E14

JDBC Access to IMS DB from DB2, CICS, and WebSphere

Kyle Charlet
charletk@us.ibm.com

St. Louis, MO **Sept. 30 - Oct. 3, 2002**

RETURN TO INDEX

Overview

- **IMS Java**
  - What Is IMS Java
  - Why Use IMS Java
  - The IMS Java Class Library Architecture
- **JDBC and J2EE**
- **Dealership Sample Application**
  - Font-end/Back-end split
- **Environments**
  - Non-Managed
    - CICS
    - DB2
  - Managed
    - WebSphere

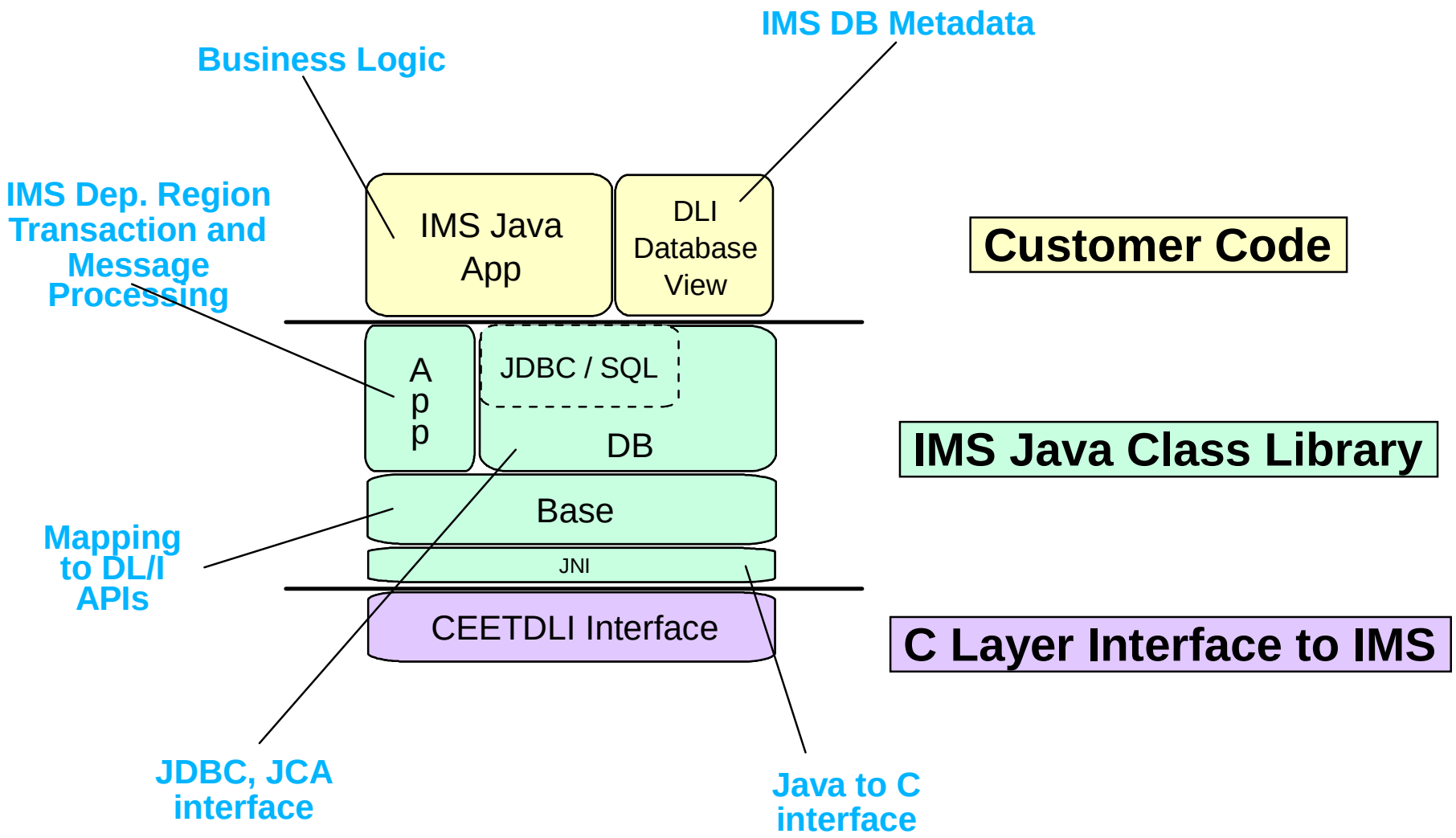
What is IMS Java?

- **A new feature in IMS v7**
- **A set of classes that...**
  - Offers Java support to access IMS Databases from various environments (IMS, CICS, DB2, WebSphere)
  - Enables SQL access through the JDBC interface
- **Java Virtual Machine (JVM) support in dependent regions**
  - JDK 1.3 support
  - JDBC 2.1 support
  - Just-In-Time (JIT) compilation
  - Resetable JVM
- **High Performance Java (HPJ) compiled**
  - Runs as a Language Environment run unit on IMS v7
  - Not supported on IMS v8 (including prior v7 HPJ applications)
  - Frozen at JDK 1.1.8
  - Frozen at JDBC 1.0

Why IMS Java?

