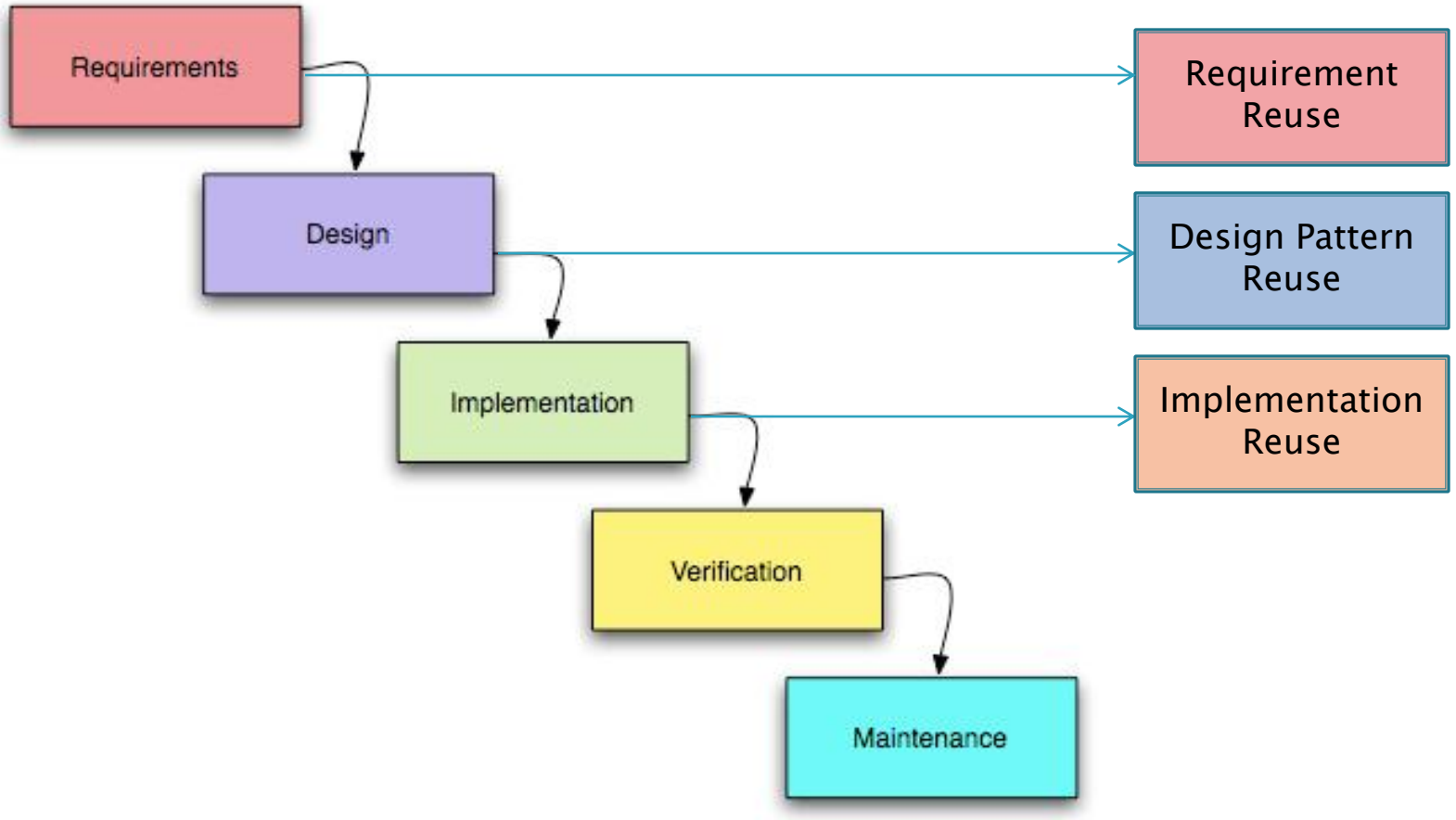
Module 4
Implementation Technologies for Reuse

Part 2. Component Technologies for Reuse

Dr. Shiping Chen

Where Are We?

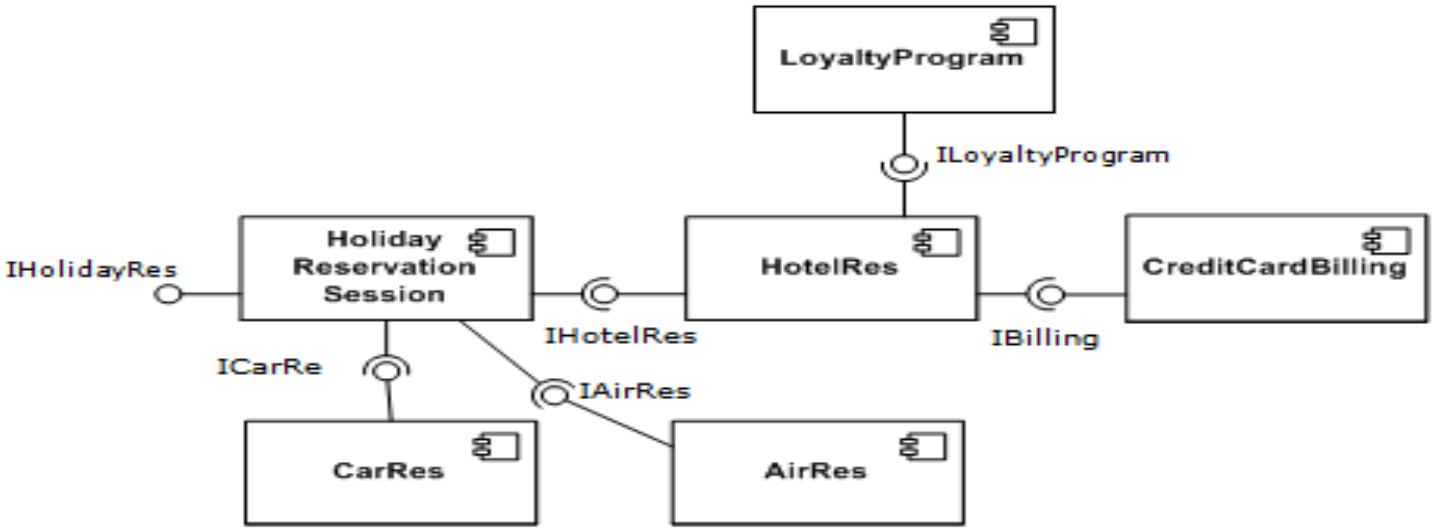


Outline

- CBSE Overview
- Component Technologies
  - Oracle/Sun EJB
  - Microsoft COM+

Software Component

- A (software) component is relatively *independent* software module, which represents a set of specific business *functionalities* (services) or *data* via whose *standard interfaces*.



