EJB 2.0 - Novas funcionalidades, novos problemas

Por Fabiane Bizinella Nardon

A primeira especificação dos Enterprise Java Beans (EJB) foi lançada há cerca de quatro anos e rapidamente esta tecnologia se tornou uma das escolhas mais comuns para o desenvolvimento de grandes e médias aplicações.

A versão 1.1 da especificação EJB já definia um modelo de desenvolvimento baseado em componentes distribuídos onde serviços básicos necessários a qualquer aplicação são disponibilizados pelo servidor de aplicações, permitindo que o desenvolvedor se concentre em programar a lógica do negócio. Entre estes serviços estão persistência de objetos, controle de transações, concorrência e segurança. As versões 2.1 e 2.0, já implementada na maioria dos servidores de aplicações do mercado, trouxeram uma série de novas funcionalidades que tornam o padrão EJB muito mais interessante. No entanto, a migração das aplicações criadas sob a especificação 1.1 para a versão 2.x pode se tornar um trabalho penoso e frustrante se não se conhecer as armadilhas que esta nova versão apresenta. Antes de tratarmos dos problemas de migração, no entanto, vamos rever rapidamente o que há de novo na versão 2.0:

- **Integração com Java Message Service**

Até a versão 1.1, existiam dois tipo de EJB: os Entity Beans, que tratavam dos aspectos de persistência de objetos, e os Session Beans, que tratavam basicamente da implementação da lógica do negócio. A partir da especificação 2.0, um novo tipo de EJB foi introduzido: os Message Driven Beans (MDB). Os MDB são EJBs cuja função é receber mensagens de um serviço de mensagens Java (JMS). Ao contrário dos Session e Entity Beans, os Message Driven Beans não possuem interfaces Local, Remote ou Home e devem implementar um método chamado onMessage. Quando uma mensagem é gerada pelo serviço de mensagens, o método onMessage é invocado pelo container e o MDB recebe o conteúdo da mensagem. A existência de Message Driven Beans permite a implementação de comportamento assíncrono nas aplicações. Estes componentes são muito utilizados em situações onde um processamento demorado deve ser executado e o cliente não deseja ou não pode esperar pelo resultado do processamento. Imagine, por exemplo, que um sistema de comércio eletrônico recebe um pedido de compra. Ao receber este pedido, ele deve contactar a administradora do cartão de crédito para validar o cartão do cliente, o que pode demorar alguns minutos. Para que o cliente não fique aguardando que esta transação se complete, o sistema envia uma mensagem a um Message Driven Bean com as informações do pedido e continua imediatamente o seu processamento normal, informando ao cliente que o pedido está sendo processado. O Message Driven Bean, por sua vez, recebe a mensagem e inicia todo o processo de comunicação com a administradora do cartão de crédito. Quando o processo acaba, o MDB poderia enviar um e-mail confirmando o pedido.

- **Interfaces Locais**

Até a versão 1.1, os Entity Beans e Session Beans tinham dois tipos de interfaces: as Remote Interfaces e as Home Interfaces. Como a única forma de acessar um destes beans era através destas interfaces, mesmo que a comunicação fosse feita entre EJBs sendo executados no mesmo container, ocorria todo o overhead de uma comunicação entre objetos remotos. A partir da versão 2.0, um novo tipo de interface foi introduzido: as interfaces locais. O acesso a um EJB via uma interface local só pode ser feito por objetos que estão sendo executados na mesma JVM. Ao definir um Session Bean ou um Entity Bean, o desenvolvedor pode escolher prover ao bean uma interface local, uma interface remota ou mesmo ambas. Além disso, podem existir também interfaces Home locais e remotas. A grande vantagem de uma interface local é que a comunicação entre objetos via interface local é muito mais rápida do que a comunicação via interface remota. Em interfaces locais, os parâmetros são passados por referência, enquanto que em interfaces remotas eles são passados por valor (a menos que o parâmetro seja outro EJB). Além disso, o desenvolvedor pode decidir expor alguns métodos na interface local e outros na interface remota ou mesmo expor o mesmo método tanto na interface local quanto na interface remota.

- **Interoperabilidade entre servidores de aplicação**

A partir da versão 2.0, o suporte ao padrão CORBA (Common Object Request Broker Architecture) é obrigatório em todos os servidores de aplicação. Isso garante a interoperabilidade entre objetos sendo executados em diferentes servidores de aplicação.