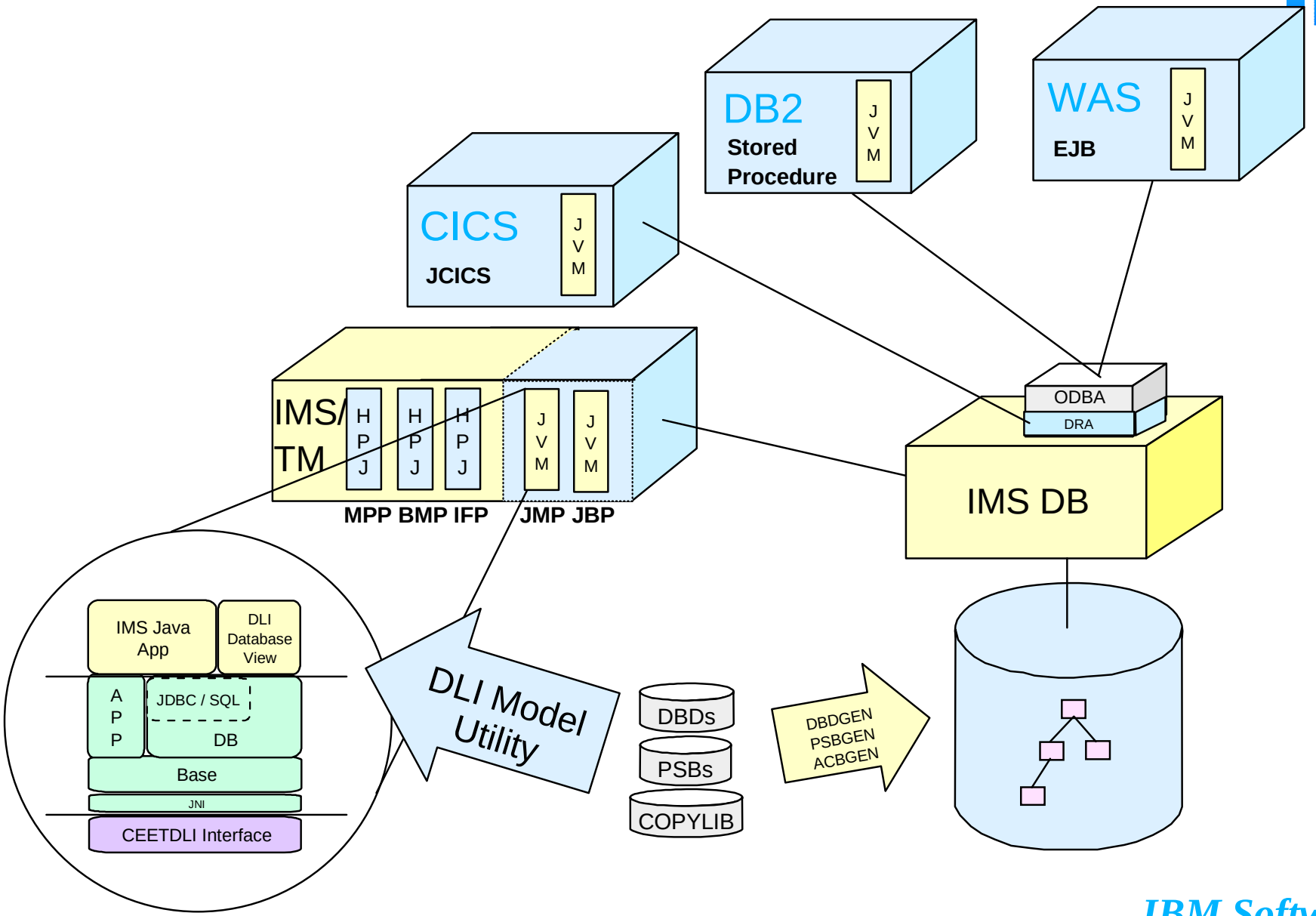
- **Colleges teach Java, very few still teach COBOL**
- **Colleges teach relational DBs with SQL access, very few teach hierarchical with SSA access**
- **JDBC is an industry standard**
  - Minimizes specific backend DB knowledge of IMS
- **Customer requests for Java support**
- **Rapid application development**