From: http://en.wikipedia.org/wiki/Component-based_software_engineering

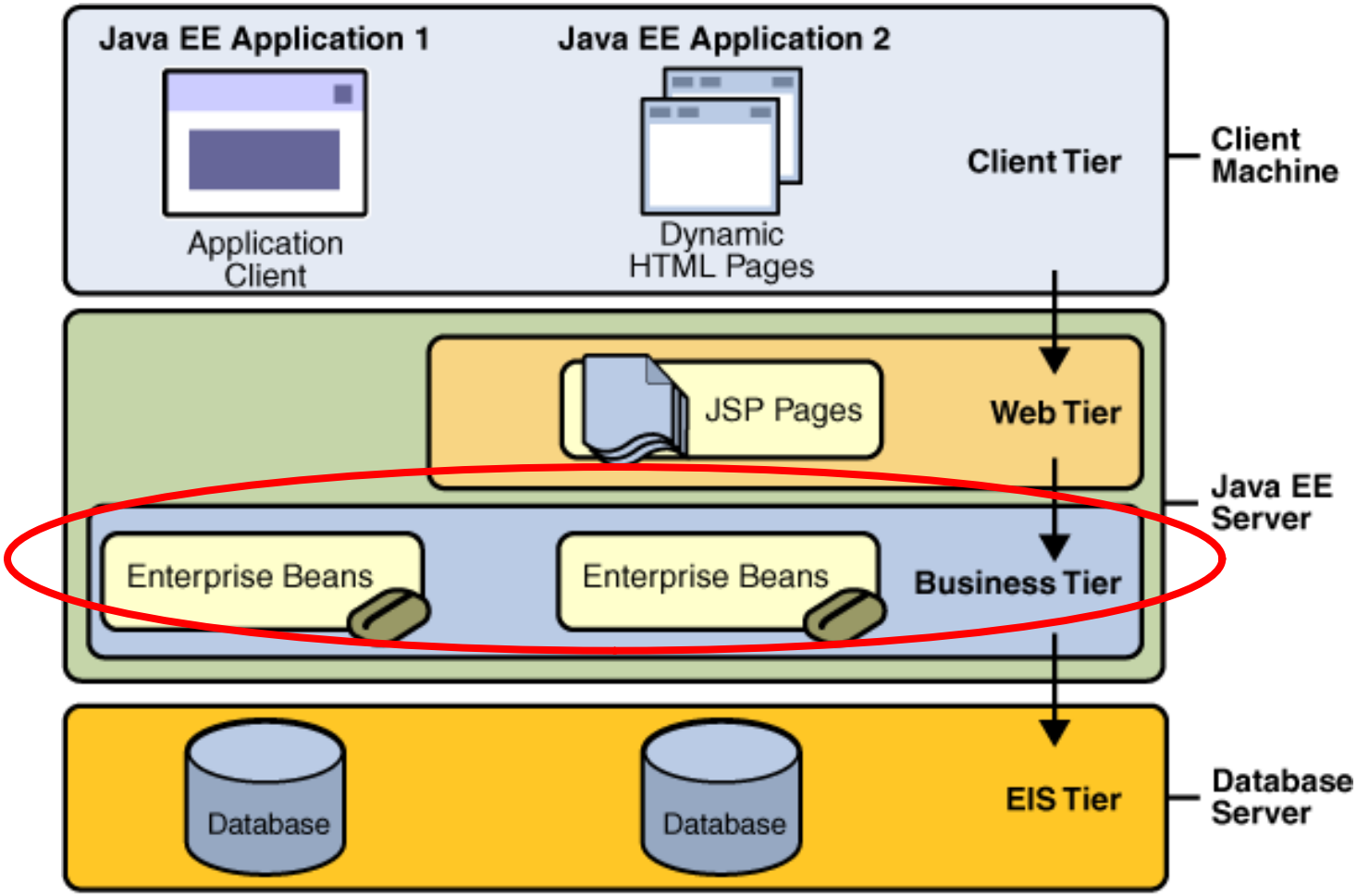
Characteristics of Components

- ***Independent***: may exist autonomously from other components;
- ***Business-oriented****:* not fine-coarse objects
- ***Flexible***: can be deployed within single execution context or across multiple processes/network
- ***Substitutable****: can be replace* at design time or run-time

Component Technologies

- CORBA: Common Object Request Broker Architecture
- EJB: Enterprise Java Bean
- COM+: Common Object Model
- Web Services

J2EE Architecture Review



Source: http://java.sun.com/javaee/5/docs/tutorial/doc/

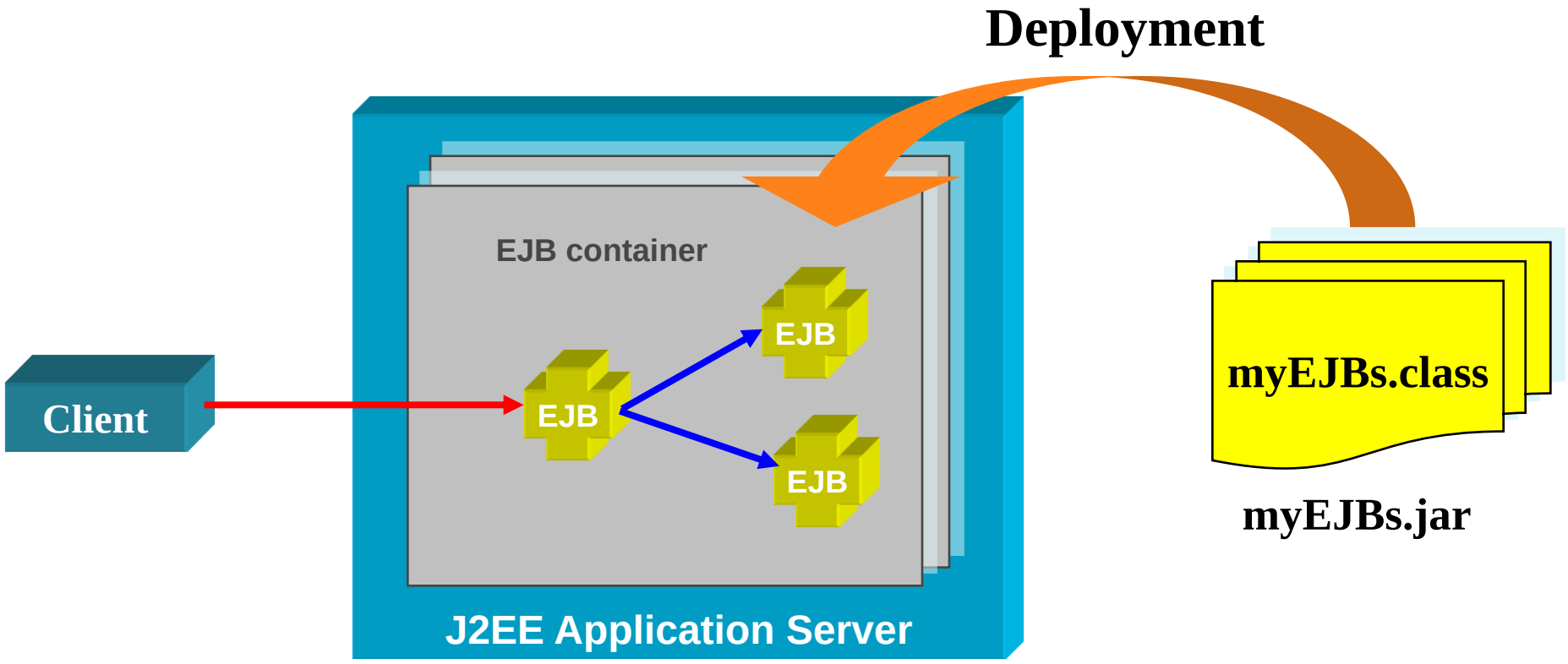
What is EJB?

- EJB = Enterprise Java Bean
- Server side components designed to enable developers to easily build-in business logics with little efforts/worries about system level issues, such as:
  - Life-cycle of objects
  - Concurrency
  - Transaction
  - Security
- Can be reused within a system and across systems

EJB in Evolution

- EJB standards progression:
  - EJB 1.1 – released in Dec. 1999, basic EJB functionalities
    - Session bean, entity beans, XML-based descriptor, etc.
  - EJB 2.0 – released in Aug. 2003, improved with:
    - Message-Driven Bean (MDB), EJB QL for searching, persistence manager, etc.
  - EJB 2.1 – released in Nov. 2003, improved with:
    - Support for Web Services, Introduced EJB timer services etc.
  - EJB 3.0 – released in May 2006, improved with:
    - Simplified EJB interfaces (Removed EJB home interface)
    - Added annotations onto EJB classes and methods
    - New persistence technologies by adopting Hibernate and JDO
- All the major EJB specifications can be downloaded from http://java.sun.com/products/ejb/docs.html

How EJB Works?



