J2EE Connection Architecture

Dev Bhattacharyya
devb@ieee.org

J2EE Container Limitations

- The Container for various reasons cannot communicate with the external world directly.
- The Container uses App Server Extensions to communicate with external world.
- JDBC, JavaMail, JNDI, JMS and J2EE CA are some of the architectures

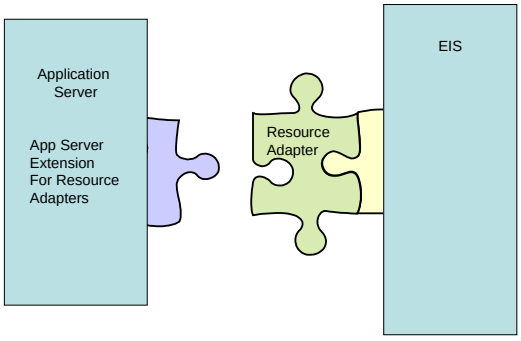
J2EE Connection Architecture

- Uniform Way to integrate J2EE Application Servers with EIS
- Facilitate Sharing of Data
- Integrate new J2EE applications with legacy
- J2EE Connection Architecture is mentioned throughout, the Resource Adapter is what this presentation concentrates on

J2EE CA Components

- Resource Adapters
- Data Mapping
- Message Brokers
- Workflow

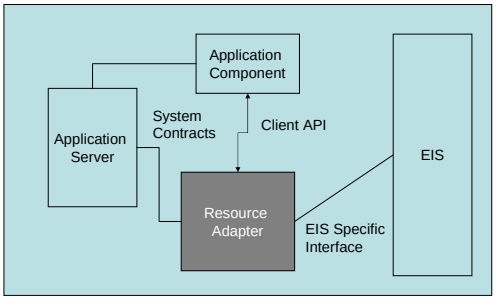
Integrating the Application with EIS



Resource Adapters

- Connecting the Container to the external world

J2EECA Components and Interactions



J2EE CA RA 1.0

- Connection Management
- Transaction Management
- Security

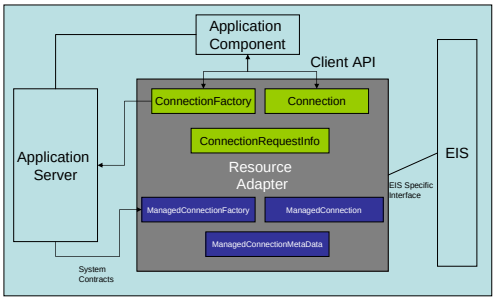
J2EE CA RA 1.5

- Improved Outbound
- New Inbound Message and Transactions
- Lifecycle and Workmanagement Contracts
- Support for Administered Objects

Java Contracts and Interfaces

- The J2EE Connector Architecture specification defines interfaces, which implement the three contracts – Connection, Transaction and Security
- Most of these interfaces are mandatory.
- Vendors can define and implement their own interfaces

Connection Management



Connection Factory

- ConnectionFactory is an interface that allows an application component to get a connection to an EIS instance.
- An application establishes a connection through the getConnection method.
- Then, this method must ask the application server to allocate a connection through the server's ConnectionManager. allocateConnection method.

ConnectionFactory (Sample Code)

```
public class StarJCAConnectionFactory implements
javax.resource.cci.ConnectionFactory {

ManagedConnectionFactory fact = null;
ConnectionManager cxManager = null;

public StarJCAConnectionFactory(ManagedConnectionFactory fact,
   ConnectionManager cxManager) {
System.out.println("In StarJCAConnectionFactory (constructor)");
this.fact = fact;
if (cxManager == null)
this.cxManager = new StarJCAConnectionManager();
else
 this.cxManager = cxManager;
}
…
…
```

Connection Factory Code Contd

```
…
…

public Connection getConnection() throws ResourceException {
System.out.println("In
   StarJCAConnectionFactory.getConnection,1");
try {
StarJCAConnection sconn =
   (StarJCAConnection)cxManager.allocateConnection(fact, null);
return (javax.resource.cci.Connection)sconn;
} catch (ResourceException ex) {
throw new ResourceException(ex.getMessage());
}
}
```

Connection

- The Connection interface provides an application with connectivity to an EIS. A close method must be provided so the application component can terminate the connection to the EIS.