Java Class Library



IBM Software

IMS Java - The Big Picture



IBM Software

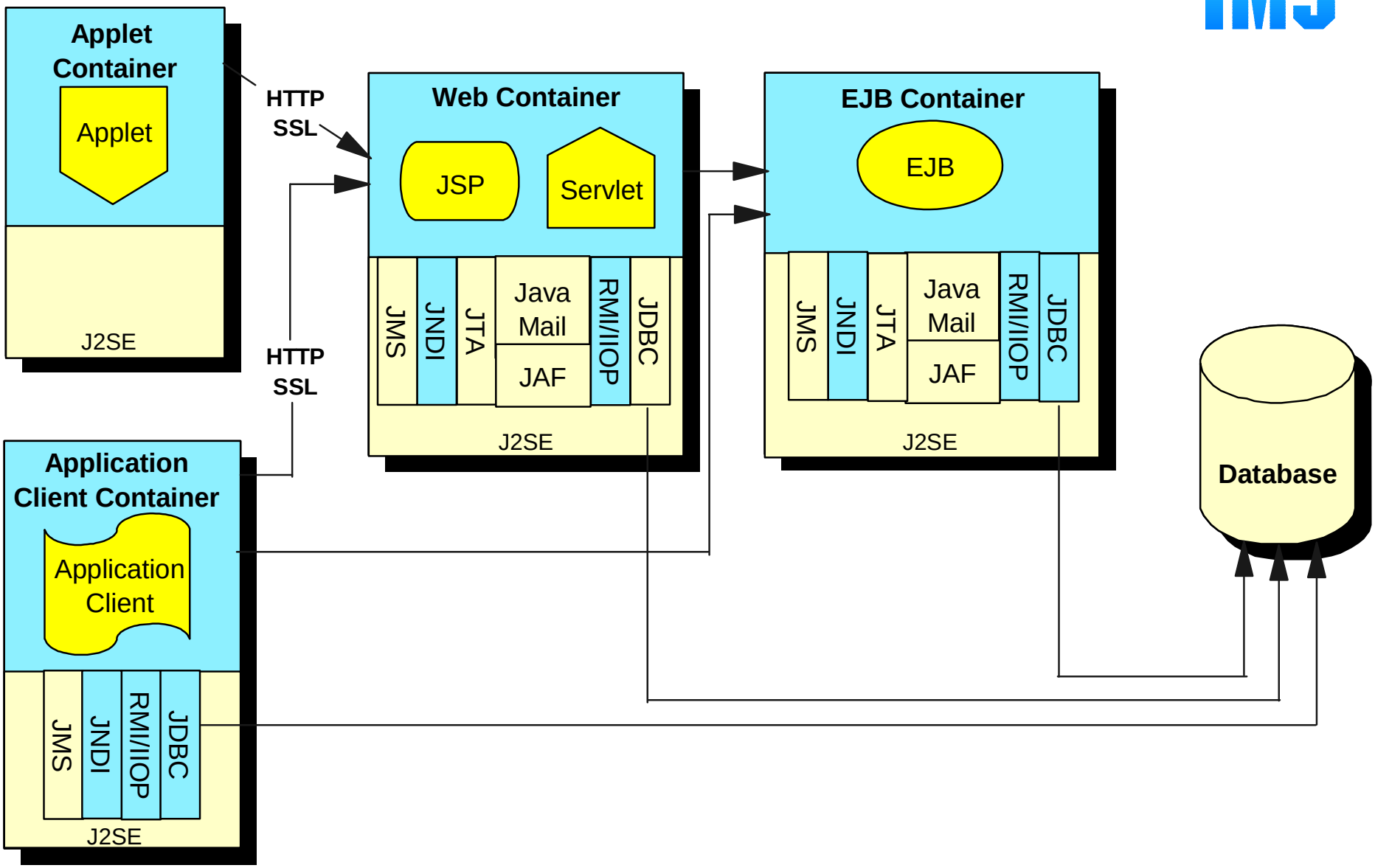
Overview

- **IMS Java**
  - What Is IMS Java
  - Why Use IMS Java
  - The IMS Java Class Library Architecture
- **JDBC and J2EE**
- **Dealership Sample Application**
  - Font-end/Back-end split
- **Environments**
  - Non-Managed
    - CICS
    - DB2
  - Managed
    - WebSphere

JDBC and J2EE Evolution

- **Standard way to Query Database (relational)**
  - Structured Query Language (SQL) - 1992
- **Communicating Query to Database**
  - Open Database Connectivity (ODBC) - C based
- **Standard API to Query Database**
  - "Java Database Connectivity" (JDBC) - Platform/DB Independent
- **Standard API to Establish Connection**
  - J2EE Connection Architecture (JCA)
- **Standard API to Build Enterprise Applications**
  - Java 2 Enterprise Edition (J2EE)

J2EE Architecture

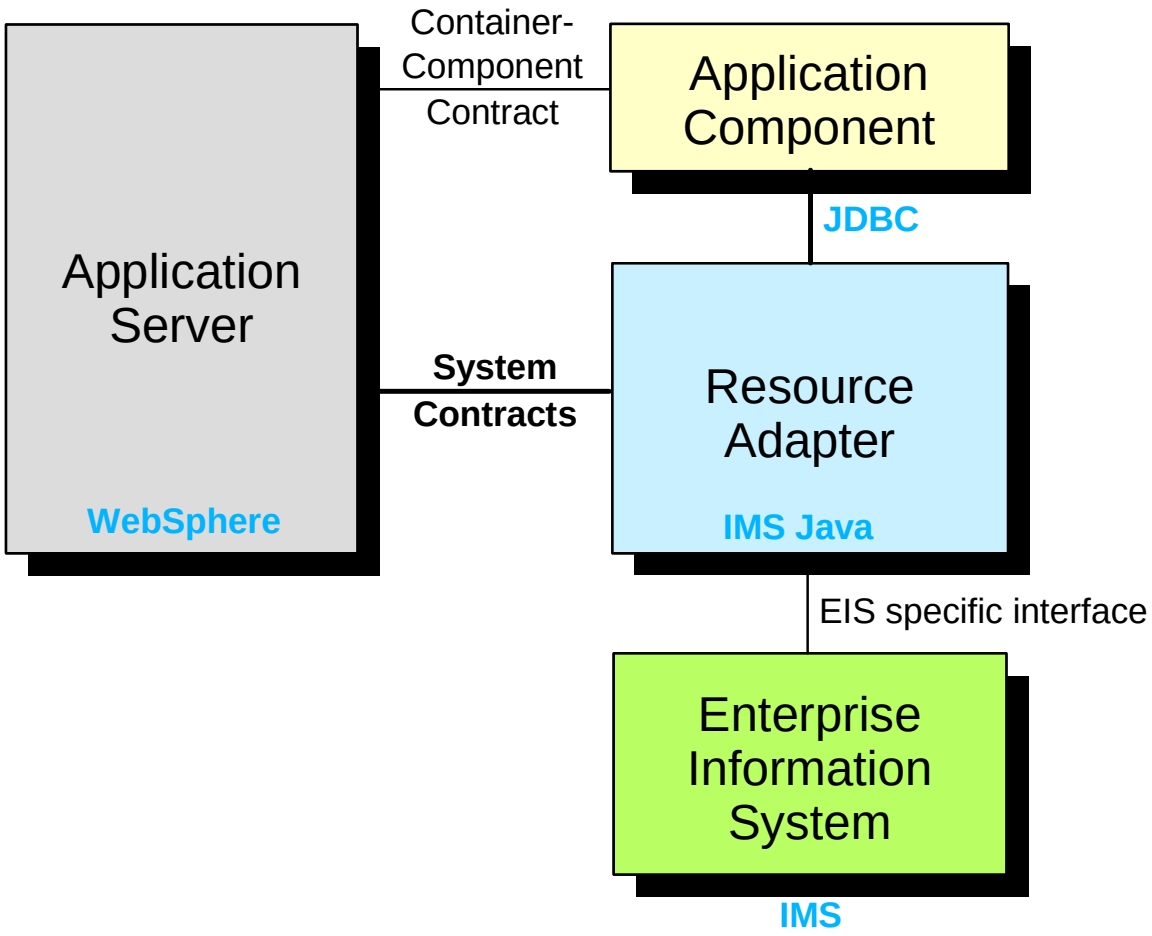


IBM Software

J2EE Connection Architecture

- **Proposed Java standard architecture for deploying a resource adapter in a J2EE compliant application server**
- **Defines contracts between...**
  - resource adapter and the application component
  - resource adapter and the application server