- Multiple EJBs can be deployed and managed in an EJB container
- Multiple EJB containers can be hosted by a J2EE application servers

EJB Container

- A ***container*** is an execution environment for a ***component***.
- The component lives in the container and the container provides the services for the component.
- Similarly, a ***container*** lives in an ***application server***, which provides an execution environment for it and other containers.

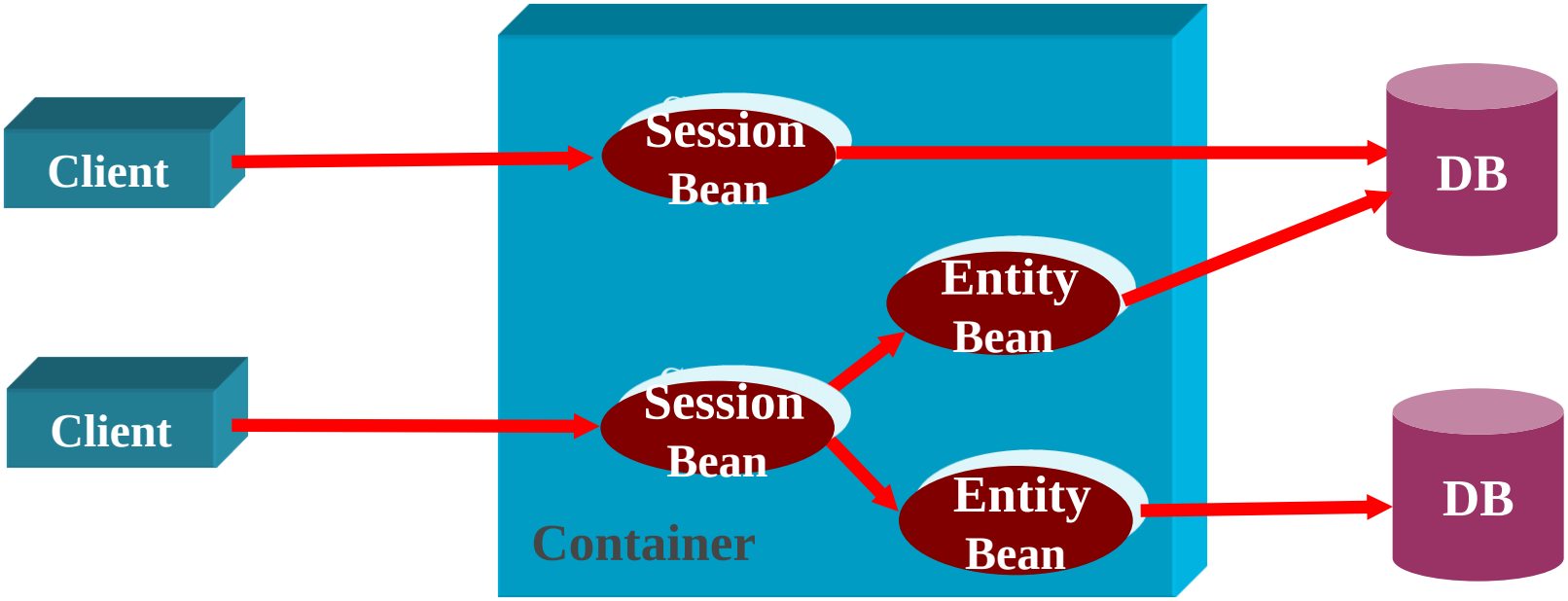
Services Provided by EJB Containers

- Declarative transactions
- Data caching
- Declarative Security
- Error Handling
- Component Framework for Business Logic
- Scalability and Fall-Over
- Portability
- Manageability
- Persistence

Classification of EJBs

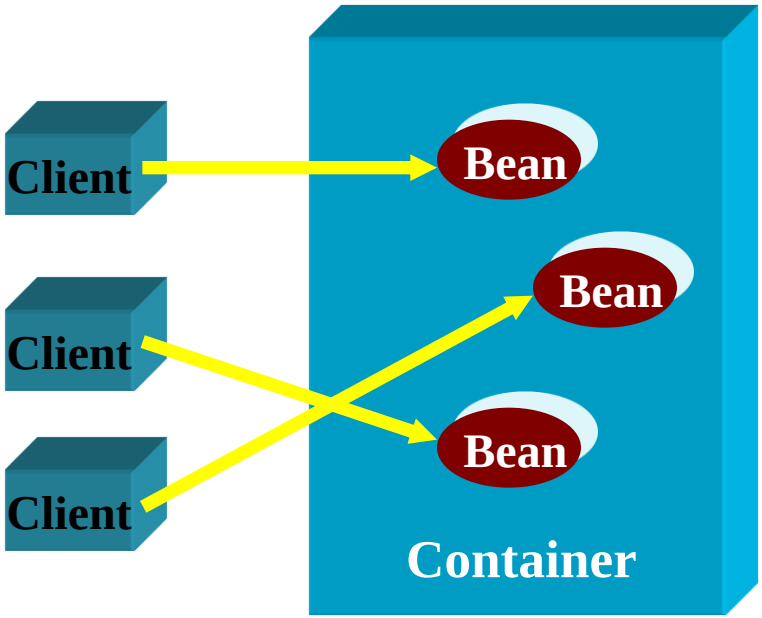
- Session Beans: host business logics
  - Stateless: shared beans
  - Stateful: one client vs one bean
- Entity Beans: host business data
  - BMP: Bean Managed Persistence
  - CMP: Container Managed Persistence
- Message-Driven Beans: Similar to session beans, but driven/triaged by messages
  - Most use JMS as messaging layer

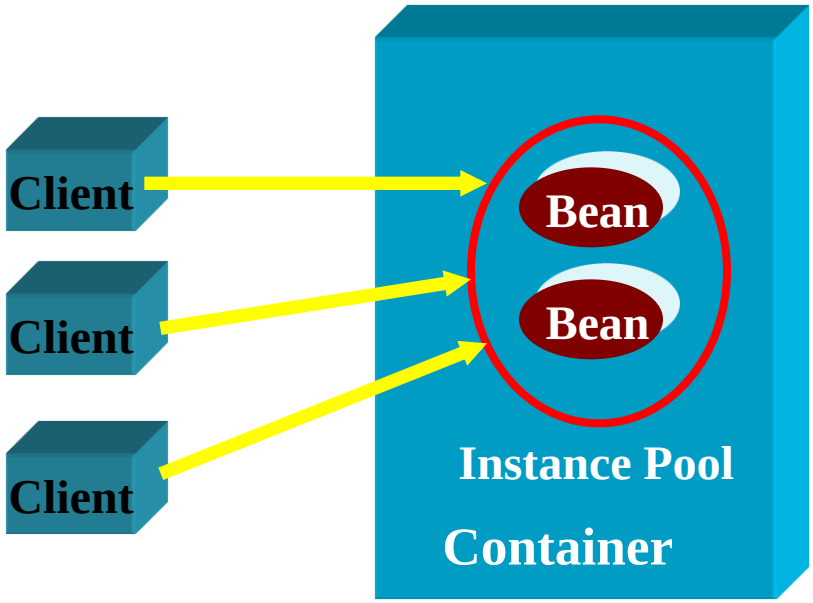
How to Use Session Beans?