- **Container Managed Persistence Entity Beans**

Talvez as mudanças mais significativas da especificação 2.0 estejam na nova forma como os Entity Beans CMP são definidos. Na versão 1.1, os campos de um Entity Bean eram definidos como variáveis públicas e era responsabilidade do desenvolvedor implementar seus métodos get e set. Além disso, não existia uma forma padronizada de definir relacionamentos entre Entity Beans. Na versão 2.0, os campos não são mais definidos por variáveis públicas e os métodos get e set são abstratos e sua implementação é gerada pelo container. Os relacionamentos entre Entity Beans são definidos por campos chamados de CMR (Container Managed Relationship), que se encarregam de realizar todas as operações necessárias para manter a sincronia entre os beans relacionados (*delete cascade* e controle de chave estrangeira, por exemplo). Isso significa que se existem duas tabelas Pedido e ItensDoPedido, o Entity Bean Pedido poderá ter um campo CMR itensDoPedido que retornará automaticamente as instâncias do Entity Bean ItensDoPedido relacionadas àquele Pedido.

- **EJB-QL, métodos Select e métodos na Home Interface**

Na especificação 2.0, foi definida uma linguagem de consultas chamada EJB Query Language, ou EJB-QL. Os métodos *finders* e os novos métodos *select* são agora definidos nesta linguagem. Os métodos select são métodos de consulta abstratos que são definidos no próprio bean e não na Home Interface, como são os finders. Estes métodos retornam como resultado da consulta campos do tipo CMP ou CMR e só são acessados internamente dentro do próprio Entity Bean.
Uma nova funcionalidade também introduzida na versão 2.0 é a possibilidade de se definir métodos arbitrários na Home interface. A implementação destes métodos deve ser feita dentro da classe que implementa o bean.

Migrando Aplicações de EJB 1.1 para 2.x

A migração de aplicações desenvolvidas em EJB 1.1 para EJB 2.x se resume no seguinte:

- Para Entity Beans:
  - Re-escrever os deployment descriptors
  - Transformar relacionamentos entre objetos em campos CMR
  - Retirar os campos públicos e definir os métodos get e set como métodos abstratos
  - Decidir se o Entity Bean terá interface local ou remota
  - Definir os finders usando EJB-QL
- Para Session Beans:
  - Decidir de o bean terá interface local ou remota e que métodos serão expostos em cada uma delas.

A partir deste procedimento inicial, pode-se começar a tirar proveito das novas funcionalidades da especificação 2.x e é neste momento que alguns problemas começam a surgir:

- **Interfaces Locais X Interfaces Remotas**: Interfaces locais tratam parâmetros e valores de retorno por referência, enquanto interfaces remotas os tratam por valor. Cuidado ao definir o mesmo método na interface local e na interface remota, pois o comportamento pode ser totalmente diferente. O ideal é utilizar interfaces locais sempre que possível, pois a performance é melhor, e utilizar o design pattern Session Façade, definindo interfaces remotas apenas para os beans que representam o Session Façade. Para Entity Beans, procure criar apenas interfaces locais. A razão para isso, é que campos do tipo CMR não podem ser expostos em interfaces remotas e é muito mais eficiente permitir que os clientes só acessem os Entity Beans através de um Session Bean
- **Validação do Conteúdo de um Entity Bean**: como na versão 1.1 os métodos set dos Entity Beans eram implementados pelo desenvolvedor, era comum colocar a validação dos campos dentro do método set. Como na versão 2.0 os métodos set são abstratos, isso não é mais possível. Existem várias soluções para este problema, mas a mais adequada parece ser não incluir os métodos abstratos set na interface local ou remota, criar novos métodos set que farão a validação e invocarão o método set abstrato e então expor somente os métodos que fazem a validação na interface local ou remota.
- **Mudanças no esquema do banco de dados**: segundo a especificação EJB 2.0, os métodos set dos campos de relacionamento (CMR) dos Entity Beans devem ser executados no método ejbPostCreate() e não no método ejbCreate(). No método ejbCreate() só são colocados os métodos set dos campos que não são CMR. Assim, a seqüência de execução da inclusão de uma nova tupla via um Entity Bean seria: 1) ejbCreate() ; 2) insere no banco de dados; 3) ejbPostCreate(); 4) altera o banco de dados para gravar as chaves estrangeiras correspondentes aos campos CMR. Isso faz com todas as colunas que representam uma chave estrangeira não possam ser do tipo NOT NULL, uma vez que elas sempre receberão valor NULL no momento da primeira inserção. Isso pode ser um problema sério para sistemas que atuam sobre um banco de dados legado.

Existem outras armadilhas na migração de aplicações EJB 1.1 para EJB 2.0, mas as novas funcionalidades disponibilizadas são bastante úteis na construção de grandes sistemas de informação. A especificação EJB 2.x pode ser encontrada em http://java.sun.com/products/ejb/docs.html.