J2EE Connection Architecture

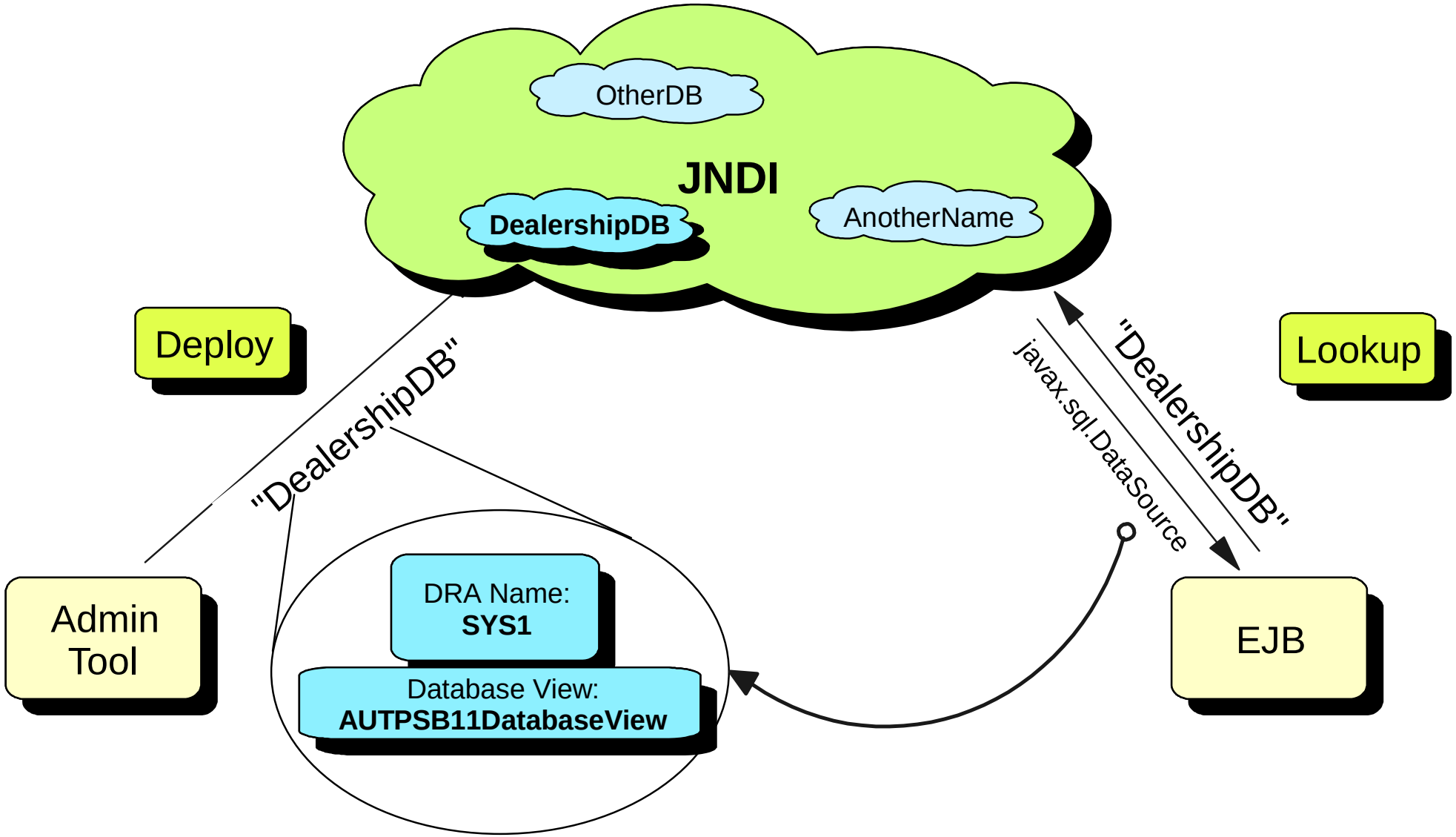


IBM Software

DataSource

- **Factory for connections to a physical data source**
- **Replacement to the DriverManager facility**
  - Required when running in a managed environment (WebSphere)
- **Typically registered with a naming service based on the Java™ Naming and Directory (JNDI) API.**
- **DataSource objects have properties that can be modified when necessary**
  - Code accessing the data source does not need to be changed