- A Session bean represents a single client inside the application server to invoke a set of business functionalities
- A Session can conduct operations on database directly; or
- Or via Entity Beans (to be discussed in the following)

Session Bean: Stateful vs. Stateless





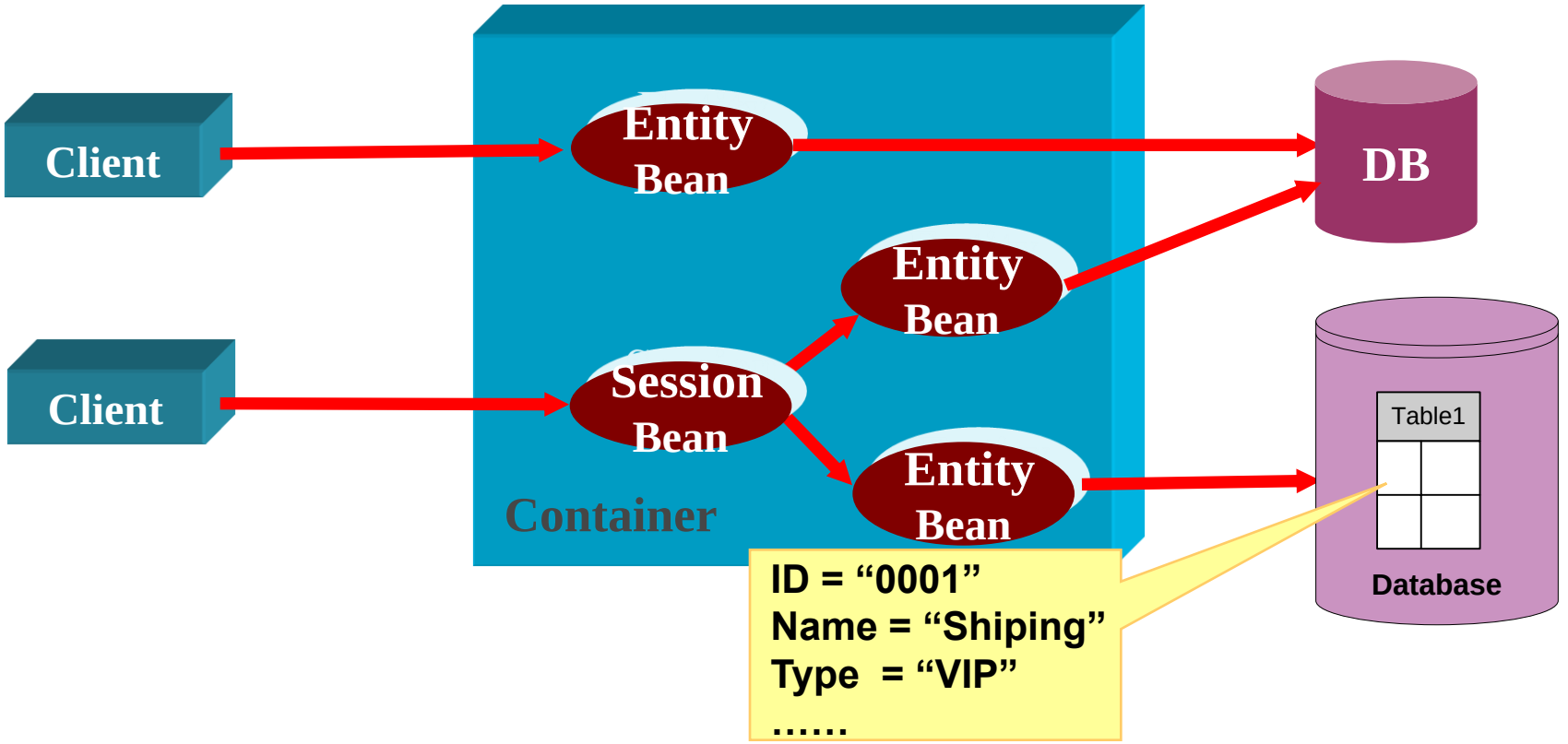
Stateful Session Beans:

- **One client has its own bean.**
- **The states remain persistent between methods**

Stateless Session Beans:

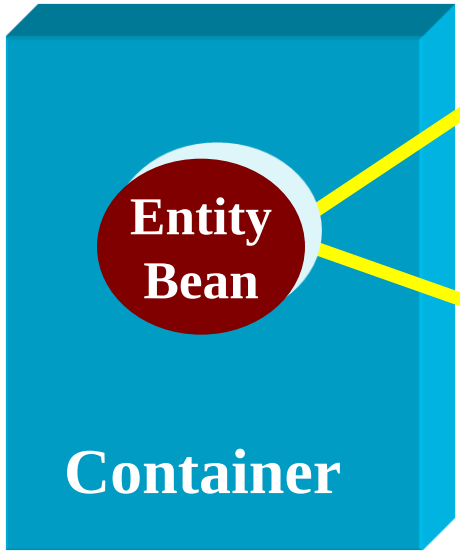
- **Clients share beans in the bean pool**
- **The states remain within a call**
- **Scale better than stateful SE**

How to Use Entity Beans?



- An entity bean class usually represents a database table
- An instance of the entity bean class represents a corresponding record of the database table;
- An entity bean is usually used by session beans;
- But can be directly accessed by external clients as well.

Persistence? BMP vs. CMP



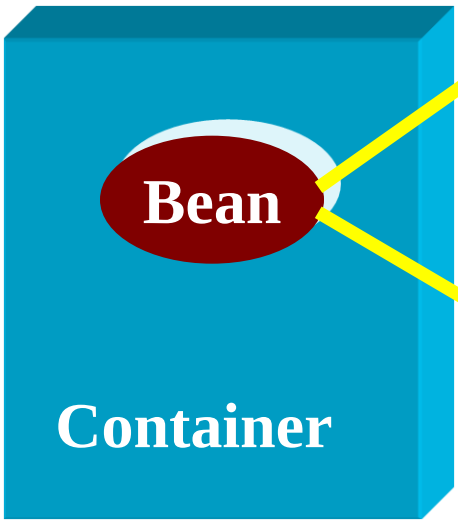

```
public void ejbLoad()
{
    Statement stmt = conn.createStatement();
    String sql = "Select name, type from …";
    stmt.executeQueuey(sql);
}
```

BMP

```
public void ejbLoad()
{

}
```

CMP

Transaction: BMT vs. CMT




```
public void myBusiness()
{
   UserTransaction ut = …
    ut.begin();
              …..
    ut.commit();
}
```

To access DB or other beans...

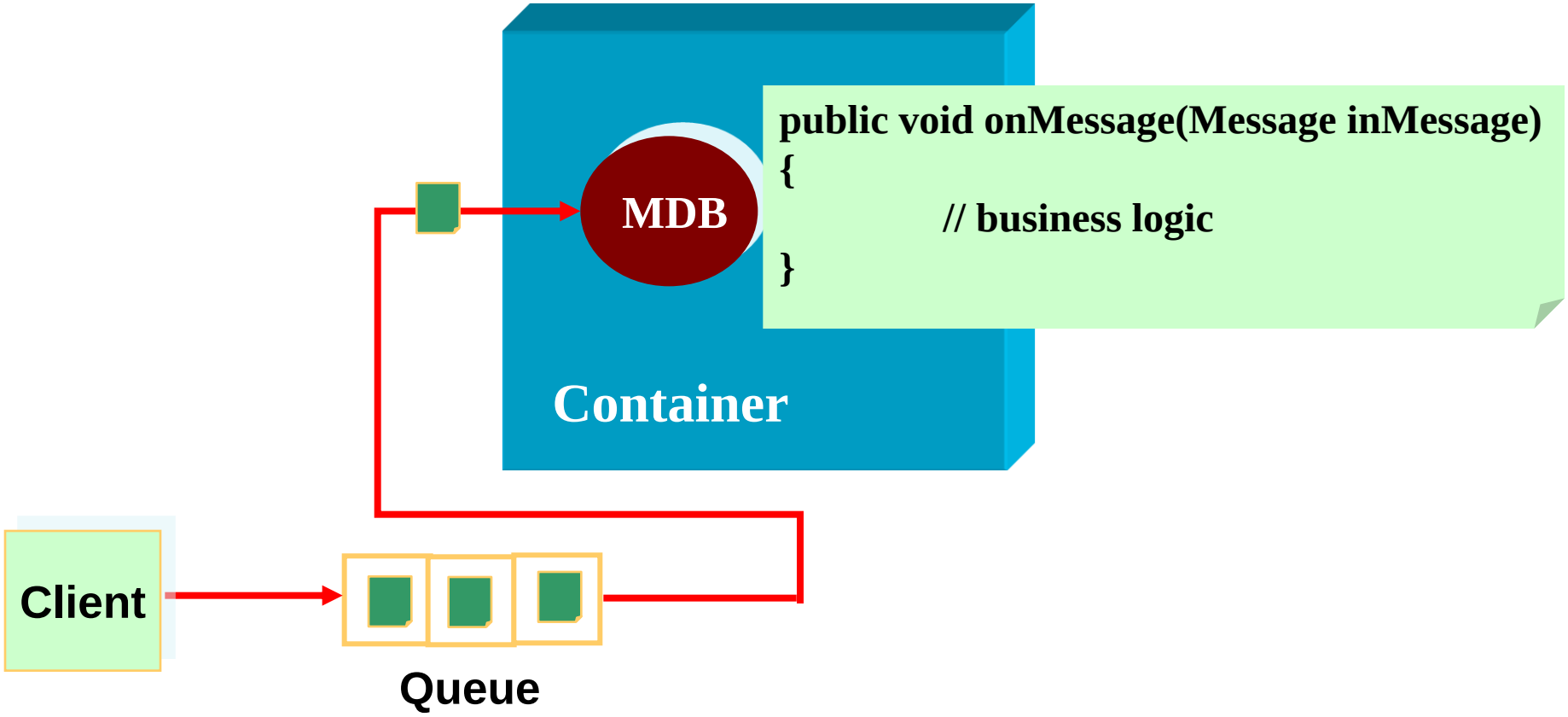
BMT

```
public void myBusiness()
{
              ……
}
```

