MAC 0434/5765 - Sistemas de Middleware Avançados - 2º Semestre de 2003

Segundo Exercício-Programa: Demarcação de Transações sobre IIOP

Data de Entrega: 10 de novembro de 2003

Neste exercício você adicionará ao JBoss módulos de suporte para demarcação de transações sobre IIOP. Os módulos que você criará permitirão que um cliente CORBA escrito em Java inicie uma transação, invoque uma seqüência de métodos sobre EJBs implantados no JBoss, e efetive (ou aborte) a transação.

Seus módulos serão responsáveis pela propagação do contexto transacional entre o cliente e o servidor de aplicações. Quando uma *thread* do cliente iniciar uma transação, um novo contexto transacional deverá ser criado e associado a essa *thread*. O contexto transacional será implicitamente propagado junto com as requisições IIOP decorrentes das chamadas CORBA efetuadas por essa *thread*, até que a transação seja efetivada ou abortada.

1 O Produto Deste EP

Você produzirá dois módulos:

- um *service MBean* que implementa o suporte para demarcação de transações do lado do servidor;
- uma biblioteca (arquivo jar) com os métodos de demarcação de transações chamados localmente por clientes CORBA.

A biblioteca cliente deverá implementar a interface local `Current` definida no módulo `CosTransactions` do OTS[1], o serviço de transações padronizado pelo OMG. O *service MBean* deverá implementar as seguintes interfaces definidas no módulo `CosTransactions` do OTS: `TransactionFactory`, `Control`, `Terminator` e `Coordinator`.

Ambos os lados (bibioteca cliente e *service MBean*) cuidarão da propagação do contexto transacional entre o cliente e o servidor, que será feira através do campo *service context* das requisições e respostas IIOP. O contexto transacional será transmitido no formato padrão especificado pelo OTS.

Seu *service MBean* não será uma implementação completa do OTS. Ele apenas extrairá o contexto transacional padrão recebido numa requisição IIOP e o traduzirá para um contexto transacional do JBoss. O contexto traduzido será tratado normalmente pelo gerenciador de transações já existente no servidor de aplicações. Como esse gerenciador não tem suporte para transações distribuídas por vários servidores de aplicações, você não precisará propagar para outros servidores o contexto transacional do JBoss, nem tratar transações distribuídas por múltiplos servidores de aplicações. Por isso sua implementação da interface `Coordinator` não precisará ser completa: basta que ela implemente os métodos `rollback_only()` e `get_txcontext()`.

[1] Nas primeiras versões da especificação CORBAServices, o nome desse serviço era *Object Transaction Service*. Posteriormente o nome oficial mudou para *Transaction Service*, mas a sigla OTS continuou a ser usada.

2 Suas Tarefas

Já que você implementará algumas das interfaces do OTS, comece entendendo bem esse serviço. A especificação do OTS está disponível em `http://www.omg.org/technology/documents/formal/transaction_service.htm`. Concentre seu estudo nas interfaces que você vai implementar e na definição do formato do contexto transacional transportado "de carona" com requisições e respostas IIOP.

A propagação do contexto transacional ficará a cargo de um par de interceptadores portáteis: um `PortableInterceptor::ClientRequestInterceptor` registrado com o ORB do lado cliente e um `PortableInterceptor::ServerRequestInterceptor` registrado com o ORB do servidor de aplicações. No cliente, você precisará ainda de um `PortableInterceptor::Current` para transferir o contexto transacional de uma *thread* cliente para uma *thread* emissora de requisições. No servidor, você precisará de outro `PortableInterceptor::Current` para transferir o contexto transacional de uma *thread* receptora de requisições para uma *thread* servente. Para aprender a usar interceptadores portáteis CORBA, veja as seguintes referências:

- a especificação do OMG (`http://www.omg.org/cgi-bin/doc?formal/02-06-25`);
- a seção 9.5 do livro *Java Programming with CORBA (Third Edition)*, de Gerald Brose et al., disponível no xerox do CAMAT;
- o tutorial disponível em `http://java.sun.com/j2se/1.4.2/docs/guide/idl/PI.html`.

O seu *service MBean* precisará interagir com o gerenciador de transações do JBoss em várias situações:

- Quando uma transação for iniciada por um cliente CORBA, ocasião em que o *service MBean* deve interagir com o gerenciador de gerenciador de transações para iniciar a transação e para obter o identificador (`Xid`) da nova transação.
- Quando uma transação for encerrada por um cliente CORBA.
- A cada invocação IIOP sobre um EJB. O *service MBean* deve verificar se a invocação ocorreu dentro de uma transação (ou seja, se um contexto transacional chegou junto com a requisição IIOP). Em caso afirmativo, o *service MBean* fará o gerenciador de transações do JBoss "importar" esse contexto transacional.

