What are business components

A business component is a reusable object that occupies middle-tier between user interface and the database

Business components
- represent persistent database objects
- implement business logic and business rules
- know how to access database
- can perform business transactions
- deployed usually in an application server
- aim at facilitating rapid development of server-side applications by connecting reusable components via their published interfaces
  - can be freely specialized and customized
Topics

- Enterprise JavaBeans (EJB)
  - EJB fundamentals
  - Entity Beans
  - Session Beans
- Business Components for Java (BC4J)
  - Creating entity components
  - Creating view components
  - Creating the application module

EJB vs BC4J - overview

- EJB represent the Java world → extends applications towards databases
  - Business logic needs to be provided to EJB components by the developer
  - Provides transaction, security and similar services, already present in databases
  - Components are called beans in EJB
    - Entity beans to contain data
    - Session beans to represent transactions, security, and similar functions
  - Relies on JavaBeans for GUI components
- BC4J represents the database world → extends databases towards applications
  - Derives business components from the underlying database design
  - Provides GUI components, already present in applications
- This genericity of EJB allows, for example, the deployment of a BC4J application in an EJB session bean
Objectives of EJB technology

- **Robustness**
  The architecture aims at providing reliable host services for the development of J2EE applications.

- **Scalability**
  The application server may be clustered to improve the scalability of hosted applications.

- **Support OO design principles**
  Enforcement of OO principles and design patterns is a necessary condition for EJB in order to support large, reusable and scalable systems.

- **Portability**
  This conforms to the Java ideology to “build once and use everywhere”.

- **Server properties**
  EJB needs to serve multiple client types, such as applets, standalone Swing GUI, web applications (JSP/Servlet), etc.

- **Single-threaded design with multithread features**
  Although a bean can be running on a single-thread it should demonstrate (simulate) a multi-threaded behavior to clients.

J2EE architecture

- Clients (HTML, Applets, Applications, etc)
- DBMS
- Message Queue
- JCA Connectors
- SessionBean
- EntityBean
- MessageBean
- Servlets, JSPs
- J2EE Server
- EJBs
- Web applications (web services, SOAP, etc)
- Other applications (legacy, ERP, etc)
Maciaszek, Liong - PSE

**EJB architecture from developer's perspective**

**Clients**
- applets, servlets, JSP, etc.

**EJB in J2EE Container**

- Remote
  - EJBObject
  - MyEJBInterface
  - EntityBean
  - SessionBean

- EnterpriseBean
  - MessageDrivenBean

- MyEJBHome
  - Implementation

- MyEJBObject
  - Implementation

**Clients**
- applets, servlets, JSP, etc.

**EJB in J2EE Container**

- Remote
  - EJBObject
  - MyEJBInterface
  - EntityBean
  - SessionBean

- EnterpriseBean
  - MessageDrivenBean

- MyEJBHome
  - Implementation

- MyEJBObject
  - Implementation

**MovieClient uses Movie EJB**

```java
8: public class MovieClient{
9:     public static void main(String args[]){
10:        MovieHome home = null;
11:        try {
12:            Context ctx = new InitialContext(System.getProperties());
13:            home = (MovieHome) PortableRemoteObject.narrow(
14:                ctx.lookup("MovieHome"), MovieHome.class);
15:            home.create("Neil Jordan", "Interview with the Vampire",
16:                "The vampire bla bla bla");
17:            //...
18:            Iterator i = home.findByDirector("Neil Jordan").iterator();
19:            System.out.println("These movies directed by Neil Jordan:");
20:            while (i.hasNext()) {
21:                Movie movie = (Movie) PortableRemoteObject.narrow(
22:                    i.next(), Movie.class);
23:                System.out.println(movie.getTitle());
24:            }
25:        }
26:    }
27: }
28: }
```

Client accessing EJB
**Entity beans**

- Represent **business objects**
- Are **persistent**
  - **Bean-Managed Persistent (BMP)** beans – persistence managed manually by the developer
  - **Container-Managed Persistent (CMP)** beans – persistence management left to the container
    - Easier to maintain
      - No SQL statements involved (EJB-Query Language (EJB-QL) is a simplified SQL variant for CMP if such a need arises)
      - No need for manual preparation of a database connection
    - Providing (mostly) abstract methods all of which will be implemented by the container
    - Container vendors provide different levels of services in managing persistent storage → dependence on vendors

---

**BMP bean - example**

```java
public class MovieBMPBean extends EntityBean, MovieInterface
...
public Collection ejbFindByDirector(String director)
throws FinderException {
    PreparedStatement pstmt = null;
    Connection conn = null;
    Vector v = new Vector();
    try {
        conn = getConnection();
        pstmt = conn.prepareStatement("select id from movie where director = ?");
        pstmt.setString(1, director);
        ResultSet rs = pstmt.executeQuery();
        while (rs.next()) {
            String id = rs.getString("id");
            v.addElement(new MoviePK(id));
        }
        return v;
    } catch (Exception e) {
        ...
    }
}
```

