PADI 2014/15

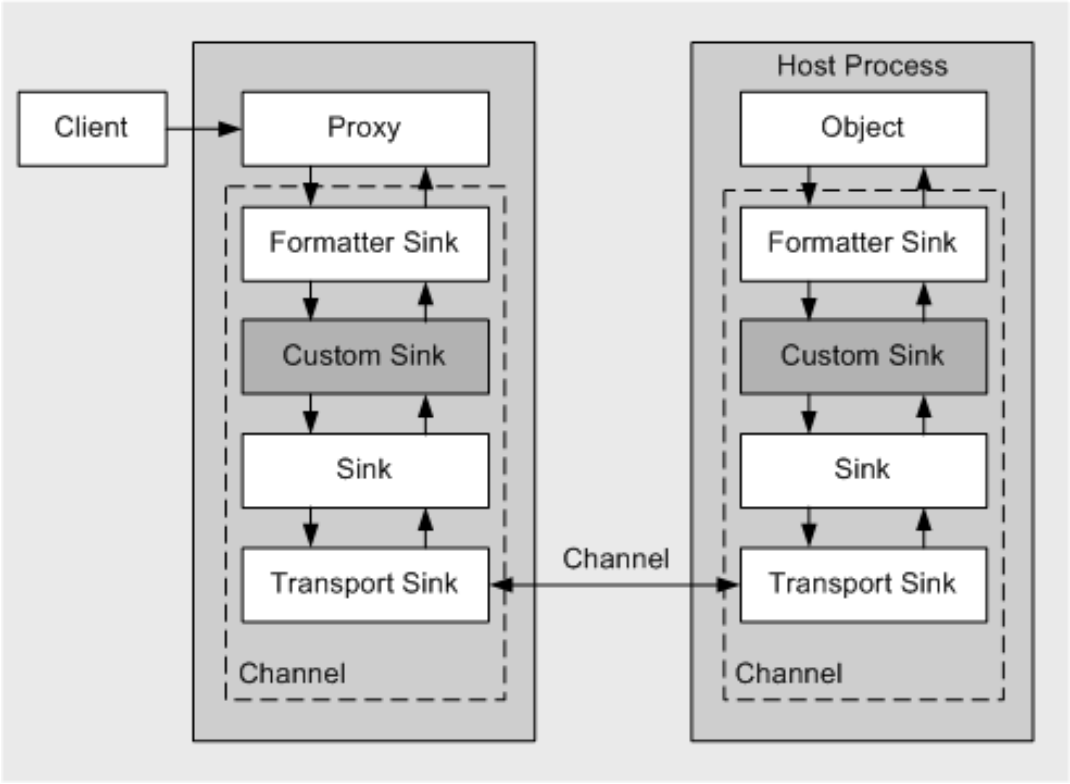
Aula 3
.Net Remoting

Sumário

1. .Net Remoting

.NET Remoting

- Comunicação entre aplicações
  - Permite invocar métodos em objectos remotos
  - Qualquer objecto pode ser usado remotamente
  - Objectos remotos devem derivar de **`MarshalByRefObject`**



Canais

- A comunicação entre dois processos distintos é realizada pelo .NET Remoting através de canais.

- Um canal é um objecto que, através de um stream de dados:
  - Empacota a informação de acordo com um tipo de protocolo
  - Envia esse pacote a outro computador.

- Esse canal pode ser unidirecional ou bidirecional

- Dois tipos:
  - TcpChannel (que envia os dados por TCP em formato binário)
  - HttpChannel (que utiliza o protocolo HTTP em formato XML)

Proxies

- Proxies são os objectos locais que são apontados pelas referências remotas.

- Quando o cliente recebe uma referência de um objecto remoto ele está efectivamente a receber uma referência a um objecto proxy local criado automaticamente pelo .NET Remoting.

- Esse proxy é encarregue de passar as chamadas dos métodos para o servidor.

Utilização de Objectos Remotos

- Requer a respectiva activação

- Tipos de activação:
  - Cliente
    - O objecto no servidor é criado quando o cliente cria a instância do objecto remoto (com new);
  - Servidor
    - O objecto no servidor é criado quando for requisitado. Ele não é criado imediatamente quando o cliente cria a instância do objecto remoto (com new), mas apenas quando o cliente fizer a primeira invocação de método ao proxy.

Activação pelo Servidor

- Na activação controlada pelo servidor, existem dois modos de activação:
- **Singleton**
  - Apenas uma instância em cada instante
  - Requer sincronização no acesso a estado partilhado
- **SingleCall**
  - Uma nova instância é criada para cada pedido.
  - Após a execução de uma chamada, a próxima chamada será servida por outra instância.

Tempo de Vida dos Objectos

- Tempo de vida dos objectos Singleton determinado por sistema de *leases*.
  - Ao expirar um *lease*, este deve ser renovado, caso contrário a memória ocupada pelo objecto é recuperada pelo garbage collector.
  - A utilização de *leases* é uma alternativa mais viável à contagem de referência
- Uso de objectos previamente instanciados
  - Tem em conta existência de referências locais

Resumo das Opções

- Canais – TcpChannel vs. HttpChannel
- Activação – Cliente vs. Servidor, Singleton vs. Single Call
- Tempo de vida – Single Call vs. Leases
- MarshallByRef vs. MarshallByValue
  - MBV: Objectos serializáveis, são copiados para o cliente
  - MBR: Cliente obtém proxy que encaminha chamadas

Exemplo: Servidor

```
class Server {
  static void Main() {
   TcpChannel channel = new TcpChannel(8086);
   ChannelServices.RegisterChannel(channel,false);
    RemotingConfiguration.RegisterWellKnownServiceType(
       typeof(MyRemoteObject),
       “MyRemoteObjectName",
       WellKnownObjectMode.Singleton);
   System.Console.WriteLine("<enter> para sair...");
   System.Console.ReadLine();
  }
}

public class MyRemoteObject : MarshalByRefObject  {
  public string MetodoOla() {
    return “ola!”;
  }
}
```

Exemplo: Cliente

```
class Client {
  static void Main() {
    TcpChannel channel = new TcpChannel();
    ChannelServices.RegisterChannel(channel,false);

    MyRemoteObject obj = (MyRemoteObject) Activator.GetObject(
       typeof(MyRemoteObject),
       "tcp://localhost:8086/MyRemoteObjectName");

    if (obj == null)
      System.Console.WriteLine("Could not locate server");
    else
      Console.WriteLine(obj.MetodoOla());
  }
}
```

Nota: permita invocação remota directamente de instancias de classes e nao exclusivamente atraves de um interface remoto como em Java. Pode aceder-se directamente a campos do objecto remoto. Em Java é sempre obrigatorio esconder a implementacao do servidor.

Exemplo: Servidor (objecto pré-existente)

```
class Server {
    static void Main() {
        TcpChannel channel = new TcpChannel(8086);
        ChannelServices.RegisterChannel(channel, false);
        MyRemoteObject mo = new MyRemoteObject();
        RemotingServices.Marshal(mo,
                "MyRemoteObjectName",
                typeof(MyRemoteObject) );
        System.Console.WriteLine("<enter> para sair...");
        System.Console.ReadLine();
    }
}

public class MyRemoteObject : MarshalByRefObject {
    public string MetodoOla() {
        return "ola!";
    }
}
```