To access DB or other beans...

CMT

MDB: Message-Driven Beans



Timer Service Bean (TSB)

- A special session bean that can be driven/triggered by time.

```
@Remote
public interface TimerServiceRemote {

        public void setTimer(long interval);
        public void setTimer(long firstInterval, long lateInterval);

        public void timeout(Timer timer);
}
```

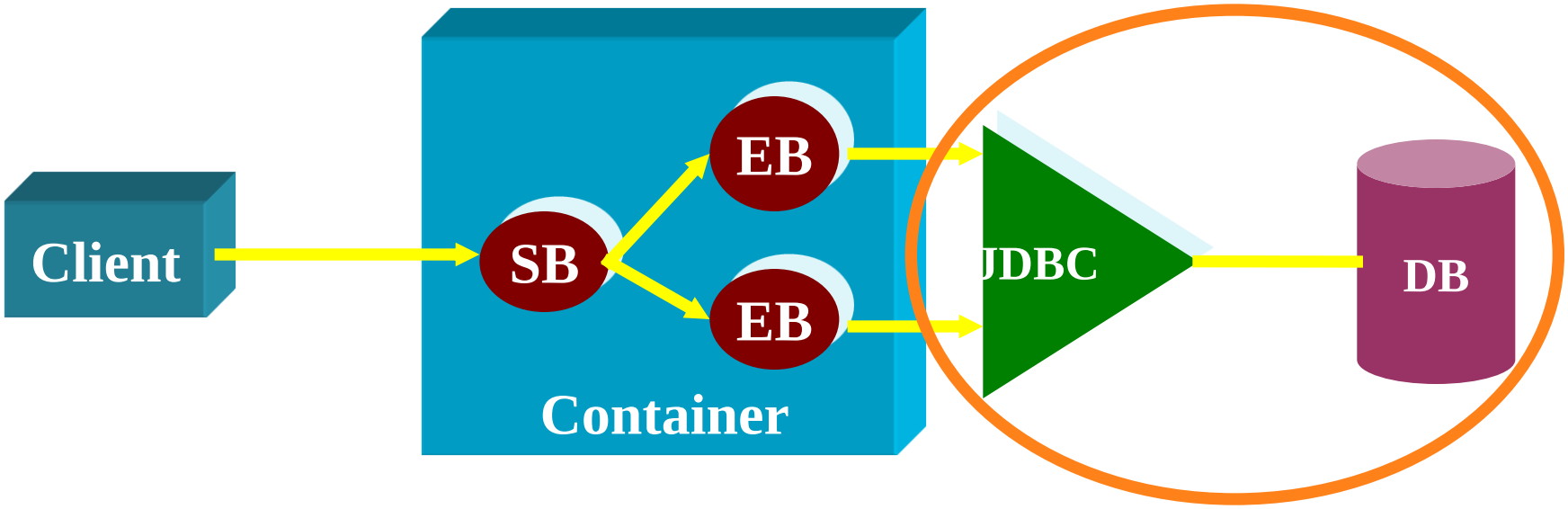
•public void setTimer(long interval);
- Timeout() will be called after interval

• public void setTimer(long firstInterval, long feqInterval);
- Timeout() will be called after firstInterval
- Timeout will be called at the frequency of freqInterval

How to use TSB?

```
@Stateless
public class TimerServiceBean implements TimerServiceRemote {

    @Resource
    TimerService timerService;

    public void setTimer(long interval)  {
        Timer timer = timerService.createTimer(interval, "Created new timer");
    }

    @Timeout
    public void timeout(Timer timer)  {
        // Do something ….
    }
}
```

JDBC in J2EE



SB: Session Bean
EB: Entity Bean

Wrapped as Data Source

JDBC is a bridge between EJB Application servers and Database.

EJB's Interface

- **Before EJB 3**
  - *Home Interface: Used to create/release EJBs*

```
public interface BrokerHome extends javax.ejb.EJBHome
{
      Broker create() throws java.rmi.RemoteException, javax.ejb.CreateException;
}
```

  - *Remote Interface: Used for app-related calls*

```
public interface Broker extends EJBObject
{
      public int newAccount(String name, String address, int credit) throws
                                    RemoteException, Exception;
      ……
}
```

EJB's Interface (cont'd)

- **EJB 3**
  - *Business (local/remote) Interface only:*

```
@remote
public interface Broker
{
      public int newAccount(String name, String address, int credit);
      ……
}
```

```
@stateless
public class BrokerBean implements Broker
{
      private int accountID;
      …

      public int newAccount(String name, String address, int credit)
      {
            ……
      }
}
```

Deployment Descriptor, i.e. ejb-jar.xml

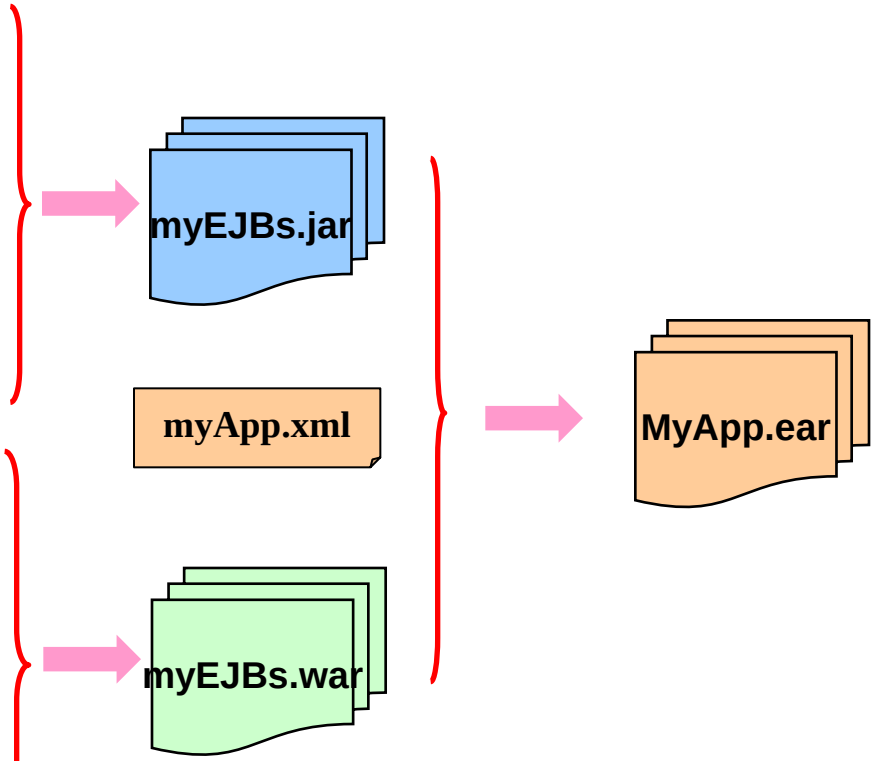
```
<ejb-jar>
  <enterprise-beans>
      <description>specify all EJBs</description>
  </enterprise-beans>

  <assembly-descriptor>
      <description>specify transaction, security etc.
  </description>
  </assembly-descriptor>
</ejb-jar>
```

