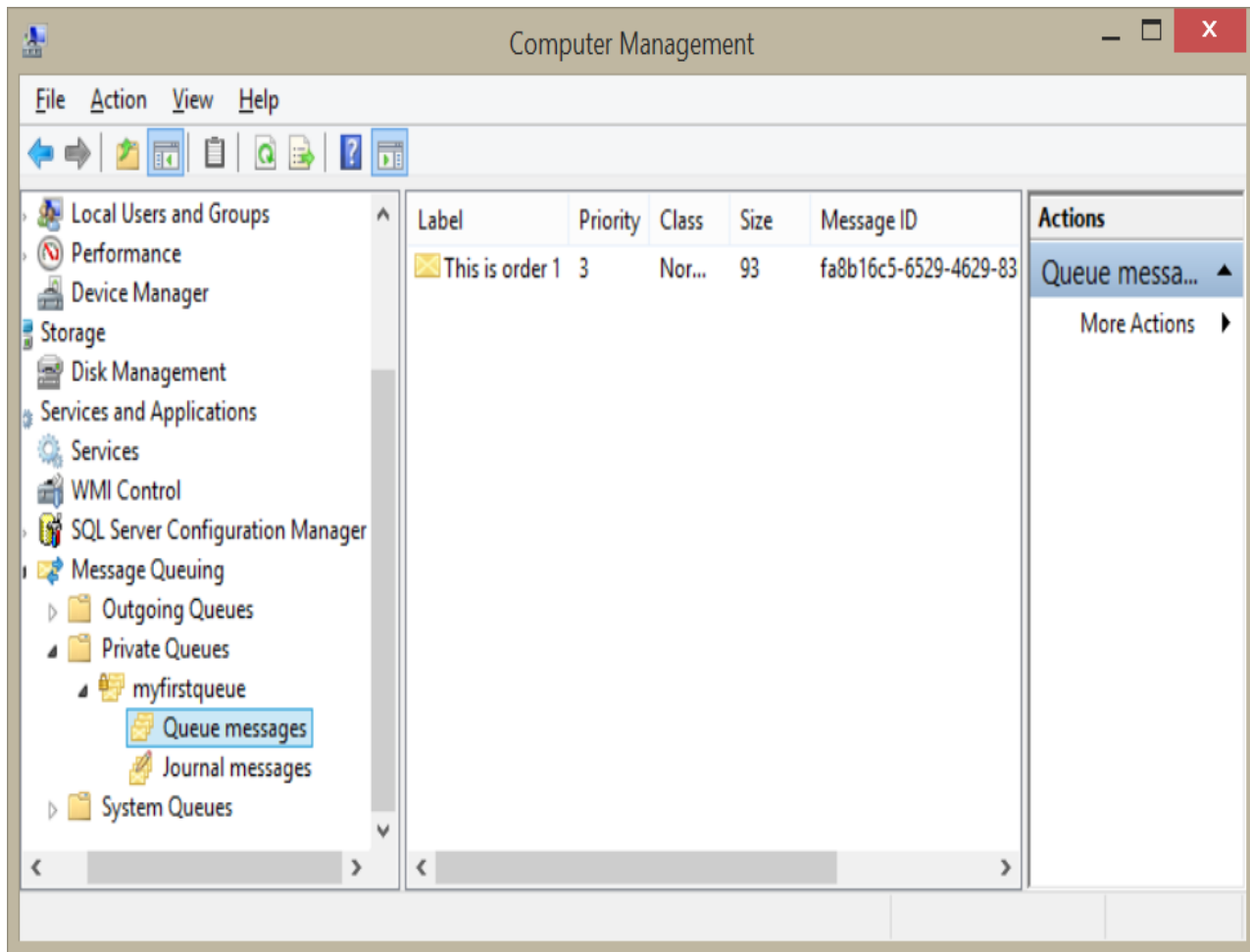
Para ver la cola de mensajes creada con el programa y el mensaje enviado, ejecutamos *Computer Management* de Windows que se encuentra en la ruta *Control Panel\System and Security\Administrative Tools* y debajo del nodo de *Services and Applications* veremos el nodo *Private Queues* debajo del nodo **Message Queuing** donde se encuentra la línea de espera creada con el programa de ejemplo.

Fig 3 Consultando la cola de mensajes creada con el computer management.



