Introducing ZeroMQ

Elton Stoneman
geekswithblogs.net/eltonstoneman
@EltonStoneman

pluralsight
hardcore dev and IT training

Introducing ZeroMQ

Cross-platform open-source messaging

Virtual queues hosted **in-process** and **in-memory**

Socket-based API surfaces **messaging patterns**

Goals

ZeroMQ technology overview
How it works, functionality it provides
History and aims

Communication & connection
Embedding the queue in-process

.NET client library
API usage & features

Pattern support
Fire-and-forget, request-response, publish-subscribe

ZeroMQ

a.k.a. ØMQ, 0MQ, ZMQ

: “zero broker”

What is ZeroMQ?

C

Embedded messaging library

In-memory queue technology

LGPL 3
Free Software
Free as in Freedom

Cross-platform and **open-source**

Aims of ZeroMQ

Reusable messaging layer

- Easy to integrate
- Cheap to run
- For small(ish) solutions

Advanced feature set

- Asynchronous communication
- Choice of protocols
- Surfaced as patterns

Zero Frills

No **serialization** or **compression**

No **encryption** or **authentication**

No **durable** messaging

Demo 1: Using ZeroMQ

Feature

Explore ZeroMQ using .NET

Task

Embed ZeroMQ in host process

Task

Send and receive messages

Demo 1: Using ZeroMQ

Demo 1: Using ZeroMQ

- **Host process**
  - *libzmq.dll* (C++ library) & *clrzmq.dll* (.NET assembly)
- **Message queues**
  - Hosted in-process
  - Accessed with Socket API & explicit pattern

```
var context = new Context();
using (var client = context.Socket(SocketType.PUSH))
{
    client.Connect("tcp://localhost:5555");
    for (int i=0; i<1000; i++)
    {
        client.Send("Message: " + i, Encoding.UTF8);
    }
}
```

Demo 1: Using ZeroMQ

- **Receiving messages**
  - Socket API & explicit pattern
  - *Bind()* to wildcard address

```
var context = new Context();
using (var server = context.Socket(SocketType.PULL))
{
    server.Bind("tcp://*:5555");
    while (true)
    {
        var message = server.Recv(Encoding.UTF8);
        //etc.
    }
}
```

ZeroMQ and .NET

.NET “**binding**” on github & NuGet

Native C++ library version 4.0; **.NET 2.1**

Simple to use: **Context** and **Socket** classes

Context

Use as a Singleton

Creates Sockets

Inter-thread communication

Message Pattern Aware

Socket for known pattern

And direction – REQ/REP

Socket

Open connection

Connect() to send

Bind() to listen

Set properties

Backlog and HWM

Send and receive messages

byte[] or string

Single or multi-part

~~Context~~
ZmqContext

~~Socket~~
ZmqSocket

Demo 2: Socket Types

Feature

Understand ZeroMQ socket types

Task

Respond to a received message

Task

Send and receive messages

Demo 2: Socket Types

Demo 2: Socket Types

- **Fire –and-forget**
  - Sender – connects with PUSH socket type
  - Listener – binds with PULL socket type
- **Listener**
  - Calls *Recv*() – polls the queue

```
var context = new Context();
using (var server = context.Socket(SocketType.PULL))
{
    server.Bind("tcp://*:5555");
    while (true)
    {
        var message = server.Recv(Encoding.UTF8);
```

Demo 2: Socket Types

- **Request-response**
  - Sender – connects with REQ socket type

```
var context = new Context();
using (var client = context.Socket(SocketType.REQ))
{
    client.Connect("tcp://localhost:5556");
    client.Send("Request", Encoding.UTF8);
```

  - Listener – binds with REP socket type

```
using (var server = context.Socket(SocketType.REP))
{
    server.Bind("tcp://*:5556");
    while (true)
    {
        var message = server.Recv(Encoding.UTF8);
        server.Send("Response", Encoding.UTF8);
```

Demo 2: Socket Types

- **Publish-subscribe**
  - Publisher – binds with PUB socket type

```
var context = new Context();
using (var client = context.Socket(SocketType.PUB))
{
    client.Bind("tcp://*:5557");
    client.Send("Notification", Encoding.UTF8);
```

  - Subscribers – connect with SUB socket type

```
var context = new Context();
using (var server = context.Socket(SocketType.SUB))
{
    server.Connect("tcp://localhost:5557");
    server.Subscribe("", Encoding.UTF8);
    message = server.Recv(Encoding.UTF8);
```

Summary

- **Introducing ZeroMQ** ✓
  - Design and goals
- **Feature set** ✓
  - Embedded library
  - Cross-platform & open source
  - High level pattern support
- **Deployment & administration** ✗
- **.NET API "binding"** ✓
  - ZMQ Context & Socket
  - github & NuGet

ZeroMQ