**MovieBMPBean for MovieActor application**
**CMP bean - example**

`public class MovieCMPBean extends EntityBean, MovieInterface {
// all methods set/get are implemented by container
public abstract void setDirector(String director);
public abstract String getDirector();
public String ejbCreate(String director, String title, String desc){
    setDirector(director); // set() methods rather storing to fields
    setTitle(title);
    setDescription(desc);
    return null; // CMP does not need the return value
}
// ... cut to save space
}`

---

**Session beans**

- represent business logic and business activities
- manipulate entity beans
- live only for the duration of the session – are not persistent
- two types based on their ability to maintain state between invocations
  - *stateless session bean* – serves requests but does not maintain any information state between each request
    - e.g. a web query to a search-engine
  - *stateful session bean* – has the capability to remember states between requests and therefore is able to enrich the interactions with clients
    - e.g. a shopping cart that maintains a list of items ordered throughout the shopping experience
Business Components for Java (BC4J)

- In Oracle’s BC4J, business components are known as entity objects.
- Unlike EJB, BC4J supports also view objects.
  - BC4J enables a quick generation of fully functional application modules complete with the presentation layer.
  - View object represents a visualization of execution of SQL query.
  - View link allows visualization of a master-detail relationship.
- IDEs, such as Oracle’s JDeveloper, provide design-time wizards, component editors and a programming environment to generate executable Java code and XML code for business components.
  - Support for routine create-read-update-delete (CRUD) operations on the content of business components.
  - Definitions of business components can be obtained from a schema for a database, from an eXtensible Markup Language (XML) schema, or similar sources of metadata information.

XML for entity components - example

```
<Entity>
  Name="Department"
  DBObjectType="table"
  DBObjectName="DEPARTMENT"
  RowClass="PSE2BusinessComponents.DepartmentImpl" >
  <Attribute>
    Name="DepartmentCode"
    IsNotNull="true"
    Precision="3"
    Type="java.lang.String"
    ColumnName="DEPARTMENT_CODE"
  </Attribute>
  <AccessorAttribute>
    Name="Employee"
    Association="PSE2BusinessComponents.FkEmployeeFkEmployeeAssoc"
  </AccessorAttribute>
  <Key>
    Name="PkDepartment"
    <AttrArray Name="Attributes">
      <Item Value="PSE2BusinessComponents.Department.DepartmentCode" />
    </AttrArray>
  </Key>
</Entity>
```
Java for entity components - example

```
package PSE2BusinessComponents;
import oracle.jbo.server.EntityImpl;

public class DepartmentImpl extends EntityImpl {
  
  public String getDepartmentName()
  {
    return (String)getAttributeInternal(DEPARTMENTNAME);
  }

  public void setDepartmentName(String value){
    setAttributeInternal(DEPARTMENTNAME, value);
  }

  public static Key createPrimaryKey(String departmentCode){
    return new Key(new Object[] {departmentCode});
  }

  public RowIterator getEmployee(){
    return (RowIterator)getAttributeInternal(EMPLOYEE);
  }

  ...}
```

XML for view components - example

```
<ViewObject
  Name="DepartmentView"
  SelectList="Department.DEPARTMENT_CODE, Department.DEPARTMENT_NAME"
  FromList="DEPARTMENT Department"
  ...}

  ComponentClass="PSE2BusinessComponents.DepartmentViewImpl"
  ...}

  <EntityUsage
    Name="Department"
    Entity="PSE2BusinessComponents.Department"
    ...}

  <ViewLinkAccessor
    Name="EmployeeView"
    ViewLink="PSE2BusinessComponents.FkEmployeeRefDepcodeLink"
    Type="oracle.jbo.RowIterator"
    IsUpdateable="false" >

  </ViewLinkAccessor>

</ViewObject>
```
Java for view components - example

DepartmentViewImpl.java for DepartmentView view object

package PSE2BusinessComponents;
import oracle.jbo.server.ViewObjectImpl;
public class DepartmentViewImpl extends ViewObjectImpl
{
    public DepartmentViewImpl()
    {
    }
}

Application module – simple view
Summary

- A **business component** is a reusable object that occupies middle-tier between user interface and the database.

- Two representative component technologies:
  - Enterprise JavaBeans (EJB), representing the Java world
  - Business Components for Java (BC4J), representing the database world

- **EJB** offers entity beans to contain data and session beans to represent transactions, security, and similar functions.

- In **BC4J** business components are known as entity objects; BC4J supports also view objects.
  - A BC4J application module for business components uses the definitions of entity and view objects in order to generate an application with default functionality.