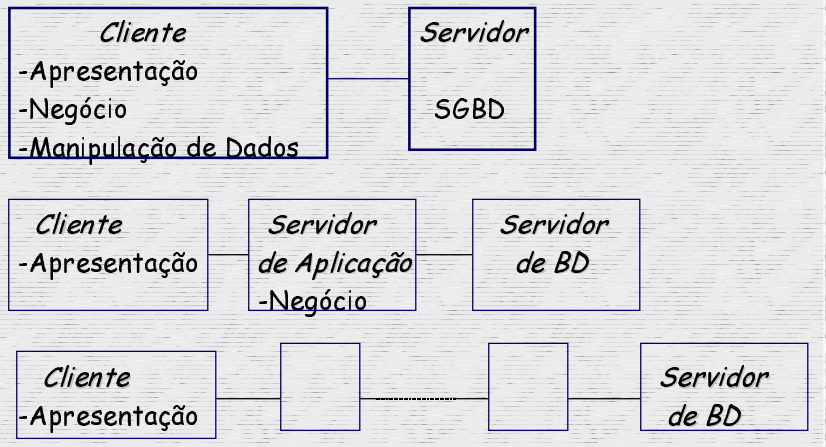
Arquitetura Enterprise Java Beans

Arquitetura EJB

- Define um modelo para desenvolvimento e *deployment* (implantação - instalação preparação do ambiente) de componentes servidores reutilizáveis Java
- Componentes Servidores

  Componentes de aplicação que executam num servidor de aplicação
- Tecnologia EJB ➡ Plataforma Enterprise Java , que suporta desenvolvimento de aplicações baseadas numa arquitetura *multitier* de objetos distribuídos

Arquitetura Multitier X Cliente/Servidor



Servidores de Aplicação - Características desejáveis

- Performance e escalabilidade
  - uso das vantagens de sistemas multithreaded e multiprocessados, compartilhamento de recursos como threads e processos, conexões com BD e sessões
  - replicação e distribuição de componentes em múltiplos sistemas
- Confiabilidade
  - ambiente multitier pode suportar vários níveis de redundância
  - com replicação e distribuição elimina-se gargalos / pontos de falha

Arquitetura EJB

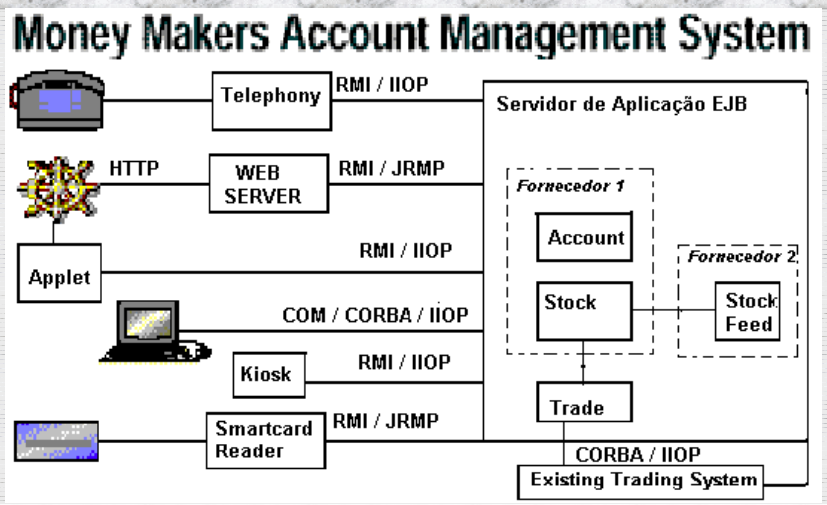
- Servidor de aplicação genérico *(EJB Server)*
- Container *(EJB Container)*
- Componente *(EJB component)*

