IRON JACAMAR

Distributed work manager Requirements and design

Who is Stefano?

- JBoss WS[1] committer since early 2007
- Committer of various Jboss projects
- JBoss / Red Hat employee since July 2010
- JBoss AS7[2] team memeber
- IronJacamar[4] team member
- Wise[3] project leader
- Follow me: www.javalinux.it, www.twitter.com/maeste https://plus.google.com/102371438013760231375/about https://github.com/maeste http://www.linkedin.com/in/maeste

[1] http://www.jboss.org/jbossws [2] http://www.jboss.org/jbossas

[3] http://www.jboss.org/wise [4] http://www.jboss.org/ironjcamar

Agenda

•What is EE : an introduction to java EE and applicacion servers. Example based on JBoss AS7
•What is JCA: an introduction of Java Connector Architecture standard and how it impacts server and resources connected to server (resource adapters)
•What is the JDBC resource adapter: a special case of resource adapters is JDBC (Java DataBase connection) one. Concepts on about drivers abstracting real DataBase to standard APIs
•JDBC driver deployment: how a JDBC driver is deployed and used in a Java EE server. Example based on JBoss AS7
•Datasource deployment: what is a datasource and how to deploy and use it in a Java EE server. Example based on JBoss AS7
•Monitor the datasource: how to monitor statistics of a datasource in JBoss AS7
•Other layers: Higher level layers to interact to database and Object to Relational Mapping (ORM) solutions in java EE: JPA, EJB/CMP

What is Java EE

- A collection of enterprise technologies
- Provides a component based approach to the design, development, assembly and deployment of enterprise applications
- Enables solutions for developing, deploying and managing n-tier server-centric enterprise applications
- An open industry standard

What is a Java EE Apllication Server

Provide a platform-independent, portable, multi-user, secure, and standard enterprise-class platform for server side deployments written in the Java Language

Implement a standardized execution environment for distributed enterprise applications

Why JEE?

Platform value for developers

Can use any Java EE implementation for development and deployment

Vast amount of Java EE community resources

Can use off-the-shelf 3rd party components

Why JEE?

Platform value to vendors

Vendors work together on specifications and then compete in implementations

In the areas of Scalability, Performance, Reliability, and so on

Freedom to innovate while maintaining the portability of applications

Why JEE?

Platform value to Business customers

Application portability

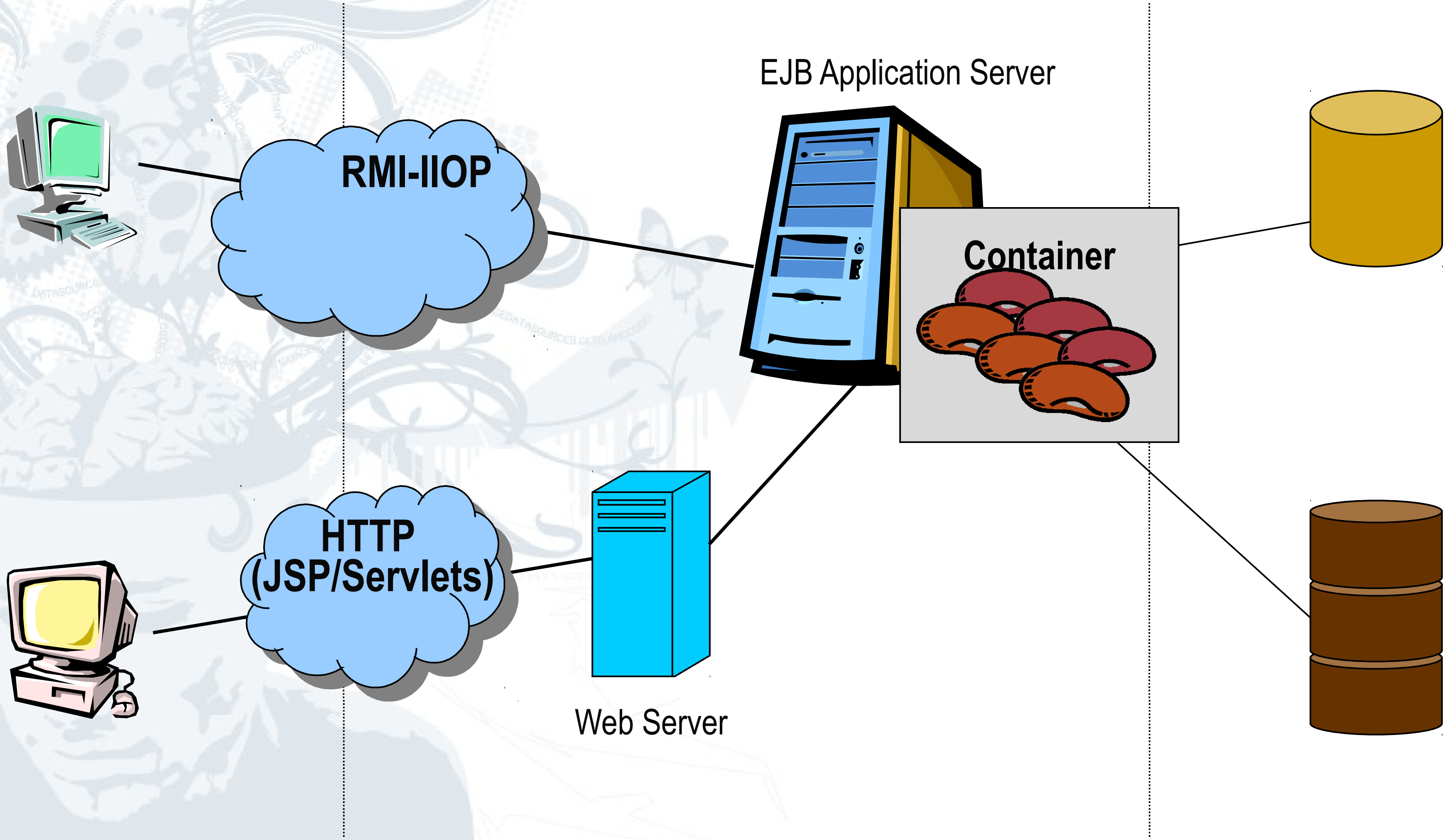
Many implementation choices are possible based on various requirements