Deployment Descriptor: An Example

```
<ejb-jar>
   <enterprise-beans>
      <session>
        <description>statelesss session bean</description>
        <display-name>myBroker</display-name>
        <ejb-name>myBroker</ejb-name>
        <home>com.csiro.stockonline.BrokerHome</home>
        <remote>com.csiro.stockonline.Broker</remote>
        <ejb-class>com.csiro.stockonline.BrokerBean</ejb-class>
        <session-type>Stateless</session-type>
        <transaction-type>Container</transaction-type>
        <resource-ref>
          <res-ref-name>jdbc/StockDB</res-ref-name>
          <res-type>javax.sql.DataSource</res-type>
          <res-auth>Container</res-auth>
        </resource-ref>
      </session>
   </enterprise-beans>
   <assembly-descriptor>
      <container-transaction>
           <method>
            <ejb-name>myBroker</ejb-name>
            <method-name>*</method-name>
           </method>
          <trans-attribute>Required</trans-attribute>
       </container-transaction>
   </assembly-descriptor>
</ejb-jar>
```

Packaging EJB & Web Application

- [EJB_Root_Dir]
  - *.class files
  - [META-INF]
    - ejb-jar.xml
    - myDeploy.xml
- [WEB-Root_Dir]
  - *.classes
  - *.jsp
  - *.html
  - [META-INF]
    - Web-jar.xml



- The *.jar and *.ear can both be deployed separately
- The deployment can be done with either IDE or ant

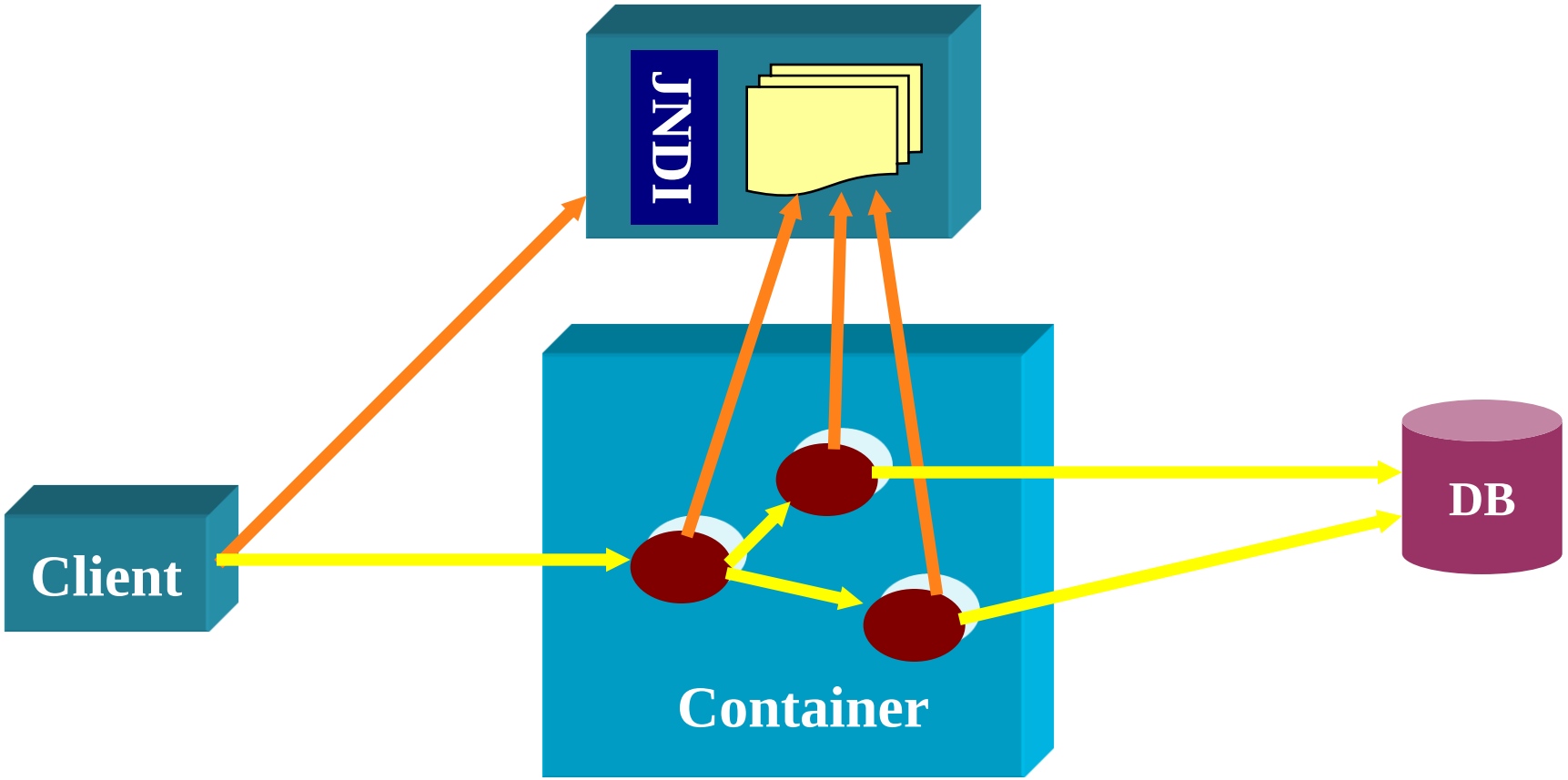
EJB Clients

- There are two possible types.
  - The first category is application clients which are stand-alone applications accessing the EJB components using the RMI-IIOP (remote) protocol.
  - The second category of application clients are components in the web container (e.g. servlets and JSPs) using local or remote interface.

From EJB Clients' Viewpoint

- The client has a smaller set of concerns than a bean developer with regard to using EJBs. Basically, he need to know
  - how to find a bean
  - how to use its methods
- And do NOT need not know:
  - the implementation of the EJB
  - callbacks that the EJB container will make on the EJB or
  - nature of the services provided to the EJB.

JNDI – Java Naming and Directory Interface



JNDI can be used to locate beans and data sources.

How to use JNDI?

1. Get initialContext, i.e. the root of the naming service (for J2EE):
   Context ctx = new InitialContext(???);

2. Look up a specific EJB by name:
   Object objRef = ctx.lookup("myBroker");

How to Set the InitialContext Properties?

- By Programming

```
package com.csiro.stockonline

import java.util.properties;
import java.naming.Context;
import java.naming.InitialContext;
…
public Class Client
{
      ….

      Properties prop = new Properties();
      // server-dependent properties for InitialContext
      prop.put(Context.INITIAL_CONTEXT_FACTORY,
            “org.jnp.interfaces.NamingContextFactory”);
      prop.put(Context.PROVIDER_URL, “localhost:1099”);
      Context ctx = new InitialContext(prop);
      …
```

How to Set the InitialContext Properties?