Cenário

Corretora Money Makers

- Sistema para gerência de carteira de fundos de ações
- Sistema self service de comércio eletrônico
- Suportar clientes e corretores-Alto volume de transações
- Arquitetura de objetos distribuídos
- Cliente "enxuto"
- Acesso: WEB browsers, telefone, smartcard,estações de trabalho
- Protocolos:
  - Clientes Java - Java Remote Method Invocation (RMI)
    (Java Remote Method Protocol -JRMP)
  - Outras linguagens - CORBA IDL sobre IIOP , COM/CORBA
  - Browsers → servlet → aplicação
    HTTP RMI

Exemplo



Plataforma Enterprise Java

- APIs que fornecem acesso a serviços de infraestrutura *enterprise-class* :
  - comunicação distribuída
  - nomes e diretório
  - transações
  - mensagens
  - acesso à dados, persistência e
  - compartilhamento de recursos
- Produtos que implementam EJB podem possuir qualidade de serviços variáveis

Relação de APIs

API	DESCRIÇÃO	API	DESCRIÇÃO
EJB	define modelo de componente servidor que fornece portablilidade e implementam serviços	Servlets e JSP	Java Servlets e Java Server Pages suporta, HTML dinâmico e gerenciamento de sessão p/ clientes usando browser
JNDI	fornece acesso ao servidor de nomes e diretórios	JMS	Java Messaging Service suporta comunicação assíncrona através de vários sistemas de messaging
RMI	cria interfaces remotas p/ computação distribuída na plataforma JAVA	JTA	Java Transaction API fornece demarcação de transação
JAVA IDL	cria interfaces remotas p/ suportar comunicação CORBA. Inclui compilador IDL e um ORB enxuto que suporta IIOP	JTS	Java Transaction Service define um serviço de gerenciamento de transações distribuídas baseado no OTS CORBA
JDBC	fornece acesso uniforme a bancos de dados relacionais		

Arquitetura EJB

- Servidor de aplicação genérico *(EJB Server)*
- Container *(EJB Container)*
- Componente *(EJB component)*

EJB Server

- Servidor de Aplicação Genérico
- Fornecer ambiente que suporte a execução de aplicações EJB
- Fornecer um ou mais containers para os componentes EJB
- Gerenciar e coordenar a alocação de recursos (threads,processos, memória, conexões BD...)
- Fornecer acesso a um conjunto padrão de serviços (de transações, nomes e diretório, segurança, persistência)

EJB Container

- Fornecer processo/thread para execução de um componente
- Fornecer contexto e gerenciar componentes implantados
- Registrar no serviço de nomes e diretórios
- Fornecer interface remota, criar e destruir instâncias
- Gerenciar transações distribuídas
- Gerenciar o estado do objeto para cada enterprise bean
- Opcionalmente pode gerenciar persistência

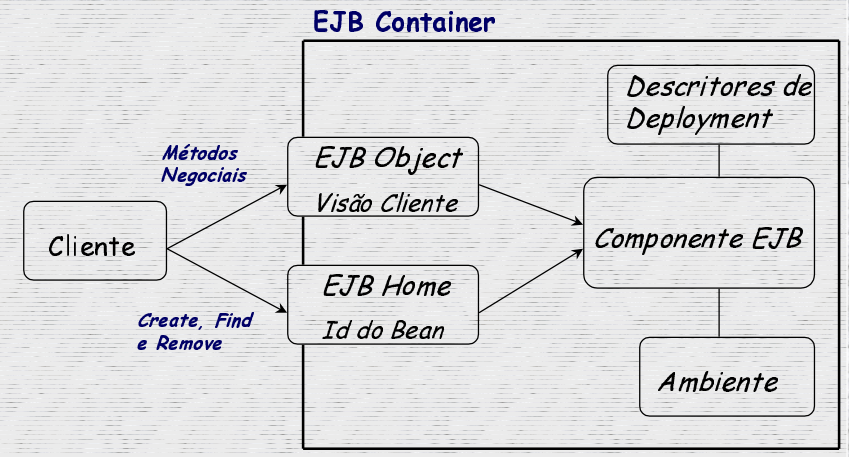
EJB Component

- Componente servidor escrito em Java
- Contém a lógica negocial da aplicação
- Pode ser combinado com outros componentes e/ou código personalizado para produzir uma aplicação
- Podem ser montados e "customizados" em tempo de *deployment* (através de ferramentas fornecidas com *EJB Server*)

