Practical WebSphere MQ

Elton Stoneman
geekswithblogs.net/eltonstoneman
@EltonStoneman

pluralsight
hardcore dev and IT training

Practical WebSphere MQ

Move queue configuration to **app config**

Implement IMessageQueue with **WebSphere MQ**

Run message handlers as **Windows Service**

Configuration

Message queue implementation

Hard-coded in MessageQueueFactory

One implementation for all

IMessageQueue configuration

Connection settings

Centralize in MessageQueueBase

Queue addresses

Individual format

Known structure

Demo 1: Message Queue Configuration

Feature

Walkthrough typed config used for messaging settings

Task

Capture queue implementations and settings in config XML

Task

Use typed config classes to access settings

Demo 1: Message Queue Configuration

Demo 1: Message Queue Configuration

- **Messaging configuration section**
  - Contains all queue implementations

```
<messageQueue name="MSMQ"
 type="Sixeyed.MessageQueue.Messaging.Msmq.MsmqMessageQueue,Sixeyed...">
```

  - With default queue type

```
<sixeyed.messageQueue.messaging defaultMessageQueueName="MSMQ">
```

  - And overrides by message type

```
<message name="doesuserexist" messageQueueName="ZeroMQ"/>
```

Demo 1: Message Queue Configuration

- **Message queue configuration element**
  - Contains queue addresses

```
<queue name="doesuserexist" address="tcp://127.0.0.1:5556"/>
<queue name="unsubscribe" address="tcp://127.0.0.1:5555"/>
<queue name="unsubscribed-event" address="pgm://127.0.0.1;239.192.1..."/>
```

  - In required format

```
<queue name="unsubscribe-crm" address="unsubscribed-event:crm"/>
```

  - And custom properties

```
<property name="queuemanager" value="SC.UNSUB"/>
<property name="hostname" value="127.0.0.1"/>
<property name="channel" value="UNSUB.SVRCONN"/>
```

Demo 1: Message Queue Configuration

- **Message queue factory**
  - Get specific queue for message
  - Or default message queue

```
var config = MessagingConfiguration.Current.Messages.
                                        SingleOrDefault(x => x.Name == name);
var queueName = config != null
                ? config.MessageQueueName
                : MessagingConfiguration.Current.DefaultMessageQueueName;
```

  - Lookup type to create

```
var queueType = MessagingConfiguration.Current.MessageQueues
                                      .Single(x => x.Name == queueName).Type;
```

Demo 1: Message Queue Configuration

- **Message queue base class**
  - Stores custom properties

```
var config = MessagingConfiguration.Current.MessageQueues.Single
                                                       (x => x.Name == Name);
foreach (var property in config.Properties)
{
    Properties.Add(property.Name, property.Value);
}
```

  - Exposes Get and Require methods

```
Initialise(Direction.Outbound, name, pattern, isTemporary);
var queueManagerName = RequirePropertyValue("queuemanager");
```

Demo 1: Message Queue Configuration

- **Message queue base class**
  - Implements GetAddress

```
public virtual string GetAddress(string name)
{
    var config = MessagingConfiguration.Current.MessageQueues.Single
                                                    (x => x.Name == Name);
    var queue = config.Queues.SingleOrDefault(x => x.Name == name);
    return queue == null ? name : queue.Address;
}
```

WebSphere MQ Implementation

IMessageQueue

Create queues and topics

Extend MessageQueueBase

MQQueueManager, MQQueue etc.

Capture config settings

Demo 2: WebSphere MQ

Feature

Implement IMessageQueue with WebSphere

Task

Support fire-and-forget

Task

Support request-response

Demo 2: WebSphere MQ

Demo 2: WebSphere MQ

- **WebSphereMqMessageQueue**
  - Queue Manager

```
var queueManagerName = RequirePropertyValue("queuemanager");
var properties = new Hashtable();
foreach (var property in Properties.Where(x => x.Key != "queuemanager"))
{
    properties.Add(property.Key, property.Value);
}
_queueManager = new MQQueueManager(queueManagerName, properties);
```

  - Send queue

```
_queue = _queueManager.AccessQueue(Address, MQC.MQOO_OUTPUT);
```

Demo 2: WebSphere MQ

- **WebSphereMqMessageQueue**
  - Build outgoing message

```
var messageJson = message.ToJsonString();
var outgoing = new MQMessage();
outgoing.Format = MQC.MQFMT_STRING;
outgoing.WriteString(messageJson);
```

  - Send & commit

```
_queue.Put(outgoing);
_queueManager.Commit();
```

Demo 2: WebSphere MQ

- **WebSphereMqMessageQueue**
  - Receive queue

```
_queue = _queueManager.AccessQueue(Address, MQC.MQOO_INPUT_AS_Q_DEF);
```