- Using Command-line

```
package com.csiro.stockonline

import java.util.properties;
import java.naming.Context;
import java.naming.InitialContext;
…
public Class Client
{
        ….

        Context ctx = new InitialContext();
        …
```

```
C:\MyProject\Stockonline>java -DContext.INITIAL_CONTEXT_FACTORY=
        “org.jnp.interfaces.NamingContextFactory” -DContext.PROVIDER_URL=“localhost:1099”
        com.csiro.j2ee.stockonline.Client
```

How to Set the InitialContext Properties?

- Using Property File (My preference way)

```
Context.INITIAL_CONTEXT_FACTORY=org.jnp.interfaces.NamingContextFactory
Context.PROVIDER_URL=localhost:1099
……
```

```
package com.csiro.stockonline

import java.util.properties;
import java.naming.Context;
import java.naming.InitialContext;
…
public Class Client
{
        ….

        Properties prop = new Properties();
        prop.load( new java.io.FileInputStream( args[0]) );
        Context ctx = new InitialContext(prop);

        …
```

```
C:\MyProject\Stockonline>java com.csiro.stockonline.Client my.prop
```

Put It All Together as An EJB Client

- Client for ‘old EJBs’ (Pre-EJB 3)

```
package com.csiro.stockonline

import java.naming.Context;
import java.naming.InitialContext;

public class Client
{
    try
    {
            Context ctx = new InitialContext();
            Object objRef = ctx.lookup(“myBroker”);
            BrokerHome brokerHome = (BrokerHome) objRef;
            Broker broker = brokerHome.create();

            int accountID = broker.newAccount(“Shiping Chen”……);

    }
    catch(Exception e)
    {
            e.printStackTrace();
    }
}
```

Put It All Together as An EJB Client

- EJB 3 Client

```
package com.csiro.stockonline

import java.naming.Context;
import java.naming.InitialContext;

public class Client
{
    try
    {
            Context ctx = new InitialContext();
            Broker broker = new Broker;
            broker = (Broker) ctx.lookup("myBroker");

            int accountID = broker.newAccount("Shiping Chen"…);
    }
    catch(Exception e)
    {
            e.printStackTrace();
    }
}
```

What can't you do within an EJB?

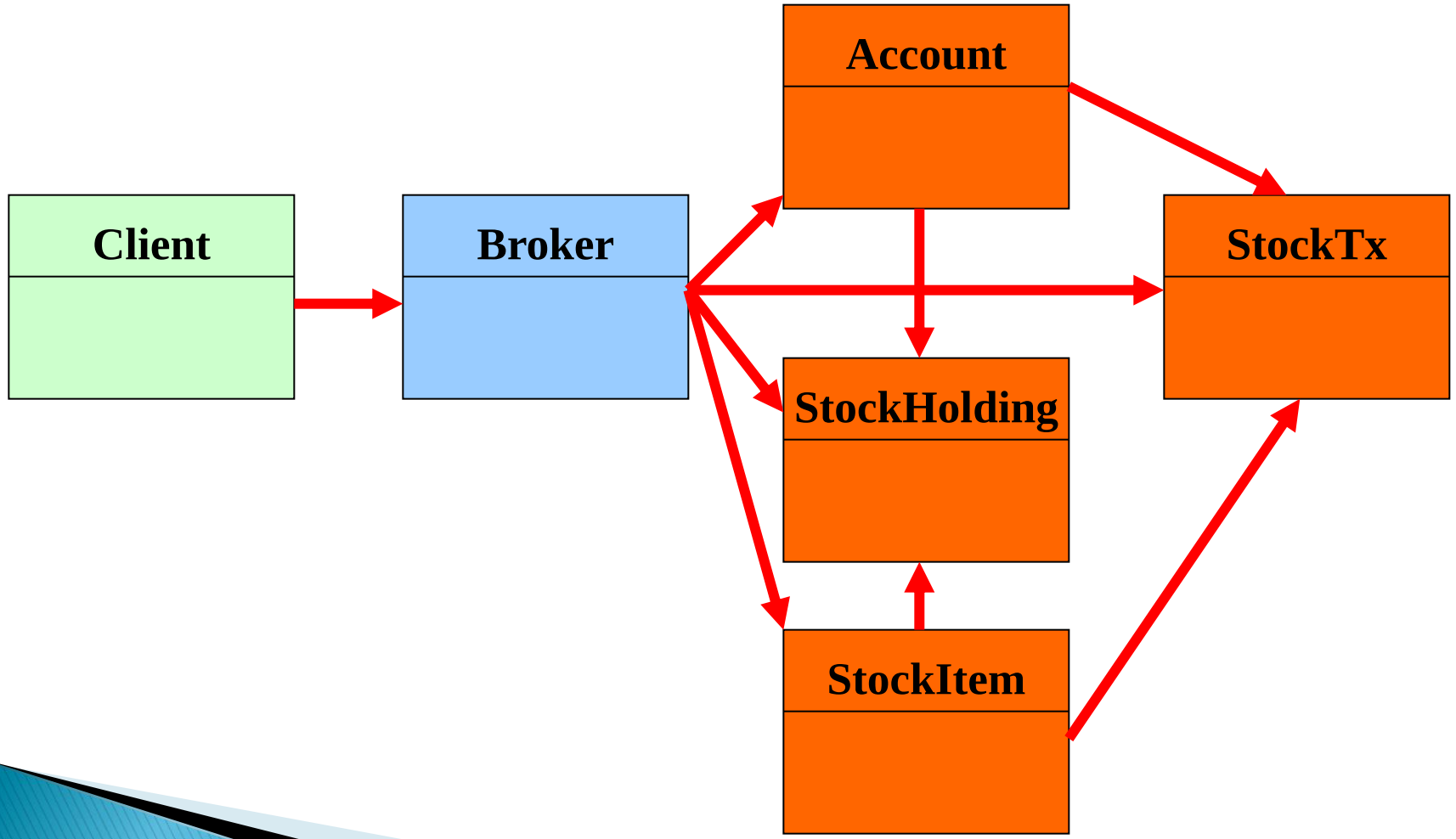
- You cannot use Reflection API to access information inaccessible to you.
- You cannot create a class loader or replace a security manager.
- You cannot set the socket factory used by ServerSocket or Socket
- You cannot use the object substitution features of the serialization protocol

What can't you do within an EJB? (cont'd)