DataSource and JNDI

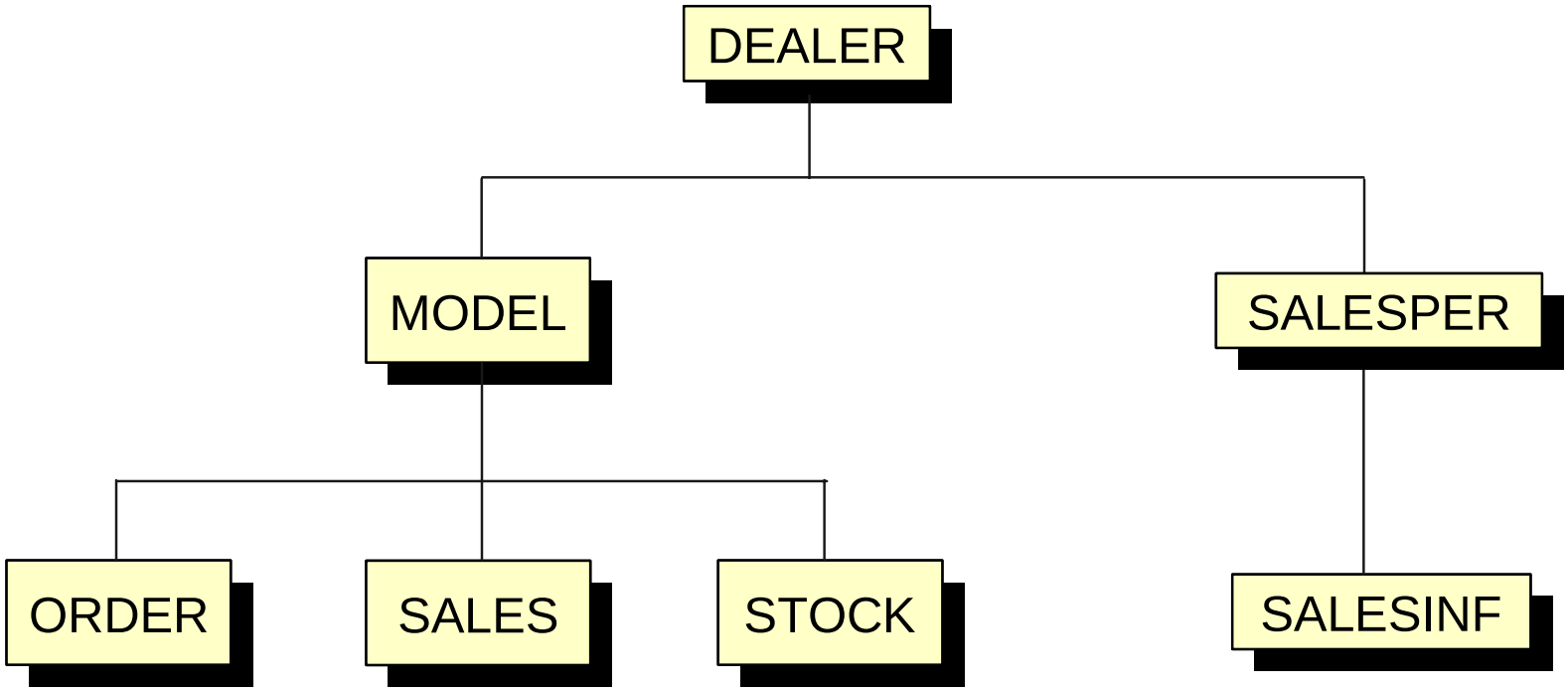


IBM Software

Overview

- **IMS Java**
  - What Is IMS Java
  - Why Use IMS Java
  - The IMS Java Class Library Architecture
- **JDBC and J2EE**
- **Dealership Sample Application**
  - Font-end/Back-end split
- **Environments**
  - Non-Managed
    - CICS
    - DB2
  - Managed
    - WebSphere

Dealership Sample Database



IBM Software

Dealership Sample Application

- **Performs specific queries to the database**
  - For example:
    - list all models
    - list details of a particular model
- **Split into a front-end and a back-end**
- **Front-End**
  - Environment specific
    - Process message queue (IMS)
    - Invoke stored procedure (DB2)
    - JCICS application (CICS)
    - Enterprise Java Bean (WebSphere)
- **Back-End**
  - Performs all query processing
  - Sends data back to the caller (front-end)
  - Environment neutral

Dealership Sample (Back-End)


```
public class AutoDealership {

   /** The database connection is created by each front-end and given
       to the single back-end **/
   public AutoDealership(Connection connection) {
     this.connection = connection;
   }

   public ListModelOutput[] listModels() throws SQLException {

       SQL processing logic here...

   }

   public FindCarOutput[] findCar(FindCarInput input) throws
           SQLException {

       SQL processing logic here...

   }
 }
```

IBM Software

Dealership Sample (Back-End)


```
public ListModelOutput[] listModels() throws SQLException {

  // Create the SQL statement - no inputs needed
  String query = "SELECT * FROM Order.ModelSegment";

  // Execute the query
  Statement statement = connection.createStatement();
  ResultSet results = statement.executeQuery(query);

  process results...

}
```

Dealership Sample (Back-End)

