Практическое задание № 4. Очереди сообщений (Message Queue)

Теория

Материал составлен на основе ресурса: Горин С.В., Крищенко В.А. Поддержка разработки распределенных приложений в Microsoft .NET Framework. Учебный курс. [Электронный ресурс]. URL: http://window.edu.ru/resource/701/41701/files/ds-dotnet.pdf (дата обращения: 15.01.2014).

При функционировании распределенных приложений часто бывает, что один или несколько компонент распределенного приложения недоступны в сети по той или иной причине (разрыв канала связи, перезагрузка компьютера или выход его из строя и др.). Рассмотренные ранее механизмы коммуникации позволяли выполнять передачу сообщений только в том случае, когда оба компонента присутствуют в сети. Технология очередей сообщений призвана устранить это ограничение.

В настоящий момент существует несколько основных разработок в области промежуточного программного обеспечения для работы с очередями сообщений. Наиболее известными разработками являются такие системы очередей сообщений, как MSMQ, Sun Java System Message Queue, IBM Websphere MQ, Oracle Advanced Queing. Промежуточная среда MSMQ – разработка Microsoft для асинхронной передачи сообщений внутри *локальной сети*, впервые появившаяся в составе операционной системы Windows NT.

Служба MSMQ позволяет произвольному набору приложений добавлять сообщения в некоторую очередь, и произвольному же набору – читать сообщения из очереди. Приложению, использующему MSMQ, доступны следующие основные операции:

- добавить сообщение в очередь;
- извлечь первое сообщение из очереди;
- установить обработчик, вызываемый при появлении сообщения.

В качестве передаваемого сообщения может выступать не только текстовая строка, но и XML-файл, записи ADO, документы MS Wod и др.

При отправке сообщения с использованием MSMQ посылающему приложению необходимо указать имя компьютера и очереди, в которую его необходимо доставить. После вызова приложением функции отправки сообщение сохраняется в локальной исходящей очереди. Затем MSMQ определяет имя компьютера и очереди, куда необходимо передать сообщение. Возможны следующие случаи:

- сообщение доставляется сразу в указанную отправителем очередь (прямая доставка);
- сообщение посылается в некоторую промежуточную очередь, определяемую маршрутизатором службы сообщений;
- MSMQ определяет, что сообщение требуется разослать в несколько очередей (возможность поддерживается, начиная с MSMQ 3.0).

После определения имени компьютера с очередью назначения, MSMQ проверяет доступность компьютера (пакетом UDP) и в случае ответа сразу пытается отправить ему сообщение, повторяя попытки с интервалом по умолчанию 5 секунд. Если сообщение не удается отправить, то обычно каждые 5 минут служба сообщений пытается найти новый пункт назначения сообщения, используя маршрутизацию MSMQ. Процесс пересылки сообщения между компьютерами повторяется, пока оно не достигнет очереди назначения. С момента поступления сообщения в указанную при отправке очередь любое использующее MSMQ приложение с необходимыми правами доступа может прочитать это сообщение.

Таким образом, MSMQ поддерживает асинхронную передачу сообщений, при которой участвующие в обмене компоненты распределенной системы не обязательно должны функционировать в один и тот же момент времени.

Приложение может вести поиск нужной ему очереди по ряду критериев. Это возможно при использовании механизма общих очередей в Microsoft Message Queuing, что требует развертывания Microsoft Active Directory.

Служба MSMQ может работать как в составе домена Active Directory, так и при отсутствии такого домена, но во втором случае невозможно использовать ряд возможностей MSMQ, а именно:

- не поддерживается шифрование передаваемых сообщений;
- не поддерживаются общие очереди и механизмы их обнаружения;
- не поддерживается маршрутизация сообщений и групповая рассылка сообщений.

В MSMQ существуют два вида очередей – общие (public) и частные (private). Информация об *общих* очередях публикуется в службе каталогов Active Directory. Путь к общей очереди имеет вид

```
ComputerName\QueueName,
```

возможна также запись пути в виде

```
.\QueueName
```

для очереди на локальном компьютере. Посылающее сообщение приложение может искать доступные в домене общие очереди по заданным критериям. Возможна также проверка приложением наличия общей очереди на удаленном компьютере и ее создание. Для использования общих очередей требуется развертывание MSMQ в рамках домена Active Directory.