Connection (Sample Code)

```
public class StarJCAConnection implements
   javax.resource.cci.Connection {

private StarJCAManagedConnection mc;

public StarJCAConnection(StarJCAManagedConnection mc) {
System.out.println("In StarJCAConnection");
this.mc = mc;
}

/* (non-Javadoc)
 * @see javax.resource.cci.Connection#close()
 */
public void close() throws ResourceException {
}
…
…
```

Connection Request Info

- ConnectionRequestInfo
- ConnectionRequestInfo represents a resource adapter's request-specific data. It is passed to the application server's ConnectionManager. allocateConnection .
- The value null can be used if there is no data to pass.
- If a resource adapter chooses to implement this interface, then it must provide the equals and hashcode methods to aide the application server in connection pooling.

Connection Request Info

```
public class StarJCAConnectionRequestInfo implements ConnectionRequestInfo {
private String user;
private String password;
public StarJCAConnectionRequestInfo(String user, String password) {
System.out.println("In StarJCAConnectionRequestInfo");
this.user = user;
this.password = password;
}
public String getUser() {
System.out.println("In StarJCAConnectionRequestInfo.getUser");
return user;
}
public String getPassword() {
System.out.println("In StarJCAConnectionRequestInfo.getPassword");
return password;
}
public boolean equals(Object obj) {
System.out.println("In StarJCAConnectionRequestInfo.equals");
if (obj == null)
return false;
if (obj instanceof StarJCAConnectionRequestInfo) {
StarJCAConnectionRequestInfo other = (StarJCAConnectionRequestInfo) obj;
return isEqual(user, other.user)
&& isEqual(password, other.password);
} else {
return false;
}
}
```

Connection Request Info

```
public int hashCode() {
System.out.println("In StarJCAConnectionRequestInfo.hashCode");
String result = "" + user + password;
return result.hashCode();
}

private boolean isEqual(Object o1, Object o2) {
System.out.println("In StarJCAConnectionRequestInfo.isEqual");
if (o1 == null)
return o2 == null;
else
return o1.equals(o2);
}

}
```

Managed Connection Factory

- ManagedConnectionFactory
- The interface ManagedConnectionFactory either matches an existing connection to the EIS with the incoming request or creates a new physical connection to the EIS.
- When the application server needs to allocate a connection to the EIS, it asks the resource adapter's ManagedConnectionFactory instance to get either an existing or a new connection.
- The configuration of the instance is facilitated by request-specific data.

Managed Connection Factory

```
public class StarJCAManagedConnectionFactory
implements ManagedConnectionFactory, Serializable {
public StarJCAManagedConnectionFactory() {
System.out.println("In StarJCAManagedConnectionFactory.constructor");
}
public Object createConnectionFactory(ConnectionManager cxManager)
throws ResourceException {
System.out.println(
"In StarJCAManagedConnectionFactory. createConnectionFactory,1");
return new StarJCAConnectionFactory(this, cxManager);
}
public Object createConnectionFactory() throws ResourceException {
System.out.println(
"In StarJCAManagedConnectionFactory.createManagedFactory,2");
return new StarJCAConnectionFactory(this, null);
}
public ManagedConnection createManagedConnection(
Subject subject,
ConnectionRequestInfo info) {
System.out.println(
"In StarJCAManagedConnectionFactory. createManagedConnection");
return new StarJCAManagedConnection(this, "test");
}
…
…
public int hashCode() {
return 1;
}
}
```

Managed Connection

- **ManagedConnection**
- The ManagedConnection interface provides an application -level connection handle from the EIS to the resource adapter's ManagedConnection instance.
- Communication between the two occurs through listeners and event notifications. Support for error logging and tracing must be present.
- Metadata about this instance and the EIS can be retrieved by invoking getMetaData, which returns information encapsulated in a ManagedConnectionMetaData instance.
- The interface also provides methods, like cleanup, to reinitialize the instance and free resources after communication ceases.
- The instance does not close the connection, however. This is handled by the application server so connection pooling can be utilized.