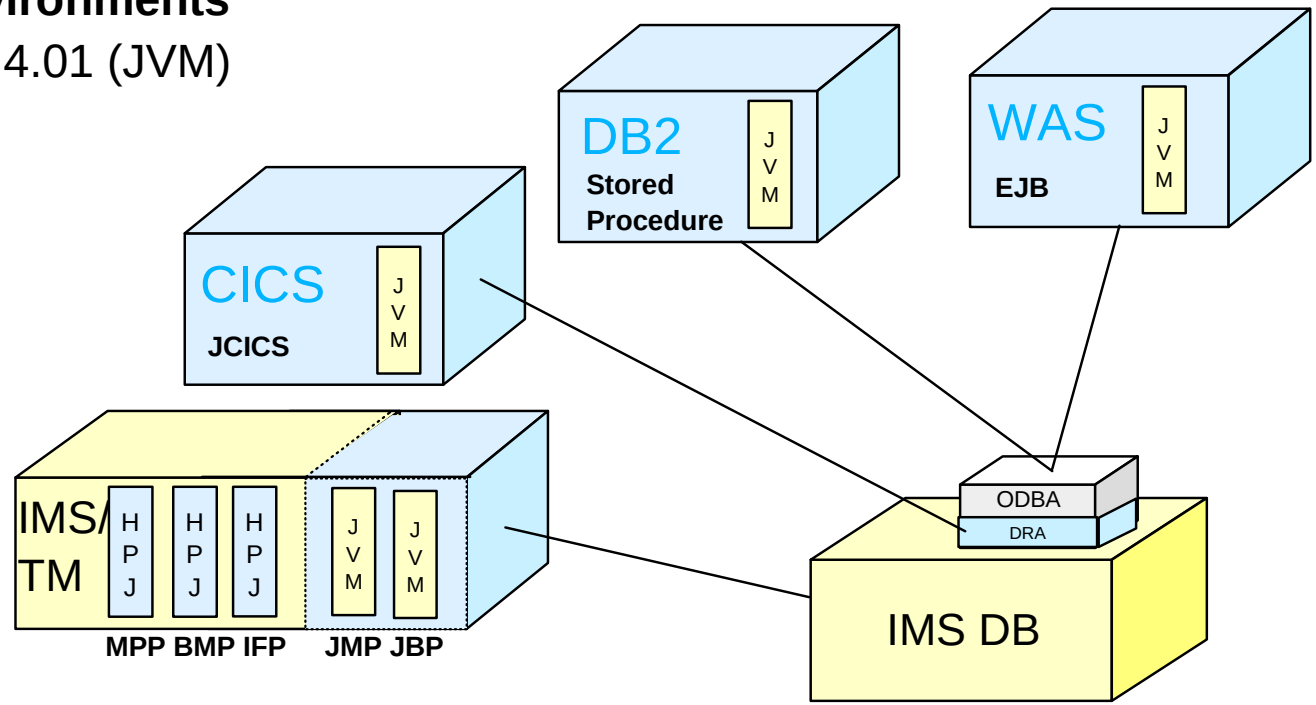

```
public Vector listModels() throws SQLException {

  create statement and execute query...

  Vector models = new Vector();
  ListModelOutput output = null;
  while (results.next()) {
    output = new ListModelOutput();
    output.setModelType(results.getString("ModelType"));
    output.setMake(results.getString("Make"));
    output.setModel(results.getString("Model"));
    output.setYear(results.getString("Year"));

    models.addElement(output);
  }

  return models;

}
```

Overview

- **IMS Java**
  - What Is IMS Java
  - Why Use IMS Java
  - The IMS Java Class Library Architecture
- **JDBC and J2EE**
- **Dealership Sample Application**
  - Font-end/Back-end split
- **Environments**
  - Non-Managed
    - CICS
    - DB2
  - Managed
    - WebSphere

Running Environments

- **Non-Managed Environments**
  - IMS TM 7.1 (Resetable JVM)
  - IMS TM 8.1 (Resetable JVM)
  - CICS 2.1 (Resetable JVM)
  - DB2 V7 (Resetable JVM)
- **Managed Environments**
  - WebSphere 4.01 (JVM)



IBM Software

Non-Managed Connection Establishment

- **Non-Managed Environment (IMS, DB2, CICS)**
  - An application can construct a DataSource prior to its use
    - Alternatively, a DataSource can be bound to the namespace
  - Application acquires a Connection from the DataSource

```
IMSJdbcManagedConnectionFactory mcf = new IMSJdbcManagedConnectionFactory();

mcf.setDRAName("SYS1");
mcf.setDatabaseViewName("samples.dealership.AUTPSB11DatabaseView");

DataSource dataSource = (DataSource)mcf.createConnectionFactory();

Connection connection = dataSource.getConnection();
```

IBM Software

CICS Java Access to IMS Data

- **Run in JVM**
  - CICS TS 2.1
  - JCICS API
    - Java version of the CICS API
  - JDK 1.3
  - JDBC 2.1
- **Can only connect to (allocate) one PSB at a time**
  - Only one Connection active at a time in an application
- **Synchpoint done at deallocate PSB**

CICS Java Access to IMS Data


```
// Import the JCICS package
import com.ibm.cics.server.*;

public class CICSAuto {

  private static Task task = Task.getTask();

  /* Invoked when the CICS transaction corresponding to this class
     is executed. */
  public static void main(CommAreaHolder cah) {
      CICSAuto application = new CICSAuto();
      application.listModels();
  }

  /* The listModels method provides a collection of all automobile
     models in the database */
  public void listModels() {

    method logic here...

  }
}
```

IBM Software

CICS Java Access to IMS Data


```
/* The listModels method provides a collection of all automobile
   models in the database */
public void listModels() {

  // Get connection
  IMSJdbcManagedConnectionFactory mcf = new IMSJdbcManagedConnectionFactory();
   mcf.setDatabaseViewName("samples.dealership.AUTPSB11DatabaseView");
   DataSource dataSource = (DataSource)mcf.createConnectionFactory();
   Connection connection = dataSource.getConnection();

   // Pass connection to back-end
   AutoDealership autoDealership = new AutoDealership(connection);

   // Call listModels method and get results in 'output' Object
   Vector output = autoDealership.listModels();

   display results...

}
```

DB2 Stored Procedure Support

- **Run in JVM**
  - DB2 Version 7
  - APAR PQ46673 (resetable JVM)
- **Stored Procedures that access IMS Databases**
  - User-written structured query language (SQL) programs that are stored at the DB2 server and can be invoked by a client application
- **DB2 handles synchpoint (not stored procedure)**
- **DRA table and name required**

IBM Software

DB2 Stored Procedure Support


```
public class DB2Auto() {

  public static void listModels(String[] make, String[] model) {

     // Get connection
     IMSJdbcManagedConnectionFactory mcf = new IMSJdbcManagedConnectionFactory();
     mcf.setDRAName("SYS1");
     mcf.setDatabaseViewName("samples.dealership.AUTPSB11DatabaseView");
     DataSource dataSource = (DataSource)mcf.createConnectionFactory();
     Connection connection = dataSource.getConnection();

     // Pass connection to back-end
     AutoDealership autoDealership = new AutoDealership(connection);

     Vector output = autoDealership.listModels();

     // Return data to client
     make[0] = (String)output.elementAt(0);
     model[0] = (String)output.elementAt(1);
   }

}
```