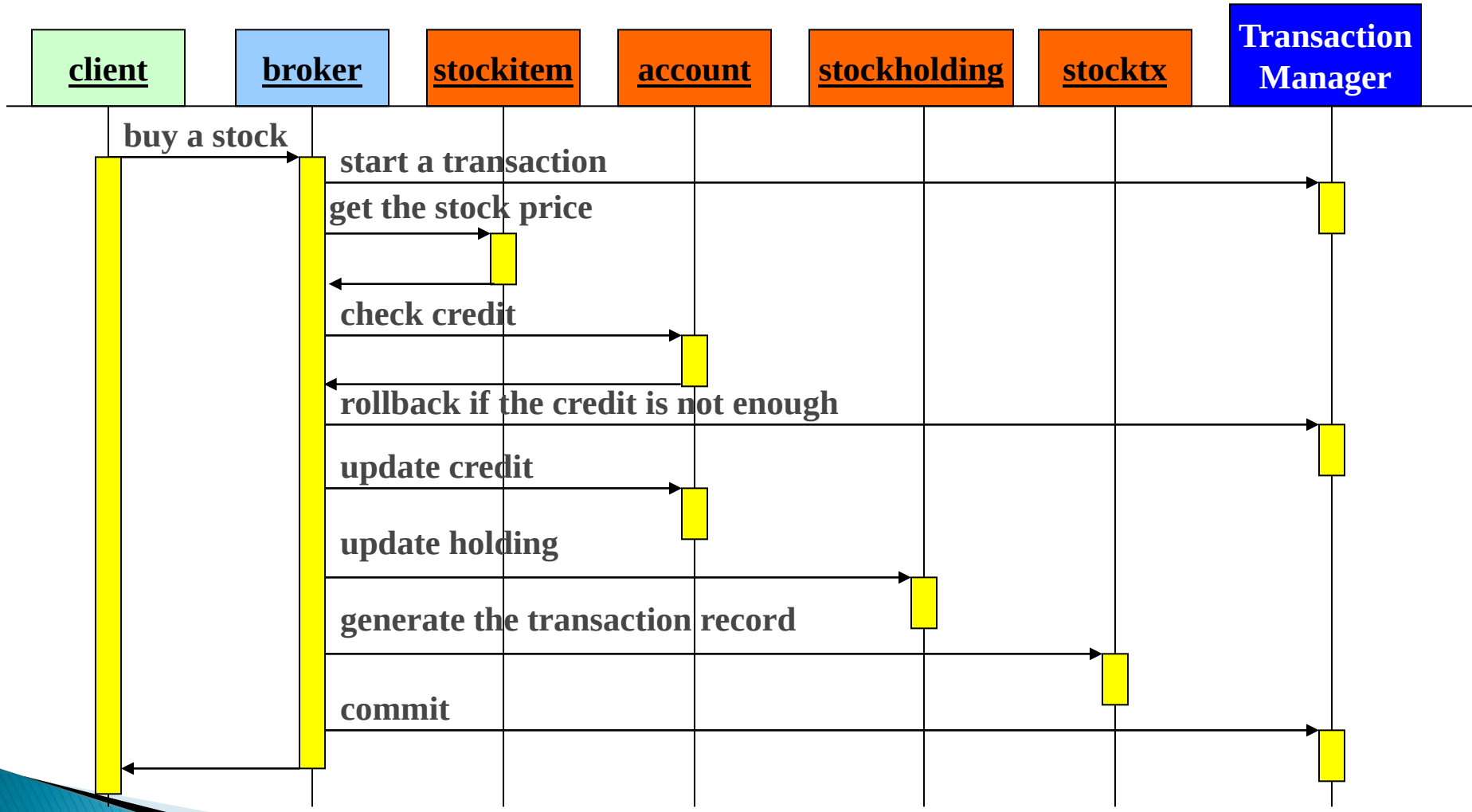
- use Threads or the Threading API
- use the AWT
- Act as a Network Server
- use Read/Write static fields
- use java.io package
- Load a native library (DLL etc)
- use "this" as an Argument or Return value

A CASE Study: Stockonline

Stockonline Design



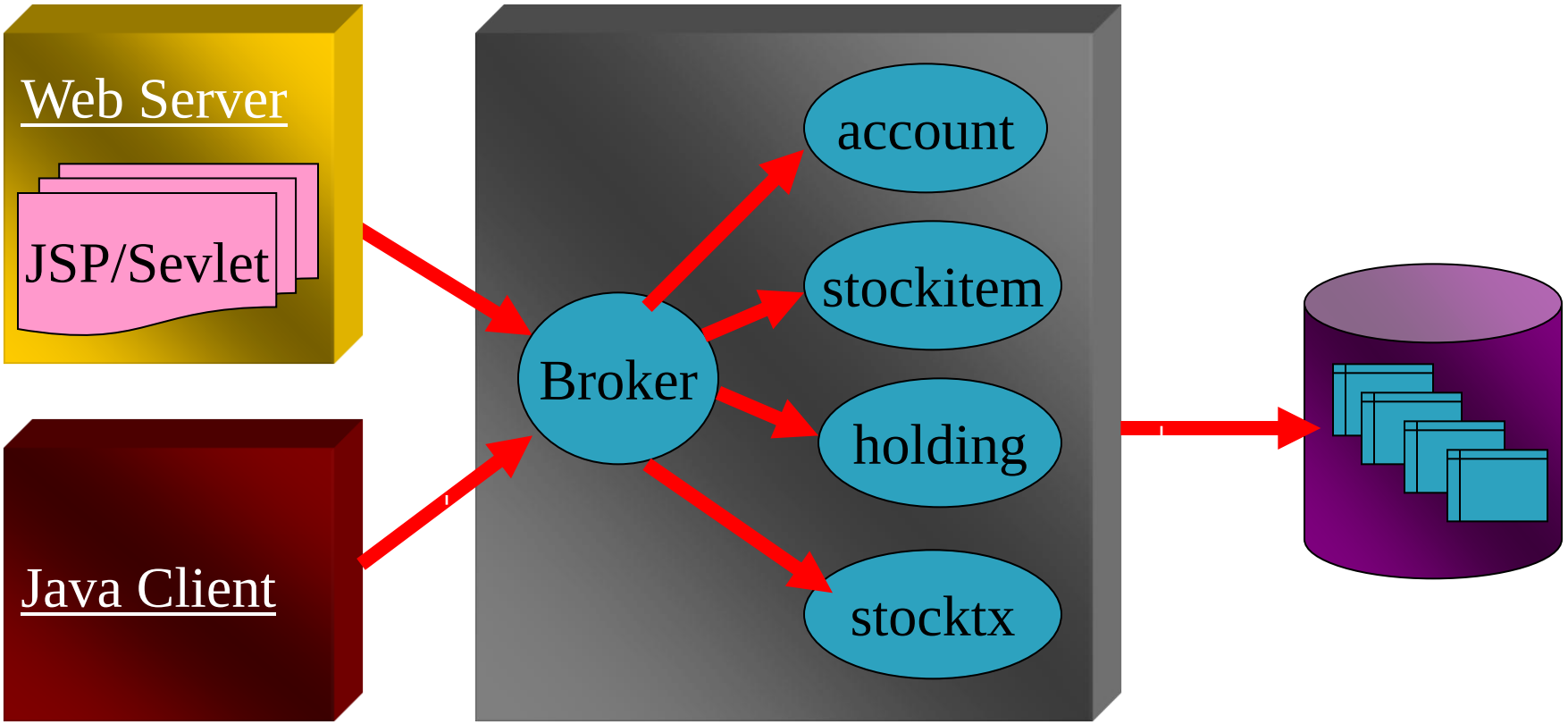
Stockonline Businss Logic



Tips for EJB Development & Deployment

- Development
  - Build skeletons of each component with zero business logics
  - Establish your own way/style to debug/monitor EJBs !?
  - Keep testing as you are developing
  - Keep making backup as you are developing
- Deployment
  - Be careful with vendor-specific deployment descriptor!
  - Be careful with datasource configuration! (vendor-specific)

Deployed Stockonline



Application Server

EJB Resources

- *Specifications:*
  - http://java.sun.com/products/ejb/docs.html
- *eBook:*
  - Mastering Enterprise JavaBeans 3.0, edition 4, by Rima Patel Striganesh *et al. 2006* *http://www.theserverside.com/tt/books/wiley/masteringEJB3/index.tss (It is free! but registration required)*
- *Online tutorials:*
  - http://java.sun.com/javaee/5/docs/tutorial/doc/

A Few Words for COM+

- COM+: An likewise (EJB) Microsoft component technology with:
  - **Some strengths**
    - Support Multiple languages (C#, VB, C/C++, J++?)
    - Good support for transaction
    - Very mature, robust, fast, scale etc.
    - Not updated as frequently as J2EE (a natural point!)
  - **And weakness:**
    - Window-only (no open specification, few open sources)
    - Basically equivalent for session beans only
    - A little complex for some non-Microsoft developers
    - Not part of .NET !

Summary of Part 2

- We learnt:
  - What is Component?
  - EJB as a Typical Component Technology
    - Session Bean (SB): Stateless vs. Stateful
    - Entity Bean (EB): BMB vs. CMB
    - Messaging-Driven Bean (MDB)
    - Time-S
    - Transaction: BMT and CMT
    - Timer Service Bean (TSB)
    - EJB Clients
    - JDBC
    - JNDI
    - A Case Study: Stockonline
  - Microsoft Corresponding Technology: COM+