Managed Connection (Sample)

```
public class StarJCAManagedConnection implements ManagedConnection {
private StarJCAConnectionEventListener myListener;
private String user;
private ManagedConnectionFactory mcf;
private PrintWriter logWriter;
private boolean destroyed;
private Set connectionSet;
StarJCAManagedConnection(ManagedConnectionFactory mcf, String user) {
System.out.println("In StarJCAManagedConnection");
this.mcf = mcf;
this.user = user;
connectionSet = new HashSet();
myListener = new StarJCAConnectionEventListener(this);
}
public Object getConnection(
Subject subject,
ConnectionRequestInfo connectionRequestInfo)
throws ResourceException {
System.out.println("In StarJCAManagedConnection.getConnection");
StarJCAConnection myCon = new StarJCAConnection(this);
addMyConnection(myCon);
return myCon;
}
```

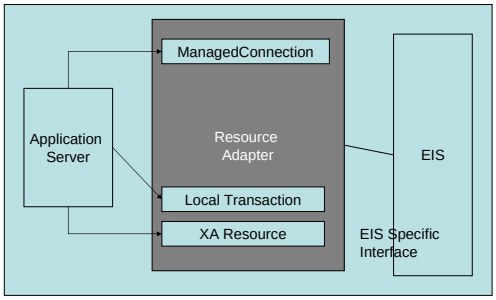
Managed Connection MetaData

- ManagedConnectionMetaData
- ManagedConnectionMetaData facilitates the retrieval of metadata about a ManagedConnection instance and the particular EIS.
- The metadata must provide the enterprise information system's product name and version, the maximum number of concurrent connections that it can support, and the username associated with the connection.

Managed Connection MetaData

```
public class StarJCAConnectionMetaData implements ManagedConnectionMetaData {
private StarJCAManagedConnection mc;
public StarJCAConnectionMetaData (StarJCAManagedConnection mc) {
System.out.println("In StarJCAConnectionMetaData. constructor");
this.mc = mc;
}
public String getEISProductName() throws ResourceException {
System.out.println("In StarJCAConnectionMetaData.getEISProductName");
return "myJCA";
}
public String getEISProductVersion() throws ResourceException {
System.out.println("In StarJCAConnectionMetaData.getEISProductVersion");
return "1.0";
}
public int getMaxConnections() throws ResourceException {
System.out.println("In StarJCAConnectionMetaData.getMaxConnections");
return 5;
}
public String getUserName() throws ResourceException {
return mc.getUserName();
}
}
```

Transaction Management



Transactions

- Transaction Management Interfaces
- The transaction management interfaces provide a framework that the application server uses to manage and perform transactions with the EIS for the enterprise application.
- These transactions are referred to as JTA or XA transactions because the transactions are performed across a variety of enterprise information systems.
- XA transactions are defined in the Java Transaction API (JTA) specification. The resource adapter can support either local transactions, both types of transactions, or neither type of transaction.

Security Interfaces

- The security contract enables secure access to the EIS from the application component. The security contract is adhered to by incorporating the J2EE Java Authentication and Authorization Service (JAAS) into the connection management interfaces.

Security

- Subject - A Subject represents a grouping of related information for a single entity, such as a person. It includes identities, each known as a Principal , and security-related attributes, which are called Credentials. The subject is passed to the resource adapter by the application server for authentication and authorization.
- Principal - As mentioned above, a Principal represents an identity.
- Generic Credential - The Generic Credential interface defines a way to access the security credentials of a Principal. With this interface, the resource adapter is able to retrieve the credentials passed from the application server.