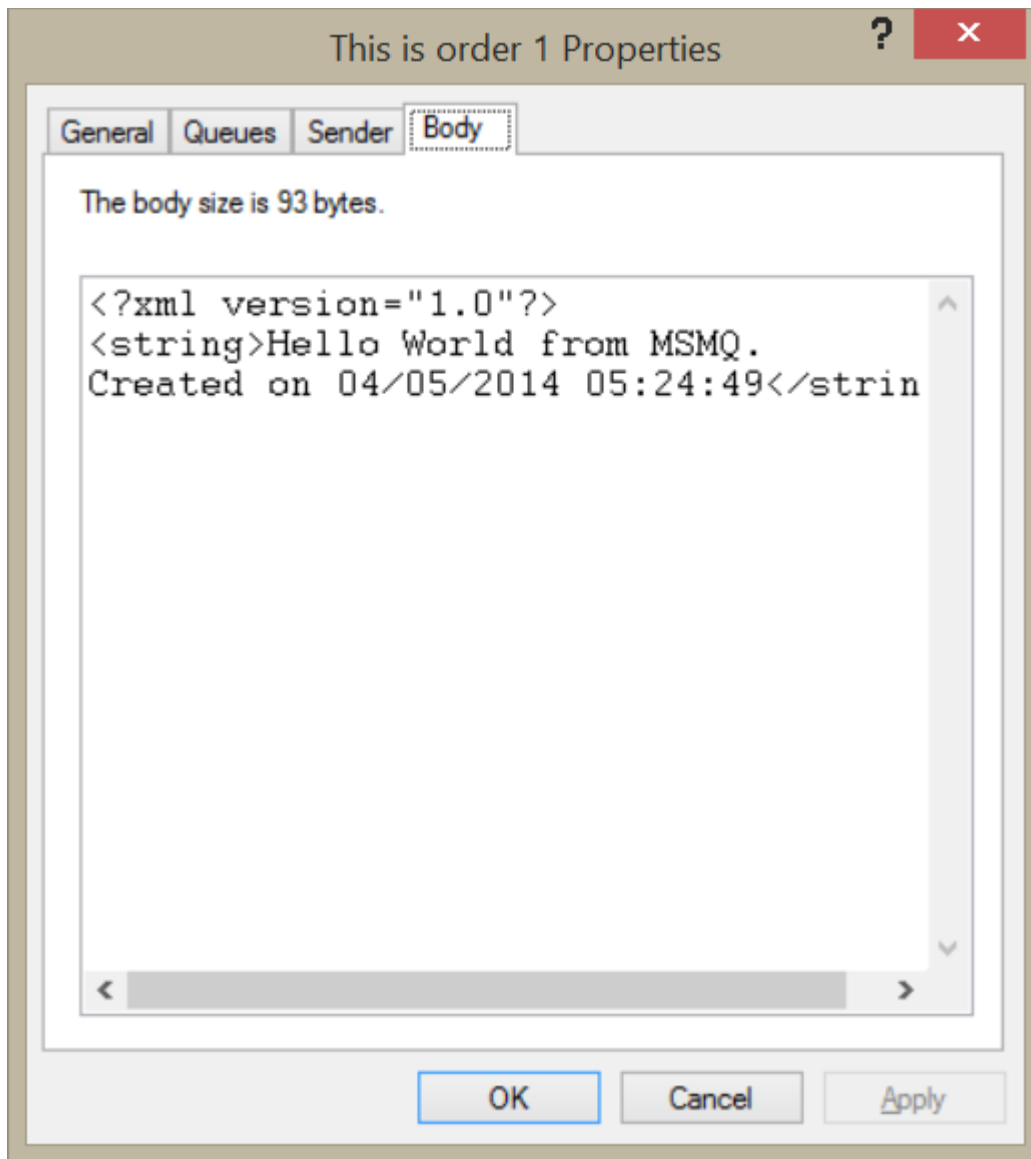
Debajo de la cola de espera **myfirstqueue**, se encuentra la carpeta **Queue messages** en donde se puede ver la etiqueta del mensaje.

Fig 4 Observando las propiedades de la message queue.



Si hacemos doble click en el mensaje aparecerá una ventana con cuatro pestañas en donde podemos ver las propiedades del mensaje, en la pestaña *body* podemos ver el contenido del mensaje.

Fig 5 Consultando el contenido del mensaje.



[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»