Частные очереди доступны как при наличии домена, так и при его отсутствии. Путь к такой очереди имеет вид

```
ComputerName\Private$\QueueName, или
.\Private$\QueueName
```

для локального компьютера.

Промежуточная среда Microsoft Message Queuing обеспечивает асинхронный обмен сообщениями и может быть использована программными компонентами распределенной системы в одном из следующих случаях:

- необходимо организовать параллельную обработку заявок несколькими компьютерами;
- одна компонента посылает другой запросы без получения ответов на них;
- взаимодействие компонент не должно быть синхронным;
- требуется интеграция с какими-либо другими независимыми системами, которые могут использовать очереди сообщений (MSMQ или IBM MQ).

Функции для работы с очередями сообщений MSMQ

1. Для создания очереди сообщений используется метод Create класса MessageQueue:

```
MessageQueue queue = MessageQueue.Create(string path), где
     queue – созданная очередь сообщений;
     path – путь (имя) к очереди сообщений в формате, указанном выше.
```

2. Открытие существующей очереди выполняется с помощью конструктора класса MessageQueue:

```
MessageQueue queue = new MessageQueue(string path), где
     queue – открытая очередь сообщений;
     path – путь к очереди сообщений.
```

3. Для проверки существования очереди используется метод Exists класса MessageQueue:

```
bool MessageQueue.Exists(string path), где path – путь к очереди сообщений.
```

4. Удаление очереди сообщений выполняется с помощью метода Delete класса MessageQueue:

```
MessageQueue.Delete(string path), где path – путь к очереди сообщений.
```

5. Удаление всех сообщений из очереди выполняется с помощью метода Purge класса MessageQueue:

```
queue.Purge(), где queue – очередь, сообщения из которой следует удалить.
```

6. Для отправки сообщения в очередь используется перегруженный метод Send класса MessageQueue:

```
queue.Send(object obj, string label), где
     queue – очередь, в которую следует отправить сообщение;
     obj – отправляемое в очередь сообщение (строка, записи БД, файл MS Word и др.);
     label – заголовок сообщения.
```

7. Для чтения сообщения из очереди используется перегруженный метод Receive класса MessageQueue:

```
Message queue.Receive(), где queue – очередь, из которой выполняется чтение сообщения.
```

8. Для просмотра сообщения (чтения сообщения без его удаления из очереди) используется метод Peek класса MessageQueue:

```
Message queue.Peek(), где queue – очередь, в которой выполняется просмотр сообщения.
```

Алгоритм коммуникации с помощью очередей сообщений

Коммуникация с использованием очередей сообщений выполняется по следующему алгоритму:

1. Сервер создает новую очередь сообщений или открывает существующую.
2. Клиент открывает существующую очередь сообщений с помощью конструктора класса MessageQueue.
3. Клиент записывает сообщения в очередь сообщений с помощью метода Send класса MessageQueue.
4. Сервер периодически выполняет проверку наличия сообщений в очереди, например, с помощью метода Peek класса MessageQueue. Если в очереди есть сообщение, то сервер переходит к шагу 5.
5. Сервер производит чтение сообщения из очереди с помощью метода Receive класса MessageQueue.
6. Пока у клиента есть сообщения, выполняется последовательность шагов 3-5. Иначе выполняется переход к шагу 7.
7. В случае необходимости сервер удаляет очередь сообщений.

Разработка приложения «Чат v 1.0»

Разработаем приложение, которое позволяет с помощью промежуточной среды MSMQ отправлять серверу сообщения сразу от нескольких клиентов. Сервер представляет собой форму, на которой отображаются все полученные им сообщения (см. рис. 1).

Рис. 1. Окно сервера при коммуникации с помощью MSMQ

Клиент представляет собой форму, содержащую два поля ввода и две кнопки (см. рис. 2).

Первое поле предназначено для ввода пути к очереди сообщений. При нажатии на кнопку «Подключиться» клиент открывает существующую на сервере очередь сообщений. Второе поле предназначено для ввода сообщения для отправки. При нажатии на кнопку «Отправить» сообщение, в котором указано имя компьютера клиента (для его

идентификации) и текст сообщения, введенный пользователем, будет отправлено в очередь сервера.