Price (free to high-end), scalability (single CPU to clustered model), reliability, performance, tools & more

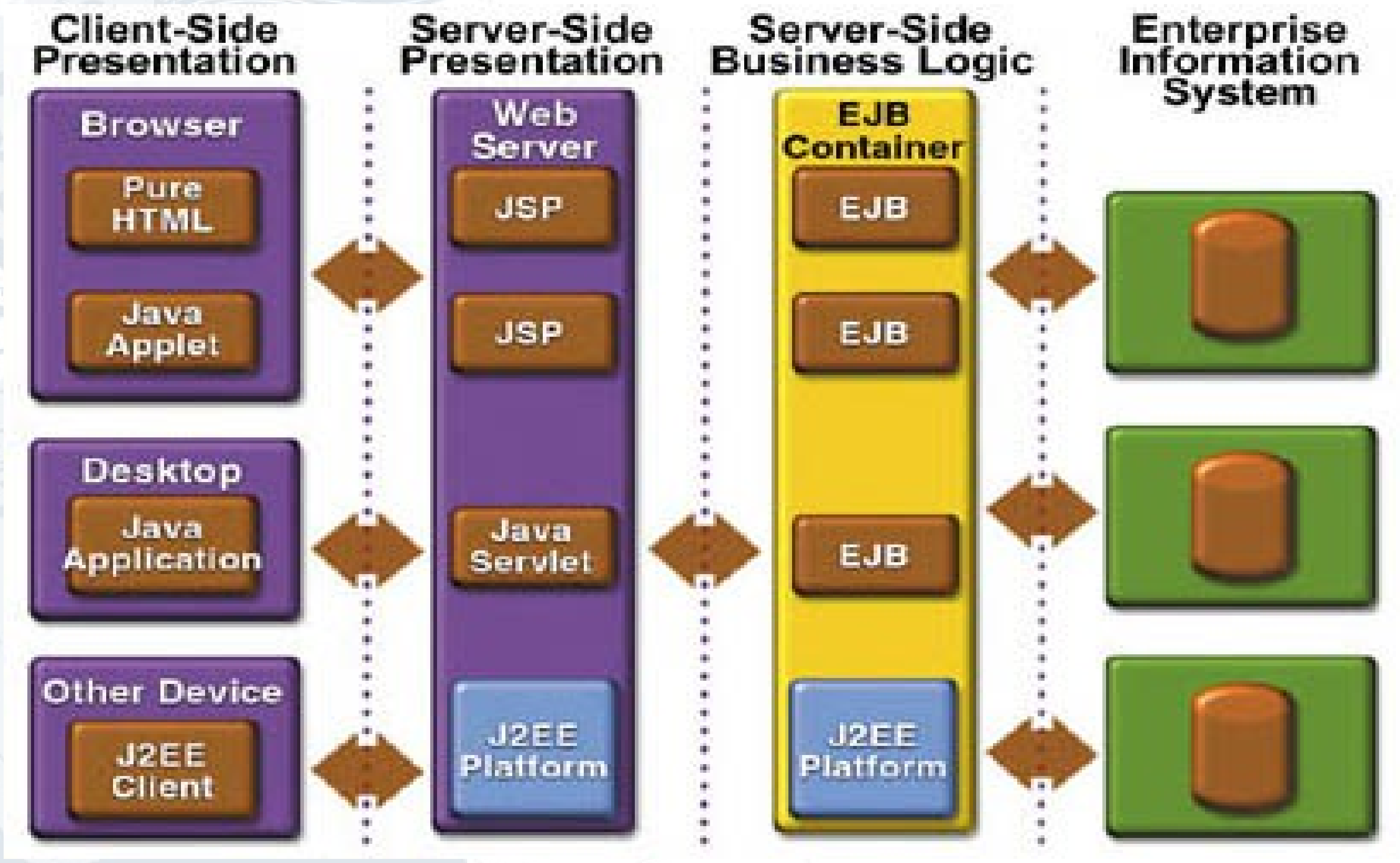
Best of breed of applications and platforms