IBM Software

WebSphere 4.01 Support

- **Runs in JVM**
  - JDK 1.3
  - JDBC 2.1
- **Applications run as Enterprise Java Beans (EJBs)**
- **J2EE Connection Architecture**
  - Managed Connection
- **Access through ODBA/DRA**

Managed Connection Establishment

- **Managed Environment (WebSphere)**
  - DataSource deployed in JNDI namespace using the WebSphere Application Server for z/OS and OS/390 Administration tool
  - Application (EJB) makes a request for the DataSource and acquires a Connection from it

```
Context ctx = new InitialContext();

DataSource dataSource = (DataSource)ctx.lookup("DealershipDB");
Connection con = dataSource.getConnection();
```

IBM Software

J2EE Resource Adapter

- **Resource Adapter Archive**
  - Obtain Resource Adapter Archive (.rar)
  - Deploy Resource Adapter Archive
- **DataSource**
  - Deploy Resource Adapter Instance
- **Enterprise Archive**
  - Build Enterprise Archive (.ear)
  - Deploy Enterprise Archive

IMS Java Resource Adapter

- **We provide a J2EE Resource Adapter ARchive (RAR) file containing:**
  - Deployment descriptor (ra.xml)
  - IMS Custom Service (IMSJdbcCustomService.xml)
    - Handles initialization and termination of IMS
  - Latest information associated with installing the IMS JDBC resource adapter (howto.html)

- **WebSphere will provide tooling to deploy RAR file into server**
  - Add IMS Custom Service
  - Set CLASSPATH and LIBPATH
  - Understand how to deploy an instance of an IMS Java Resource Adapter (via ra.xml)

DataSource Deployment

- **Use WebSphere Application Server for z/OS and OS/390 Administration tool**
  - Shipped with WebSphere on OS/390
- **Create J2EE Resource Instance**
  - Specify IMSJdbcDataSource as the J2EE Resource Type
- **Configure**
  - DRA Startup Table
  - Generated DLIDatabaseView
- **Deploy**

Enterprise Archive

- **J2EE Enterprise Application Archive**
  - Complete J2EE application
- **Must contain**
  - One or more J2EE modules (Java Archive)
  - Deployment descriptor (application.xml)
    - Represents a top level view of a J2EE application's contents
- **May contain**
  - One or more Web modules (Web Archive)
  - Libraries referenced by J2EE modules

Enterprise Archive (Servlet)

- **Looks up the EJB home interface in JNDI**
- **Using the home interface, creates the EJB remote interface**
- **Invokes methods on the remote interface (e.g. listModels)**
  - Remote interface uses IIOP to communiate to EJB
    - Pass-through interface
- **Passes results of method to a Java Server Page (JSP) for displaying on a web browser**

IBM Software

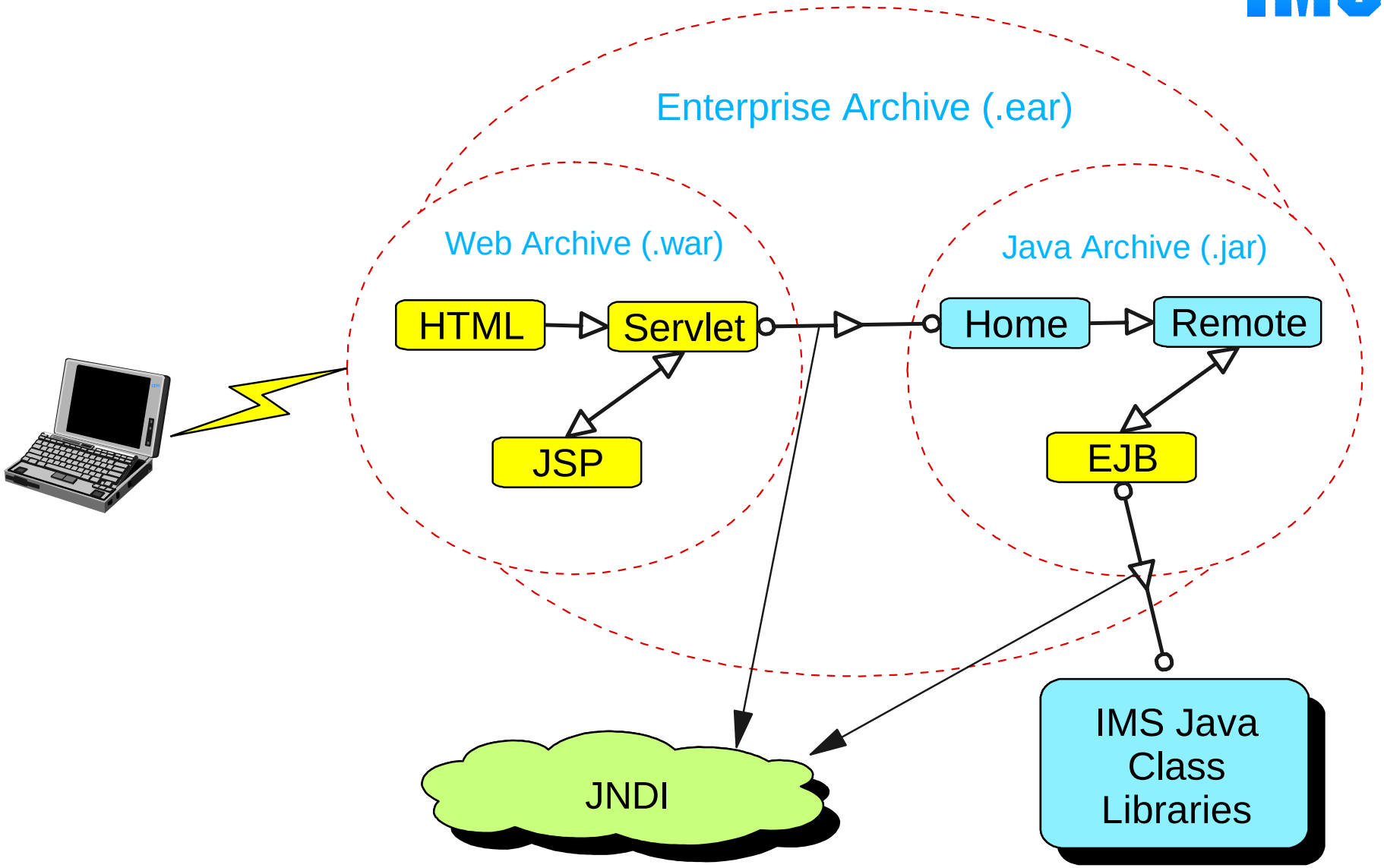
Enterprise Archive (EJB)

