Message Brokers

Amadeu Dias
amadeu@di.fc.ul.pt

Estrutura

- O que são Message Brokers
- O Porquê!
- Arquitectura Geral
- Publisher-Subscriber
- Aspectos a ter em conta
- Referências

O que são Message Brokers

- Middleware MOM específico:
  - Também chamados de Integration Brokers
  - Orientado para interacções entre “servidores”
- Tratam de:
  - Encaminhamento e distribuição;
  - Processamento;
  - Filtragem ...

  de **mensagens** para diferentes aplicações

O porquê!

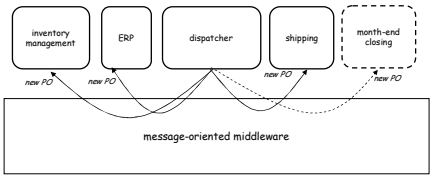
- Insuficiência dos mecanismos já existentes:
  - RPC
  - MOM básico
- Lidar com processos complexos de :
  - Interacções Assíncronas
  - Endereçamento Dinâmico
- ...sendo necessário extender o MOM básico

O porquê! Exemplo

- Caso
  - Processamento de uma ordem de encomenda numa cadeia de produção

O porquê! Exemplo

- Ambiente



O porquê! Exemplo

- MOM básico
  - Cada aplicação é que decide para que filas e/ou destinatários a enviar
- Ordem de encomenda (OE):
  - Origem
    - Dispatcher
  - Destino
    - Inventory
    - ERP
    - Shipping
- Encaminhamento
  - O Dispatcher têm que explicitar quais as filas / aplicações que tem que receber a OE
  - Cada mensagem só seria entregue a uma aplicação pelo que teriam existir várias filas (uma por aplicação interessada)
  - Tudo isto suportado pela aplicação que envia (Dispatcher)

O porquê! Exemplo

- Problemas
  - Ao surgir nova aplicação interessada (Mont-end closing) :
    - A aplicação Dispatcher tem que re-escrever a lógica de encaminhamento necessária para escrever para as filas

O porquê! Exemplo

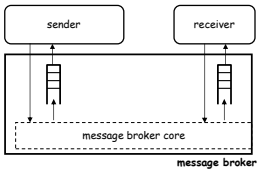
- Message Broker (MB)
  - São explicitadas regras de encaminhamento por cada aplicação que recebe mensagens
- Exemplo
  - O Dispatcher diz ao MB que tem uma mensagem disponível
  - O MB encarrega-se de fazer o encaminhamento para quem está interessado nessas mensagens
  - As regras podem ser aplicadas ao nível do broker ou de fila

O porquê! Exemplo

- Num MOM
  - Cada aplicação é que tem que saber qual o formato a usar ao escrever para as filas
- Num MB
  - Cada aplicação escreve e recebe no formato que quer mas pode associar no MB uma transformação a aplicar as mensagens quando estas lhes são entregues
- Exemplo
  - O Dispatcher especifíca na OE um preço em $US
  - Ao escrever para a fila da aplicação de Shipping o MB transforma esse valor em €

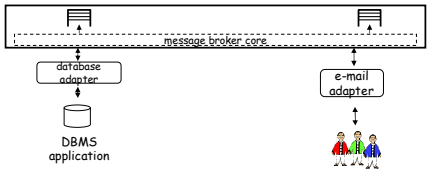
O porquê! Sumário

- Passar para o middleware varias decisões
  - Regras
  - Tipo de encaminhamento
  - Transformações específicas para cada aplicação
- Existe separação lógica entre as aplicações que enviam e que recebem
- Cria-se um sistema capaz de integrar várias aplicações heterogéneas



Arquitectura Geral

- A arquitectura de um sistema de Message Broker é constituída por:
  - Núcleo
  - Adaptadores



Arquitectura

- Núcleo
  - Executa e mantém as regras
  - Faz a gestão das filas internas
  - Mantém os dados

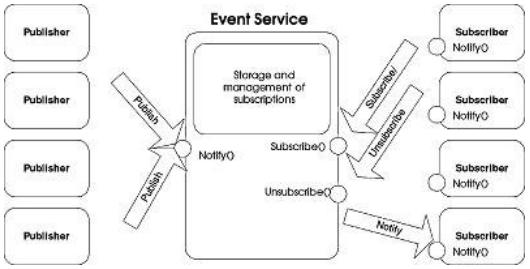
Arquitectura

- Adaptadores
  - Interagem com sistemas específicos
  - Realizam a tradução de mecanismos de comunicação e de esquemas de dados
    - Rede
      - Tamanho de pacotes
      - Representação dos dados (IDL, XML)
    - Esquemas de dados
      - Content Enricher / Content Filter
      - Normalizer

