Enterprise JavaBeans

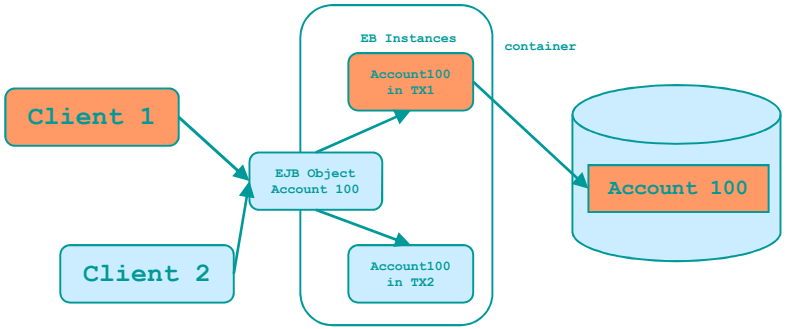
- *"An EJB is a non-visual, remote object designed to run on a server and be invoked by clients."*
- *"…designed to make it easy for developers to create applications, freeing them from low-level system details of managing transactions, threads, load balancing, and so on."*
- component-based framework for building *n*-tier client/server systems
  - components may be plugged into a server
  - allows for non-Java world
    - compatible with CORBA view of the world
- growing support
  - Oracle, IBM, Inprise, Tandem, Symantec, Sybase, BEA, etc.

Enterprise JavaBeans

- a loose synthesis of RMI, JNDI (Java Naming and Directory Interface) and servlets?
  - mechanical aspects for interface derives from RMI
    - will eventually support IIOP (when RMI does)
    - EJB specification is actually protocol neutral
      - JRMP/IIOP/…
- javax.ejb package
  - all interfaces, including:
    - EJBContext
    - EJBHome
    - EJBMetaData
    - EJBObject
    - EntityBean
    - SessionBean
    - SessionSynchronization

Enterprise JavaBeans

- a client creates an EJB instance within the container at the server and manipulates it via an EJB object it as if it were a local object
- EJB instance lives in an EJB container, which
  - *"provides services such as transaction and resource management, versioning, scalability, mobility, persistence and security"*
    - a given container may only provide a subset of these
    - support may be implicit or explicit
      - bean-managed or container managed persistence, for example
- container is itself contained in an EJB server
  - *"Extensible application servers realized"*



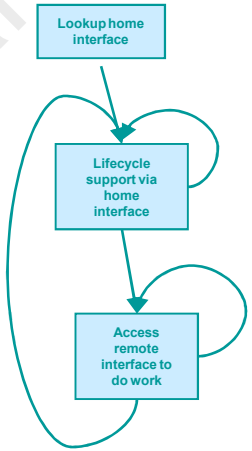
Enterprise JavaBeans

- two types of EJB
  - session bean
    - corresponds to a single client performing some actions on the server
    - for a stateful bean, manages the information relating to a conversation between client and server
    - Bean's ID may be host/port pair or simply a random, unique key generated by the container…
  - entity bean
    - *(optional in v1.0 spec., required in v2.0)*
    - represents and manipulates *persistent* application-domain data—an Employee object, a database tuple or a SELECT result tuple
    - may service multiple clients
    - ID represents the information in some way

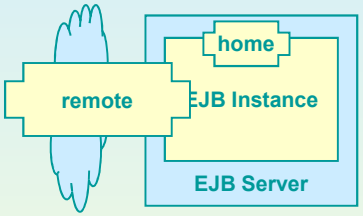
Enterprise JavaBeans

- passivation and activation
  - apply to both session and entity beans
    - *passivation* is the process by which the state of a bean is serialized to persistent storage and then is swapped out
    - *activation* is the process by which the state of a bean is restored from persistent storage
  - in both SessionBean and EntityBean interfaces
    - ejbActivate, ejbPassivate methods
    - only really relevant for a stateful SessionBean
- SessionSynchronization interface
  - allows a SessionBean instance to be notified by its container of transaction boundaries.
  - only required if a bean wishes to synchronize its state with the transactions