- Receives requests from the servlet
- Obtains a connection
- Forwards requests to back-end dealership application
- Sends the results back to the servlet to display

Enterprise Archive (EJB)

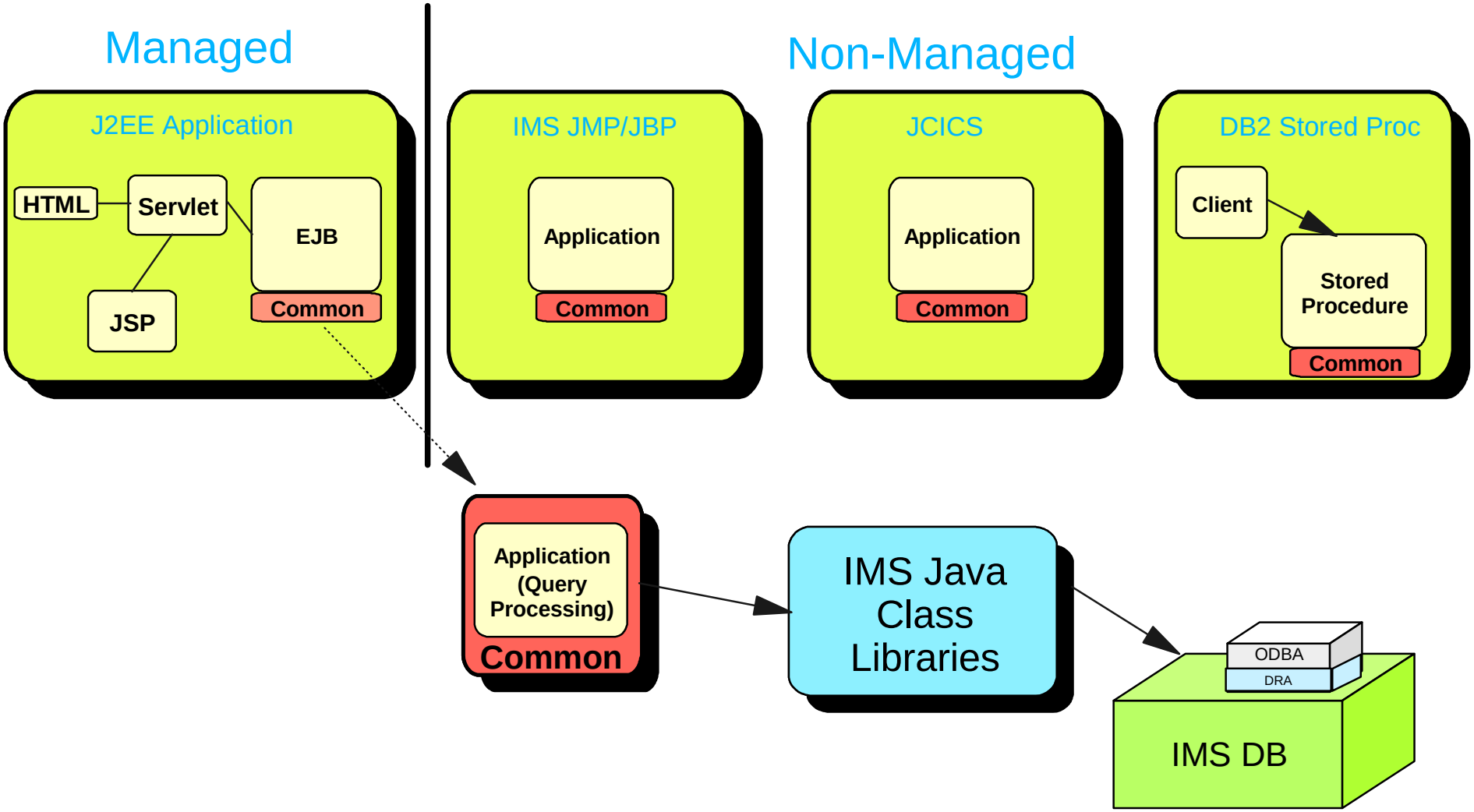

```
public Vector listModels() throws javax.ejb.EJBException {

  InitialContext initialContext = new InitialContext();
  // perform JNDI lookup to obtain the DataSource and get the Connection
  DataSource dataSource = (DataSource)initialContext.lookup("DealershipDB");
  Connection connection = dataSource.getConnection();

  AutoDealership dealer = new AutoDealership(connection);
  Vector output = dealer.listModels();

  return output;
}
```

Enterprise Archive



IBM Software

Conclusion



IBM Software