Publisher-Subscriber

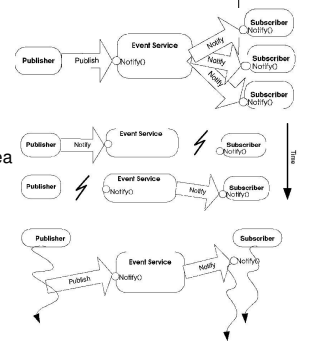
- Implementado na maioria dos Message Brokers
  - CORBA,...
- Assenta sobre:
  - Publisher: produz conteúdos que são consumidos
  - Subscriber: "assina" um conjunto de conteúdos que são do seu interesse

Publisher-Subscriber



Publisher-Subscriber propriedades

- Separação:
  - Espaço
    - Anonimato e confidencialidade
  - Tempo
    - Não permanência simultânea
  - Sincronização
    - Não obrigatoriedade de bloqueio nas actividades



Publisher-Subscriber Comparações

- PS vs RPC
  - O vulgar mecanismo de RPC não tem separação no:
    - Tempo: ambos tem que participar activamente na acção
    - Espaço: quem invoca tem que ter referência para o serviço invocado
    - Sincronização: quem invoca fica bloqueado à espera de resposta
  - Com RPC assíncrono evita-se o bloqueio de quem invoca (não espera resposta)

Publisher-Subscriber Comparações

- PS vs Filas de Mensagens (MOM)
  - Apresenta separação a nível de:
    - Tempo: as mensagens são guardadas e não requerem a participação simultânea de ambos
    - Espaço: Os participantes não necessitam de saber que produz ou consome
  - A nivel de sincronização:
    - O consumidor necessita de ficar bloqueado aquando da aquisição das mensagens das filas

Publisher- Subscriber Classificações

- Push
  - Os intervenientes são passivos
- Pull
  - Os intervenientes são activos

Publisher- Subscriber Classificações

- Topic-based
  - Cada subscriber assina um tópico individual descrito por uma “keyword”
  - Cada tópico fica associado assim a um canal
  - Cada topico é visto como um serviço próprio
  - Tipos endereçamento:
    - Explo: Newsgroups
    - Flat (informatica)
    - Hierachical (informatica.disciplinas.pg.tm)

Publisher-Subscriber Classificações

- Content- based
  - Sujeito às propriedades ou meta-dados da mensagem
  - As propriedades dos eventos a receber são expressas por:
    - Condições de filtragem aquando da inicialização da subscrição
    - Eventos padrão que são inicializados com as propriedades a observar

Publisher-Subscriber Classificações

•Content-based (exemplo)

```
public class StockQuote implements Serializable {
   public String id, company, trader;
   public float price;
   public int amount;
}
public class StockQuoteSubscriber implements Subscriber {
   public void notify(Object o) {
      buy();       // company == 'TELCO' and price < 100
   }
}
// ...
String criteria = ("company == 'TELCO' and price < 100");
Subscriber sub = new StockQuoteSubscriber();
EventService.subscribe(sub, criteria);
```

Publisher-Subscriber Classificações

- Type-based
  - Existe um conjunto pré-definido de tipos de eventos que podem estar hierarquizados
  - No momento da subscrição explicita-se que apenas estamos interessados em subscrever um determinado tipo de eventos
  - Apenas esse tipo de eventos é entregue ao subscriber

Publisher-Subscriber Classificações

- Type-based (exemplo)

```
public class LondonStockMarket implements Serializable {
   public String getId() {...}
}
public class Stock extends LondonStockMarket {
   public String getCompany() {...}
   public String getTrader() {...}
   public int getAmount() {...}
}
public class StockQuote extends Stock {
   public float getPrice() {...}
}
public class StockRequest extends Stock {
   public float getMinPrice() {...}
   public float getMaxPrice() {...}
}
public class StockSubscriber implements Subscriber<StockQuote> {
   public void notify(StockQuote s) {
      if (s.getCompany() == 'TELCO' && s.getPrice() < 100)
         buy();
   }
}
// ...
Subscriber<StockQuote> sub = new StockSubscriber();
EventService.subscribe<StockQuote>(sub);
```

MB
Aspectos ter em conta

- Prós +
  - Boa aproximação para integração de aplicações heterogéneas
  - Bom suporte para comunicação assíncrona
- Contras –
  - Devido à lógica associada, o processamento de mensagens grandes pode afectar o desempenho dos sistemas.

Referências

- Gustavo,Casati,Kuno, Machiraju, “Web Services - Concepts, Architecture and Applications”, Springer
- Hohpe,Woolf, “Enterprise Integration Patterns”, Addison Wesley
- Eugster, Felber, Guerraoui,Kermarrec, “The Many Faces of Publish/Subscribe”
- Oki,Pfluegl,Siegel,Skeen,”The Information Bus – An architecture for Extensible Distributed Systems”
- Alonso,Pautasso,”Web Services – Concepts, Architecture and Applications”