Modelo EJB para Componentes

- Padroniza as interfaces entre container e componente servidor (componentes desenvolvidos p/ um container ➡ portávies p/outro)
- Extende o modelo Java Beans p/ suportar componentes servidores
- Servidor de aplicação fornece um container para gerenciar a execução de um componente servidor
- Cliente invoca um componente servidor, o container aloca uma thread para execução do componente
- O container gerencia todas as interações entre o componente, os sistemas externos e recursos

Arquitetura EJB



Container EJB

- Gera as interfaces HOME e OBJECT (Remote) em tempo de implantação *(deployment)*
- Registra a EJBHome num diretório usando a API JNDI
- Usando JNDI , o cliente localiza a EJBHome
- Intercepta todas as requisições do cliente direcionadas as interfaces HOME e OBJECT antes de delegá-las ao componente EJB
- Insere as regras de ciclo de vida, transações, estado, segurança e persistência, baseado nos valores atribuídos aos descritores de *deployment*

Interfaces HOME e OBJECT

- Interface HOME - EJBHome
  - Identifica a classe Enterprise Bean
  - Fornece acesso aos serviços de ciclo de vida do enterprise bean
  - Usada para criar, buscar e remover instâncias enterprise bean
- Interface REMOTE - EJBObject
  - Fornece acesso aos métodos negociais do enterprise bean
  - Representa a visão que o cliente tem do enterprise bean
  - Expõe todas as interfaces relacionadas com a aplicação

Arquitetura EJB

- Suporta Objetos Transientes e Persistentes
  - Objeto Transiente - Session Bean

    Representa um único cliente dentro de um servidor EJB
  - Objeto Persistente - Entity Bean

    Representa , em forma de objeto, dados persistentes mantidos em armazenamentos permanentes , como um BD

Session Beans

- Representa o trabalho sendo executado por um cliente, usando uma ou mais chamadas de métodos
- Na maioria dos casos existe somente durante uma sessão cliente/servidor
- Podem ser transacionais ou não (não recuperáveis após *crash*)
- Podem ser completamente sem estado, ou manter o estado conversacional entre chamadas de métodos e transações gerenciado pelo container, caso precise ser retirado da memória

Session Beans *(cont)*

- Opções de gerenciamento de estado ➡ atributo StateManagementType no objeto Descritor de Deployment.
- Se não possuir estado, o container automaticamente reseta o estado dentro de uma instância depois de cada chamada de método
- Se possuir estado o container automaticamente mantém o estado conversacional do objeto até que o objeto Session seja destruído, mesmo que seja temporariamente retirado da memória

Entity Beans

- Chave primária identifica cada instância de um entity bean
- Criados através da inserção de dados num BD,ou criando um objeto
- Transacionais e recuperáveis depois de um *crash*
- Gerenciar sua persistência ou delegar ao container EJB
- Delegação, o container deverá automaticamente executar as funções de acesso à BD
- Opções de persistência ➡ atributo ContainerManagedFields no objeto descitor de deployment

Contexto e Propriedades

- Objeto Contexto
  - Instância ativa enterprise bean ➡ instância de contexto (regras de gerenciamento e estado corrente da instância enterprise bean)
  - Session Bean usa o objeto Session Context
  - Entity Bean usa o objeto Entity Context
- Tabela de Propriedades - Environment Object
  - Associada a cada enterprise bean
  - Contém valores customizados de propriedades (localização BD, linguagem default)
  - Estabelecidos no processo de deployment, ou na montagem da aplicação