Enterprise JavaBeans



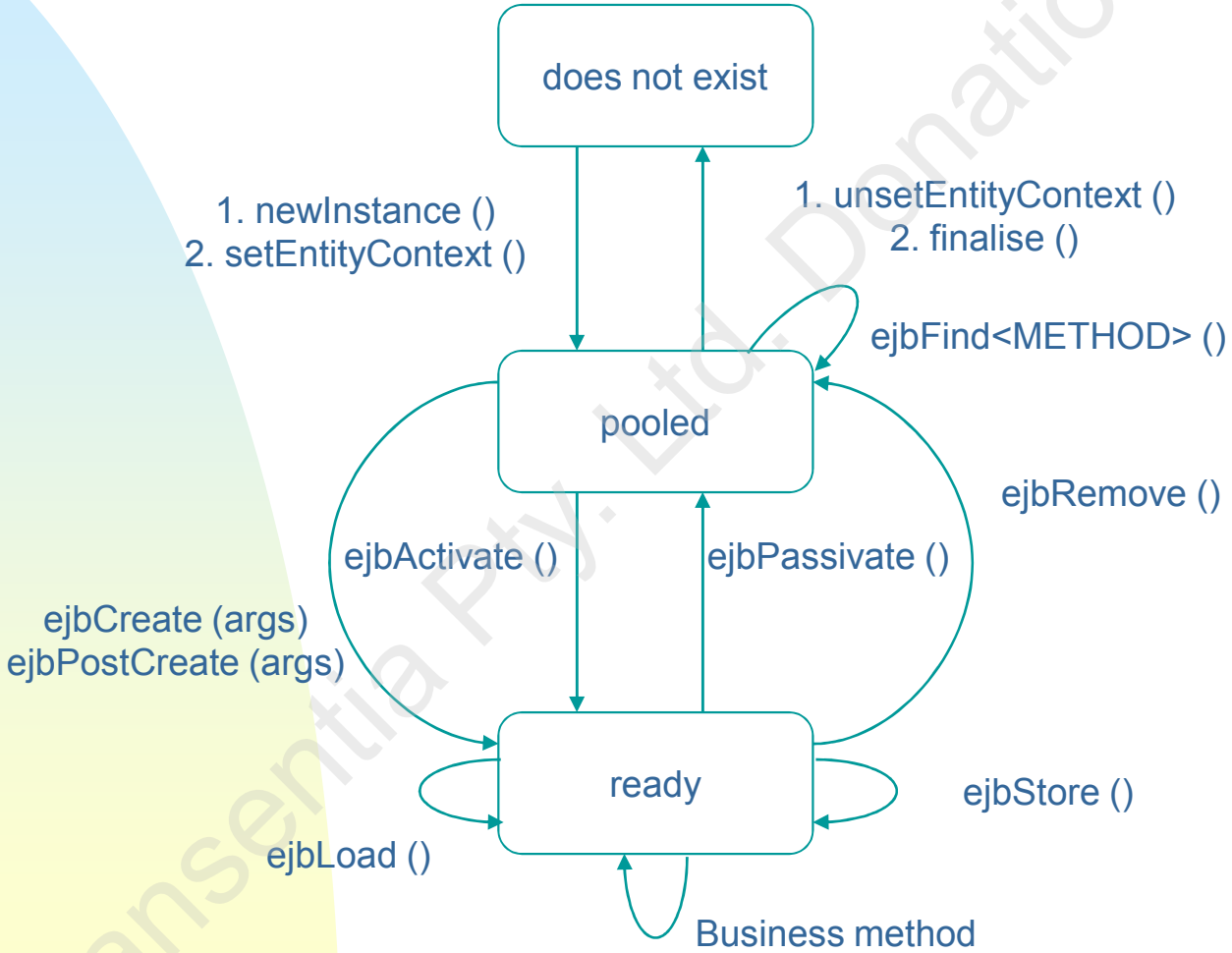
- EJB involves two major interfaces
  - home interface
    - extends javax.ejb.EJBHome interface
    - a contract between EJB and container
    - codifies *lifecycle support* for the EJB instance
      - creating, initializing, destroying and finding EJBs
      - client uses Java Naming and Directory Interface (JNDI) to locate an EJB's home interface