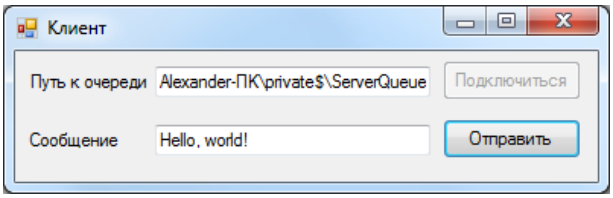

Рис. 2. Окно клиента при коммуникации с помощью MSMQ

Перед тем как начать разработку приложения и обращаться к пространству имен System.Messaging необходимо сначала добавить в проект ссылку на это пространство имен (см. рис. 3-4)

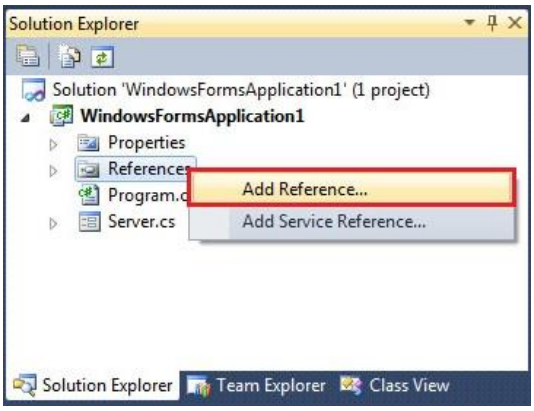

***Рис. 3. Добавление ссылки на пространство имен System.Messaging. Этап* I**

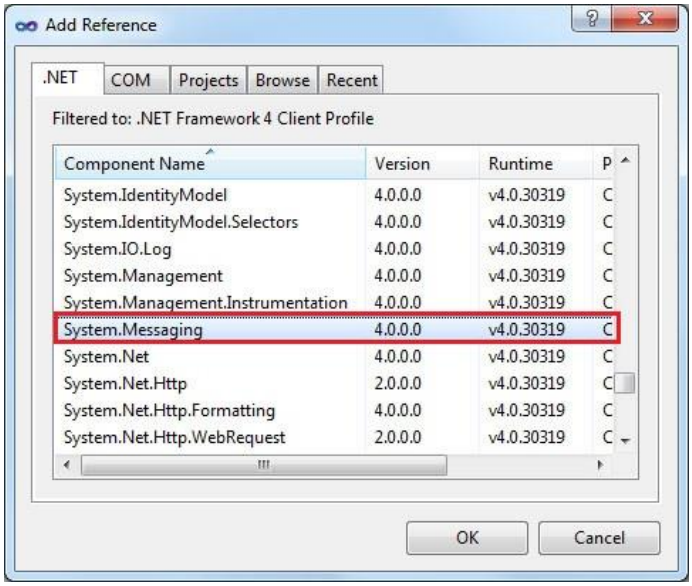

***Рис. 4. Добавление ссылки на пространство имен System.Messaging. Этап* II**

Код реализации сервера с комментариями приведен ниже:

```
    public partial class frmMain : Form
    {
        private MessageQueue q = null;          // очередь сообщений
        private Thread t = null;                // поток, отвечающий за работу с очередью
сообщений
        private bool _continue = true;          // флаг, указывающий продолжается ли работа с
мэйлслотом

        // конструктор формы
```

```
        public frmMain()
        {
            InitializeComponent();
            string path = Dns.GetHostName() + "\\private$\\ServerQueue";    // путь к очереди
сообщений, Dns.GetHostName() - метод, возвращающий имя текущей машины

            // если очередь сообщений с указанным путем существует, то открываем ее, иначе
создаем новую
            if (MessageQueue.Exists(path))
                q = new MessageQueue(path);
            else
                q = MessageQueue.Create(path);

            // задаем форматтер сообщений в очереди
            q.Formatter = new XmlMessageFormatter(new Type[] { typeof(String) });

            // вывод пути к очереди сообщений в заголовок формы, чтобы можно было его
использовать для ввода имени в форме клиента, запущенного на другом вычислительном узле
            this.Text += "     " + q.Path;

            // создание потока, отвечающего за работу с очередью сообщений
            Thread t = new Thread(ReceiveMessage);
            t.Start();
        }

        // получение сообщения
        private void ReceiveMessage()
        {
            if (q == null)
                return;

            System.Messaging.Message msg = null;

            // входим в бесконечный цикл работы с очередью сообщений
            while (_continue)
            {
                if (q.Peek() != null)   // если в очереди есть сообщение, выполняем его
чтение, интервал до следующей попытки чтения равен 10 секундам
                    msg = q.Receive(TimeSpan.FromSeconds(10.0));

                rtbMessages.Invoke((MethodInvoker)delegate
                {
                    if (msg != null)
                        rtbMessages.Text += "\n >> " + msg.Label + " : " + msg.Body;     //
выводим полученное сообщение на форму
                });
                Thread.Sleep(500);          // приостанавливаем работу потока перед тем, как
приступить к обслуживанию очередного клиента
            }
        }

        private void frmMain_FormClosing(object sender, FormClosingEventArgs e)
        {
            _continue = false;      // сообщаем, что работа с очередью сообщений завершена

            if (t != null)
            {
                t.Abort();          // завершаем поток