Gerenciamento de Transações

- A especificação sugere, mas não impõe, transações baseadas na API JTS (Java Transaction Service)
- JTS binding do CORBA OTS (Object Transaction Service)
- JTS suporta transações distribuídas
- Duas alternativas:
  - As aplicações EJB comunicam com o serviço de transacões usando a API JTA (Java Transaction Api), que fornece a interface de programação para iniciar transacões, juntar-se a transações existentes, dar *commit* e *roll-back* em transações
  - Todas as funcões de transação podem ser executadas pelo container e pelo EJB Server ➡ atributo TransactionAttribute especificado no objeto descritor de *deployment*

Transaction Attribute

- **TX_BEAN_MANAGED** ➡ demarcação manual de transações usando JTA
- **TX_NOT_SUPPORTED** ➡ o componente não pode ser executado dentro do contexto de uma transação. Se o cliente ao invocar o componente EJB possuir uma transação o container suspenderá esta transação até o final da chamada ao método do componente
- **TX_SUPPORTS** ➡ o componente pode rodar com ou sem o contexto de transação. Se o cliente possuir uma transação na chamada ao método do componente, o método irá rodar neste contexto, caso contrário roda sem a transação

Transaction Attribute *(cont)*

- **TX_REQUIRED** ➡ o componente precisa rodar dentro do contexto de uma transação. Se o cliente não possuir uma transação na chamada ao método do componente, o container automaticamente cria uma transação nova para o método.
- **TX_REQUIRES_NEW** ➡ o componente precisa rodar dentro do contexto de uma nova transação. O container sempre cria uma transação nova para o método. Se o cliente possuir uma transação na chamada ao método do componente, o container a suspende até o témino do método
- **TX_MANDATORY** ➡ o componente precisa sempre ser executado dentro do contexto de uma transação. Se o cliente não possuir uma transação na chamada ao método do componente, o container lança a exceção TransactionRequired

Descritores de Deployment

- Os componentes EJB ➡ empacotados como componentes individuais, coleções de componentes ou como um sistema completo de aplicação. Especificados em XML
- Especifica como criar e manter um objeto componente EJB
- Os componentes EJB ➡ distribuídos num arquivo ejb-jar(indicadores de conteúdo, arquivos contendo as classes EJB, objetos descritores de deployment, e opcionalmente os objetos de propriedades-*Environment Properties Objects*)
- Define: nome da classe EJB, o espaço de nome no JNDI que representa o container, o nome da interface HOME e REMOTE, e o nome do objeto de propriedades
- Os descritores de *deployment* são usados para estabelecer os valores dos atributos de runtime do componente EJB (estabelecidos na montagem ou em tempo de deployment)

EJB Deployment *(cont)*

- Array de objetos ControlDescriptor ➡ semânticas de transação que devem ser aplicadas ao componente
- Array de objeto AccessControlEntry ➡ as regras de segurança a serem aplicadas
- Dois tipos de de descritores de *deployment*
  - **Objeto SessionDescriptor ➡ Session Bean possui ou não estado**
  - **Objeto EntityDescriptor ➡ quais campos dentro do objeto devem possuir persistência gerenciada pelo container**
- Segurança
  - Uso automatizado dos serviços de segurança suportados pela plataforma Java
  - As regras de segurança são definidas através de um conjunto de objetos AccessControlEntry dentro do objeto descritor de *deployment*.

Papéis EJB

- Fornecedor do Sevidor EJB
- Fornecedor do Container EJB
- Fornecedor do Componente EJB
- Montador da aplicação
- Implantador *(Deployer)*
- Administrador de Sistemas

Características da Arquitetura EJB

- Gerenciabilidade
  - Cliente leve mais fácil de gerenciar
- Flexibilidade
  - lógica em módulos, encapsulada atrás de uma interface abstrata
  - alteração de código sem alteração na interface
- Integração e Reusabilidade
  - biblioteca de componentes
  - desenvolvimento "reduzido " à montagem de componentes