  - Receive message

```
var inbound = new MQMessage
{
    Format = MQC.MQFMT_STRING
};
if (maximumWaitMilliseconds > 0) { //... }
else
{
    _queue.Get(inbound);
}
```

Demo 2: WebSphere MQ

- **WebSphereMqMessageQueue**
  - Create response queue

```
_queueManager.AccessQueue("dynamic.response.model",
  MQC.MQOO_INPUT_EXCLUSIVE, queueManagerName, "dynamic.response.*", "");
```

  - Initialise reply queue

```
_queue = _queueManager.AccessQueue(Address, MQC.MQOO_OUTPUT,
                                            queueManagerName, null, null);
```

  - Initialise response queue

```
_queue = _queueManager.AccessQueue("dynamic.response.model",
               MQC.MQOO_INPUT_EXCLUSIVE, queueManagerName, Address, "");
```

Demo 2: WebSphere MQ

- **WebSphereMqMessageQueue**
  - Configuration

```xml
<messageQueue name="WebSphereMQ" type="Sixeyed.MessageQueue...">
  <properties>
    <property name="queuemanager" value="SC.UNSUB"/>
    <property name="hostname" value="127.0.0.1"/>
    <property name="channel" value="UNSUB.SVRCONN"/>
  </properties>
  <queues>
    <queue name="doesuserexist" address="doesuserexist"/>
    <queue name="unsubscribe" address="unsubscribe"/>
  </queues>
</messageQueue>
```

WebSphere MQ Topics

Publish-Subscribe

Publish to Topic

Topic relays to Subscription(s)

Subscribers listen on Queues

Separate .NET topic client class

Demo 3: WebSphere MQ Pub-Sub

Feature

Complete IMessageQueue implementation with pub-sub

Task

Set up WebSphere MQ topic, queues and subscriptions

Task

Implement pub-sub client code & config

Demo 3: WebSphere MQ Pub-Sub

Demo 3: WebSphere MQ Pub-Sub

- **WebSphereMQ Topics**
  - etc – plus subscriptions as different from queue subscribers?
  - plus topic string
  - unsubscribe/user address?

Demo 3: WebSphere MQ Pub-Sub

- **WebSphereMqMessageQueue**
  - Initialise MQ*Topic* client

```
if (Pattern == MessagePattern.PublishSubscribe)
{
    _topic = _queueManager.AccessTopic(Address, "",
                                          MQC.MQTOPIC_OPEN_AS_PUBLICATION, MQC.MQOO_OUTPUT);
}
```

  - Send message

```
if (Pattern == MessagePattern.PublishSubscribe)
{
    _topic.Put(outgoing);
}
```

Demo 3: WebSphere MQ Pub-Sub

- **WebSphereMqMessageQueue**
  - Configuration

```
<queues>
 <!-- ... -->
 <queue name="unsubscribed-event" address="unsubscribed/user"/>
 <queue name="unsubscribe-legacy" address="unsubscribe.legacy"/>
 <queue name="unsubscribe-crm" address="unsubscribe.crm"/>
 <queue name="unsubscribe-fulfilment" address="unsubscribe.fulfilment"/>
</queues>
```

Message Handler Windows Service

C

Consolidate to single handler

TopShelf: run as console or service

Replace console output with log4net

Install one handler per queue

Demo 4: Message Handler Windows Service

Feature

Package message handler as Windows Service

Task

Walkthrough log4net and TopShelf integration

Task

Walkthrough service-aware end-to-end tests

Demo 4: Message Handler Windows Service

Demo 4: Message Handler Windows Service

- **log4net**
  - Initialise logger

```
static Log()
{
    XmlConfigurator.Configure();
    _Log = LogManager.GetLogger("Sixeyed.MessageQueue.Handler");
}
```

  - Write output

```
public static void WriteLine(string format, params object[] args)
{
    _Log.Debug(string.Format(format, args));
}
```

Demo 4: Message Handler Windows Service

- **TopShelf**
  - Separate handler code in *QueueListener*
  - Run *QueueListener* through *HostFactory*

```
HostFactory.Run(hostConfig =>
    {
        hostConfig.AddCommandLineDefinition("listenOnQueueName", q =>
                                                 { listenOnQueueName = q; });
        hostConfig.ApplyCommandLine();
        hostConfig.Service<QueueListener>(
            sc =>
                {
                    sc.ConstructUsing(() => new QueueListener());
                    sc.WhenStarted(s => s.Start(listenOnQueueName));
                    sc.WhenStopped(s => s.Stop());
                });
        //...
```

Summary

- **WebSphere IMessageQueue** ✓
  - MQQueueManager & MQQueue
- **Request-Response** ✓
  - Dynamic response queues
- **Publish-Subscribe** ✓
  - Topics & topic strings
  - Subscriber queues
- **Practical refactoring** ✓
  - Move queue settings to config
  - Move queue addresses to config
  - Run message handler as Windows Service

Summary