            }

            if (q != null)
            {
                //MessageQueue.Delete(q.Path);      // в случае необходимости удаляем очередь
сообщений
            }
        }
    }
```

Код реализации клиента с комментариями приведен ниже:

```
public partial class frmMain : Form
{
    private MessageQueue q = null;      // очередь сообщений, в которую будет производиться запись сообщений

    // конструктор формы
    public frmMain()
    {
        InitializeComponent();
    }

    private void btnConnect_Click(object sender, EventArgs e)
    {
        if (MessageQueue.Exists(tbPath.Text))
        {
          // если очередь, путь к которой указан в поле tbPath существует, то открываем ее
            q = new MessageQueue(tbPath.Text);
            btnSend.Enabled = true;
            btnConnect.Enabled = false;
        }
        else
            MessageBox.Show("Указан неверный путь к очереди, либо очередь не существует");
    }

    private void btnSend_Click(object sender, EventArgs e)
    {
        // выполняем отправку сообщения в очередь
        q.Send(tbMessage.Text, Dns.GetHostName());
    }
}
```

Задание

1. Убедитесь, что у вас установлена служба очередей сообщений. Для этого в «Панели управления» выберите раздел «Программы и компоненты» выберите пункт «Включение и отключение компонентов Windows». В открывшемся дереве найдите вершину «Сервер очереди сообщений Майкрософт». Если данный компонент не установлен, то доустановите его.
2. Для просмотра имеющихся на компьютере очередей сообщений в «Панели управления» перейдите в пункт «Администрирование», выберите ярлык «Управление компьютером». В открывшемся окне (в дереве навигации) щелкните по вершине «Службы и приложения» и выберите пункт «Очередь сообщений». Изучите какие очереди сообщений уже имеются на вашей машине.
3. Изучите разработанное приложение. Запустите исполняемый файл сервера и клиента на одной машине, затем перенесите один из компонентов распределенного приложения на другой вычислительный узел (лучше на ноутбук, т.к. администратор ЛВС вуза может запретить обмен сообщения между различными узлами ЛВС) и попробуйте его запустить. Объясните, почему приложение перестало работать? Запустите несколько клиентов на одной машине с сервером и попробуйте отправить сообщения серверу. В процессе изучения (тестирования) приложения просмотрите очереди сообщений компьютера и убедитесь, что сообщения действительно приходят в очередь.
4. Запустите несколько серверов на одной машине. Отправьте им сообщения от нескольких клиентов. Объясните, почему приложение перестало корректно работать.
5. Модифицируйте приложение так, чтобы существовала возможность на сервере идентифицировать клиентов не по имени вычислительного узла, а по нику/логину пользователя.
6. Модифицируйте приложение так, чтобы получился полноценный чат. Клиент может отправлять сообщения всем клиентам, участвующим в беседе. Для этого каждый

клиент должен иметь возможность просмотра всех сообщений от всех клиентов, а сервер должен содержать список клиентов, которые хотят участвовать в беседе, чтобы каждый раз выполнять им рассылку сообщений.