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

- 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 a 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.
**EJB architecture from developer’s perspective**

- **Clients** (applets, servlets, JSP, etc)
- **EJB in JEE Container**
  - EJBObject
  - EJBHome
  - MyEJBObject
  - MyEJBHome
- **Bean-Managed Persistent (BMP) beans** – persistence managed manually by the developer
  - EJB architecture from developer’s perspective
- **Container-Managed Persistent (CMP) beans** – persistence management left to the container
  - EJB architecture from developer’s perspective
- **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-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
- **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
    - Stateful session bean – has the capability to remember states between requests and therefore is able to enrich the interactions with clients

---

**Client accessing EJB**

```java
public class MovieCMPBean extends EntityBean, MovieInterface {

    //... cut to save space

    public String ejbCreate(String director, String title, String desc) {
        return null; // CMP does not need the return value
    }
}
```

**BMP bean - example**

```java
public class MovieBMPBean for MovieActor application

public class MovieBMPBean extends EntityBean, MovieInterface {

    public Collection ejbFindByDirector(String director) throws FinderException {
        Vector v = new Vector();
        String sql = "select id from movie where director = ?";
        PreparedStatement pstmt = conn.prepareStatement(sql);
        pstmt.setString(1, director);
        ResultSet rs = pstmt.executeQuery();
        while (rs.next()) {
            v.addElement(new MoviePK(id));
        }
        return v;
    }
}
```

**CMP bean - example**

```java
public class MovieCMPBean for MovieActor application

public class MovieCMPBean extends EntityBean, MovieInterface {

    //... cut to save space

    public String ejbCreate(String director, String title, String desc) {
        return null; // CMP does not need the return value
    }
}
```
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

Department.xml (excerpt) for Department entity object

```
<entity>
  <name>Department</name>
  <dbobject type="table">
    <dbobjectname>DEPARTMENT</dbobjectname>
    <attributes>
      <attribute name="DepartmentCode" isnotnull="true" precision="3" type="java.lang.String">
        <columnname>DEPARTMENT_CODE</columnname>
      </attribute>
      <attribute name="DepartmentName">
        <columnname>DEPARTMENT_NAME</columnname>
      </attribute>
      <attribute name="ManagerName">
        <columnname>MANAGER_NAME</columnname>
      </attribute>
      <attribute name="DepartmentAddress">
        <columnname>DEPARTMENT_ADDRESS</columnname>
      </attribute>
      <attribute name="DepartmentCity">
        <columnname>DEPARTMENT_CITY</columnname>
      </attribute>
      <attribute name="DepartmentState">
        <columnname>DEPARTMENT_STATE</columnname>
      </attribute>
      <attribute name="DepartmentZip">
        <columnname>DEPARTMENT_ZIP</columnname>
      </attribute>
      <attribute name="DepartmentPhone">
        <columnname>DEPARTMENT_PHONE</columnname>
      </attribute>
      <attribute name="DepartmentFax">
        <columnname>DEPARTMENT_FAX</columnname>
      </attribute>
      <attribute name="DepartmentEmail">
        <columnname>DEPARTMENT_EMAIL</columnname>
      </attribute>
      <attribute name="DepartmentWeb">
        <columnname>DEPARTMENT_WEB</columnname>
      </attribute>
    </attributes>
  </dbobject>
  <rowclass>DepartmentImpl</rowclass>
</entity>
```

Java for entity components - example

DepartmentImpl.java (excerpt) for Department entity object

```
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 RowIterator getEmployee() {
    return (RowIterator)getAttributeInternal(EMPLOYEE);
  }
  public static Key createPrimaryKey(String departmentCode){
    return new Key(new Object[] {departmentCode});
  }
}
```

XML for view components - example

DepartmentView.xml (excerpt) for DepartmentView view object

```
&viewobject
  <name>DepartmentView</name>
  <selectlist>Department.DEPARTMENT_CODE, Department.DEPARTMENT_NAME</selectlist>
  <fromlist>DEPARTMENT Department</fromlist>
  ... 
  <entityusage name="Department" entity="PSE2BusinessComponents.Department" />
  <viewattribute name="DepartmentCode" isnotnull="true" entityattrname="DepartmentCode" />
  <viewlink accessor="PSE2BusinessComponents.FkEmploeeRefDepcodeLink" type="oracle.jbo.RowIterator" isupdateable="false" />
</viewobject>
```

Java for view components - example

DepartmentViewImpl.java for DepartmentView view object

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

public class DepartmentViewImpl extends ViewObjectImpl {
  ... 
}
```

Application module – simple view

[Image of an application module example]

© Pearson Education 2005
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