Defining the Meta Contents

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE connector PUBLIC "-//Sun Microsystems, Inc.//DTD Connector 1.0//EN"
      "http://java.sun.com/dtd/connector_1_0.dtd">
<connector>
  <display-name>StarJCA</display-name>
  <description>Starwood ResourceAdapter</description>
  <vendor-name>Starwood</vendor-name>
  <spec-version>1.0</spec-version>
  <eis-type>RRA</eis-type>
  <version>1.0</version>
  <license>
    <description>Starwood ResourceAdapter</description>
    <license-required>false</license-required>
  </license>
  <resourceadapter>
    <managedconnectionfactory-class>com.starwood.jca.StarJCAManagedConnectionFactory     </managedconnectionfactory-
      class>
    <connectionfactory-interface>javax.resource.cci.ConnectionFactory     </connectionfactory-interface>
    <connectionfactory-impl-class>com.starwood.jca.StarJCAConnectionFactory     </connectionfactory-impl-class>
    <connection-interface>javax.resource.cci.Connection     </connection-interface>
    <connection-impl-class>com.starwood.jca.StarJCAConnection   </connection-impl-class>
    <transaction-support>NoTransaction</transaction-support>
    <authentication-mechanism>
      <description>BasicPassword authentication </description>
      <authentication-mechanism-type>BasicPassword</authentication-mechanism-type>
      <credential-interface>javax.resource.spi.security.PasswordCredential</credential-interface>
    </authentication-mechanism>
    <authentication-mechanism>
      <description>Kerbv5 Authentication </description>
      <authentication-mechanism-type>Kerbv5</authentication-mechanism-type>
      <credential-interface>javax.resource.spi.security.GenericCredential</credential-interface>
    </authentication-mechanism>
    <reauthentication-support>false</reauthentication-support>
    <security-permission>
      <security-permission-spec/>
    </security-permission>
  </resourceadapter>
</connector>
```

Building the Adapter (ANT Build)

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE connector PUBLIC "-//Sun Microsystems, Inc.//DTD Connector 1.0//EN" "file://c:/connector_1_0.dtd">
<project default="build">
  <property value="C:\redhat_8\jboss\jboss-3.0.4\server\default\lib\jboss-j2ee.jar" name="j2ee.jar"/>
  <target name="init">
    <delete dir="classes"/>
    <mkdir dir="classes"/>
    <mkdir dir="javadoc"/>
  </target>
  <target name="build" depends="init">
    <javac debug="on" destdir="classes">
      <classpath>
        <pathelement location="${j2ee.jar}"/>
      </classpath>
      <src path="."/>
    </javac>
    <jar jarfile="starjca.jar">
      <fileset dir="classes">
        <include name="**/jca/**"/>
      </fileset>
    </jar>
    <jar destfile="starjca.rar">
      <fileset dir=".">
        <include name="starjca.jar"/>
        <include name="META-INF/**"/>
      </fileset>
    </jar>
  </target>
</project>
```

Clients

- Non-Managed
  - POJOs and Servlets connecting directly.
- Managed
  - Through Session Beans and MBeans

Client (Non-Managed POJO)

```
public TestClient() {
try {
System.out.println("In TestClient - Non Managed Connection Started");
mcf = new StarJCAManagedConnectionFactory();
cxf =
(javax.resource.cci.ConnectionFactory) mcf
.createConnectionFactory();
connection = cxf.getConnection();
StarJCAInteractionSpec spec = new StarJCAInteractionSpec();
spec.setQuery("Hello");
spec.setUrl(local_test);
StarJCARecord result = new StarJCARecord();
StarJCAInteraction six = (StarJCAInteraction) connection.createInteraction();
System.out.println("In TestClient Interaction Received");
result = (StarJCARecord)six.execute(spec, null);
System.out.println("In TestClient Result " + result.getRecordName());
spec.setUrl(spg_authtoken);
six = (StarJCAInteraction) connection.createInteraction();
System.out.println("In TestClient Interaction Received");
result = (StarJCARecord)six.execute(spec, null);
System.out.println("In TestClient Result " + result.getRecordName());
} catch (Exception ex) { ex.printStackTrace(); }
}
```

Client (Managed)

```
  Context initCtx = null;
  Object obj = null;
  try {
    initCtx = getInitialContext();
    Object ref = initCtx.lookup("java:/StarJCA");
  } catch (Exception e) {
      System.out.println(
               "Error with context: " + e);
      e.printStackTrace();
  }
}
```

JBoss Configuration

- Starjca-ds.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<connection-factories>
  <no-tx-connection-factory>
    <adapter-display-name>StarJCA</adapter-display-name>
    <jndi-name>eis/StarJCA</jndi-name>
    <rar-name>starjca.rar</rar-name>
    <connection-
   definition>javax.resource.cci.ConnectionFactory</connection-definition>
  </no-tx-connection-factory>
</connection-factories>

<!-- call the transaction connection factory -->
<!-- tx-connection-factory -->
```

Deploying the RAR

- Run the ANT Script
- Copy the RAR file to
- /server/default/deploy folder within JBoss 4 installation

References

- JBoss 4.0 Documentation
- Sun Java Site.
  - The J2EE Connector Architecture's Resource Adapter
  - Jennifer Rodoni
- IBM Red Books:
  - Build JCA-compliant resource adapters with WebSphere Studio Application Developer
  - Michael McMahon (mmcmaho@us.ibm.com), Advisory Software Engineer, IBM
  - Mikhail Genkin (genkin@ca.ibm.com), Technical Solution Designer, IBM
- Connect the enterprise with the JCA Part 1 and 2
  - Java World - Dirk Reinshagen
- Sun Java Site.
  - Creating Resource Adapters with J2EE Connector Architecture 1.5
  - Alejandro E. Murillo and Binod P. G.,
- Sun Java Site.
  - J2EE Connection Architecture Specs

Questions

- Email: devb@ieee.org