Orbix Filters

- "Ganchos" que permitem a execução de código adicional antes ou depois de cada operação
  - » *Per-Process Filters*
    - instalados em por processo servidor ou cliente
  - » *Per-Object Filters*
    - instalados por objeto
    - menos usados (não vamos falar deles)

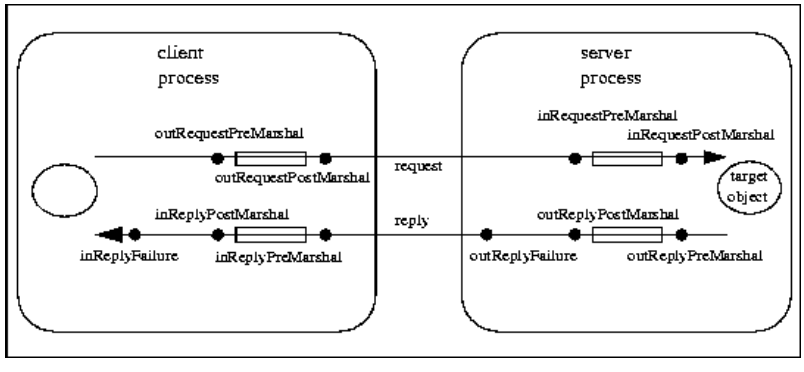
Orbix Filters (cont.)

- Per-Process Filters:
  - » quatro "ganchos" *pre-marshal*
    - dois do lado do cliente
    - dois do lado do servidor
  - » quatro "ganchos" *post-marshal*
    - dois do lado do cliente
    - dois do lado do servidor
  - » dois ganchos de exceção
    - um de cada lado

Filtros: Ganchos Por Processo

- **`outRequestPreMarshal`** (no cliente)
- **`outRequestPostMarshal`** (no cliente)
- **`inRequestPreMarshal`** (no servidor)
- **`inRequestPostMarshal`** (no servidor)
- **`outReplyPreMarshal`** (no servidor)
- **`outReplyPostMarshal`** (no servidor)
- **`inReplyPreMarshal`** (no cliente)
- **`inReplyPostMarshal`** (no cliente)
- **`outReplyFailure`** (no servidor)
- **`inReplyFailure`** (no cliente)

Filtros: Ganchos Por Processo (cont.)



Orbix: Modos Secundários de Ativação de Servidores

- Por cliente (usuário)
  - » cada usuário ganha um servidor próprio
- Por processo client
  - » cada processo cliente ganha um servidor próprio
- Múltiplos clientes
  - » o mesmo servidor é compartilhado por processos clientes de múltiplos usuários

Orbix: Protocolos

- O Orbix suporta dois protocolos
  - » um protocolo proprietário, apelidado ironicamente de POOP (*Plain Old Orbix Protocol)*
  - » o IIOP, protocolo inter-ORB padronizado pelo OMG
- O protocolo default ainda é o POOP
  - » isso mudará numa versão futura do Orbix

Orbix: *Object References*

- *Object references* podem ser "stringficadas" de duas maneiras:
  - » para o formato proprietário usado com o POOP (o formato nativo do Orbix)
  - » para o formato padrão IOR (*Interoperable Object Reference*), usado com o IIOP
- Vamos ver qual a relação entre esses dois formatos

Orbix: *Object References* Nativas

- Formato:

```
:host:server:marker:IR_host:IR_server:IDL_interf
```

- Os seis campos dessa *object reference:*
  - » nome do *host*
  - » nome do servidor
  - » identificador de um objeto dentro do servidor
  - » nome do host do repositório de interfaces
  - » nome do servidor do repositório de interfaces
  - » nome da interface IDL

Orbix: Como Funciona o POOP (1)

- Caso o cliente de posse da *object reference* ainda não tenha estabelecido uma conexão TCP com o processo servidor, a biblioteca do Orbix do lado do cliente contacta o **orbixd** (Orbix *daemon*) da máquina servidora
  - » O campo **`host`** da *object reference* é usado para alcançar a máquina servidora
  - » O **orbixd** está aguardando conexões num *port* bem conhecido dessa máquina

Orbix: Como Funciona o POOP (2)

- » O cliente manda ao **orbixd** uma mensagem **IT_ping** contendo o nome do servidor (campo **`server`** da *object reference*)
- » Caso o servidor não esteja rodando, o **orbixd** aloca um *port* para o servidor e o põe em excução
- » O **orbixd** responde ao **IT_ping** com uma mensagem contendo o port no qual o servidor aguarda conexões
- » A biblioteca do cliente usa esse *port* para se conectar com o servidor

Orbix: Como Funciona o POOP (3)

- O campo **`marker`** é usado para endereçar os objetos implementados pelo servidor
- Note que no POOP os servidores usam *ports* alocados dinamicamente pelo **orbixd** da máquina servidora
  - » O port do servidor não aparece na object reference nativa do Orbix
  - » Vamos ver que ele aparece na IOR (com o IIOP os servidores devem usar ports pré-definidos)