  - remote interface
    - extends javax.ejb.EJBObject interface
    - used by client to call methods & do 'real' work
    - defines the contract between client and EJB (with EJB object acting as proxy for the instance)

Enterprise JavaBeans

the lifecycle of an entity bean instance



Enterprise JavaBeans

- EJB roles
  - server provider
    - runtime context for container provider
    - handles ‘smarts’ of transactions, etc.
  - container provider
    - provides runtime context for EJB
    - provides tools to help deployer
  - enterprise java bean provider
    - writes bean & packages it into an *ejb-jar*
  - deployer
    - put EJB into the backend runtime environment
    - establishes security & other policies for EJB
  - application assembler
    - assemble/combine EJBs into client-side applications

Enterprise JavaBeans

- ejb-jar
  - contains
    - manifest file
    - home interface
    - remote interface
    - bean class file(s)
    - deployment descriptor
      - serialized instance of javax.ejb.deployment.SessionDescriptor
      - used to establish default timeout & stateful/less interaction requirements for bean
      - usually generated automatically by EJB provider
    - serialized Properties instance
      - establishes default environment for bean

Enterprise JavaBeans

```
// a simple client that uses an entity bean
import javax.ejb.*;
import javax.naming.*;
import java.rmi.*;
public class EJBClient
  {
  public static void main (String [] args)
    {
    javax.naming.Context initialContext = new javax.naming.InitialContext ();
    AccountHome accountHome = (AccountHome) initialContext.lookup ("AccountHome");
    Account account = accountHome.create ();
    account.deposit (5000.00);
    System.out.println ("Account " +
      ((AccountPK) account.getPrimaryKey ()).accountId + " created");
    }
  }
```

simple to use…

…but requires a fair bit of work from the implementor

```
// snippet from the implementation of an Entity Bean
import javax.ejb.*;
public class AccountBean implements EntityBean
  {
  private transient SessionContext ctx;
  public double balance;
  public String accountID;
  public AccountPK ejbCreate (string accountID, double balance)
    throws Exception { }
  public deposit (double amount)  { balance += amount; }
  public void ejbLoad () throws Exception { }
  public void ejbStore () throws Exception { }
  public void ejbActivate () throws Exception { }
  public void ejbDestroy () throws Exception { }
  public void ejbPassivate () throws Exception { }
  public void setSessionContext (SessionContext ctx)
    throws Exception
    {
    this.ctx = ctx;
    }
  }
```

```
note cast; needs to be _narrow ()
if using IIOP—loss of portability
```

Enterprise JavaBeans

```
// called by the context when the client calls the create method defined in the home interface
// note that the create methods on an entity bean must return the bean's Primary Key
public AccountPK ejbCreate (String accountID, double balance) throws Exception
  {
  this.accountID = accountID;
  this.balance = balance;
  Connection con = null;
  PreparedStatement ps = null;
  try
    {
    con = getConnection ();
    ps = con.prepareStatement ("INSERT INTO ejbAccounts (id, bal) values (?, ?");
    ps.setString (1, accountID);
    ps.setDouble (2, balance);
    if (ps.executeUpdate () != 1)
      throw new CreateException ("JDBC did not create a row for this bean.");
    AccountPK primaryKey = new AccountPK ();
    primaryKey.accountID = accountID;
    return (primaryKey);
    }
  catch (SQLException sqle)  { }
  finally
    {
    // close ps and con...
    }
  }

// called by the context after ejbCreate but before create returns
// gives the bean a chance to fully initialize itself from with the transaction
public void ejbPostCreate (String accountID, double balance)
  {
  setModified (false);
  }
```

Enterprise JavaBeans

- over time, EJBs will make use of the various new infrastructural APIs:
  - transaction support comes via JTS
    - a low-level API used by sophisticated transactional application programs, resource managers, transaction processing monitors, transaction-aware communication managers, and transaction managers
  - also Java Message Service (JMS)
    - provides a standard Java API for enterprise messaging services such as reliable queuing, publish and subscribe and various aspects of push/pull technologies
  - also Java Management API (JMAPI)
    - a set of extensible objects and methods for the development of seamless system, network, and service management solutions for heterogeneous networks