Os arquivos fonte do gerenciador de transações do JBoss estão nos seguintes subdiretórios da árvore de fontes:

- `transaction/src/main/org/jboss/tm` (o *transaction manager* propriamente dito)
- `server/src/main/org/jboss/tm/usertx` (o lado cliente e o tratamento da propagação de contexto transacional)

Esse gerenciador de transações implementa a especificação JTA (*Java Transaction API*), disponível em `http://java.sun.com/products/jta/index.html`.

3 Interface entre a Biblioteca Cliente e o *Service MBean*

A biblioteca cliente poderia interagir com o *service MBean* usando apenas as interfaces do OTS já mencionadas (`TransactionFactory`, `Control`, `Terminator` e `Coordinator`). Mesmo assim, é conveniente que o *service MBean* implemente também uma nova interface, cujo único propósito é minimizar o número de chamadas remotas:

```
// interface IDL que estende a interface CosTransactions::TransactionFactory
module org {

    module jboss {

        module tm {

            interface TransactionFactoryExt : CosTransactions::TransactionFactory {
                CosTransactions::PropagationContext create_context(
                                                        in unsigned long time_out);
            };
        };
    };
};
```

Essa interface estende `TransactionFactory` com uma operação que cria uma transação e retorna o contexto transacional correspondente. A nova operação (`create_context`) substitui uma sequência de três chamadas a operações definidas pelo OTS: uma chamada a `create`, seguida de uma chamada a `get_coordinator` (sobre o objeto `Control` retornado por `create`) e de uma chamada a `get_txcontext` (sobre o objeto `Coodinator` retornado por `get_coordinator`).

4 Integração com o Invocador IIOP do JBoss

A cada requisição CORBA, o invocador IIOP precisa obter o contexto transacional que foi extraído da requisição. (A essa altura do processamento da reuisição o `ServerRequestInterceptor` já deve ter rodado e extraído esse contexto.) O invocador IIOP deve associar esse contexto (devidamente convertido para uma instância da classe `org.jboss.tm.TransactionImpl`) ao objeto `Invocation` que representa a invocação corrente. Como o invocador IIOP atual do JBoss *não* efetua tais ações, você precisará mexer nele. Duas alternativas são aceitáveis:

- Alterar o invocador IIOP existente, para que ele interaja com o seu *service MBean*. O invocador IIOP alterado deverá continuar sendo capaz de funcionar sem o seu *service MBean* (e portanto sem oferecer suporte para transações sobre IIOP).
- Substituir completamente o invocador IIOP existente por um outro, que faz parte do conjunto de módulos que voltado para demarcação de transações sobre IIOP. O novo invocador IIOP dependerá do *service MBean* que implementa parte do OTS.

Esta parte do EP talvez pareça trabalhosa e intimidadora, mas as mudanças no código do invocador IIOP devem ser pequenas. Em qualquer das alternativas, o invocador IIOP continuará a ser muito semelhante ao que existe hoje.

5 Testes

Para testar e exercitar seu EP, escreva um EJB bem simples que ofereça métodos tipo débito/crédito. Escreva também dois clientes CORBA: um que faz demarcação de transações e outro que não faz. Moste que uma mesma sequencia de chamadas a métodos do seu EJB pode produzir resultados diferentes, dependendo do fato dela ser executada como uma transação ou não. Na presença de uma exceção como "saldo insuficiente para o débito", a seqüência será parcialmente executada (isto é, seus resultados serão parciais) caso o cliente não tenha demarcado uma transação. Isso nunca deverá ocorrer quando a mesma sequência rodar como uma transação, pois as ações parciais serão desfeitas.

Para evitar dores de cabeça com o mapeamento reverso de Java para IDL, faça os métodos do seu "EJB de testes" receberem e retornarem tipos básicos.

6 Recomendações Finais

Este exercício deve ser feito preferencialmente em equipes de duas pessoas. O ideal é o esquema de "programação pareada" (*pair programming*) de XP. Como alguns podem ter restrições de horário que os impeçam de trabalhar assim, aceitarei também exercícios individuais.

Dúvidas sobre o enunciado devem ser enviadas para `reverbel-sma@ime.usp.br`.

Bom trabalho!