Large developer pool

N-tier J2EE Architecture

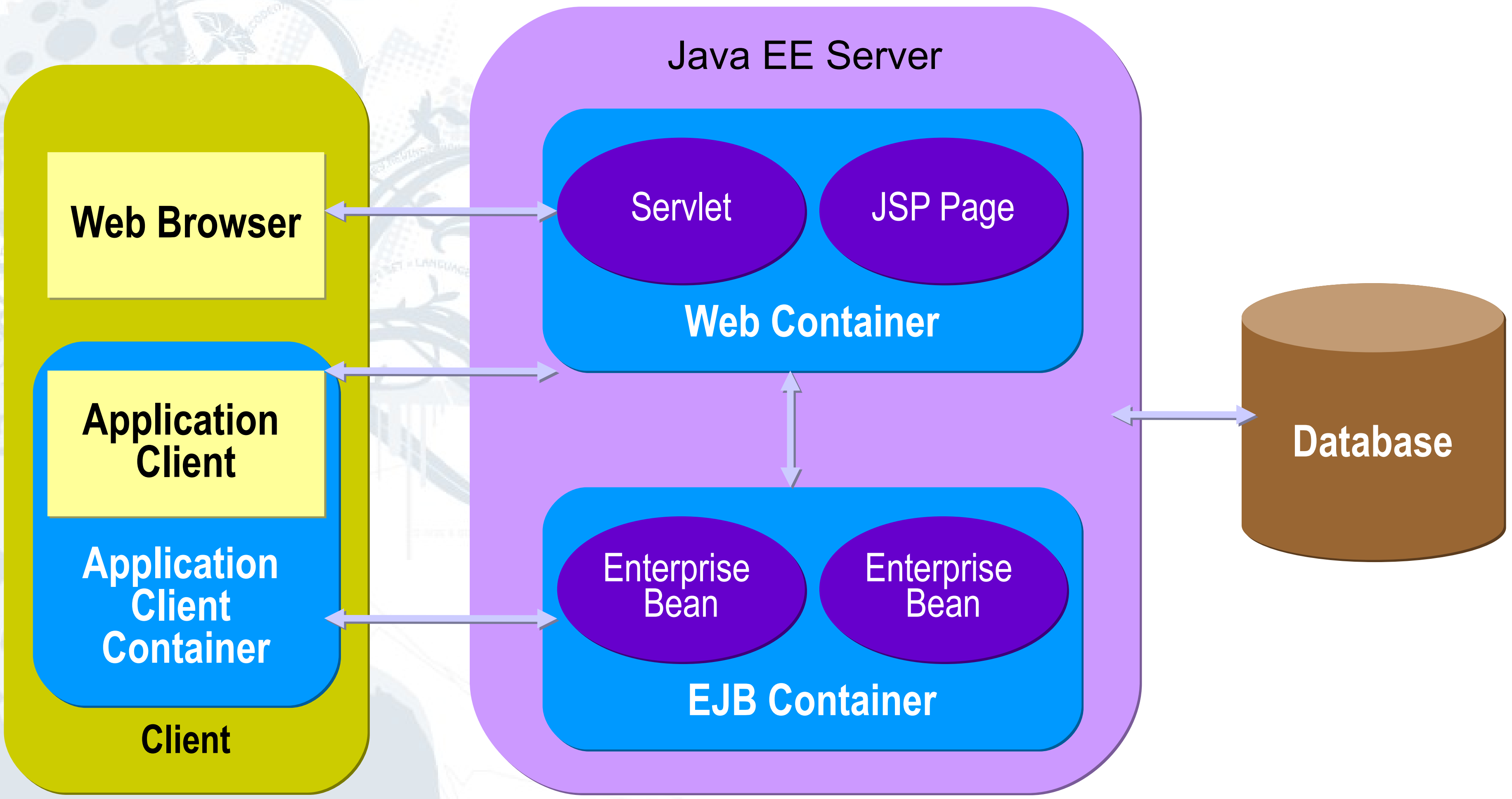


Java EE Architecture – Tiers and Components View



Client Tier Web Tier Business Tier Data Tier

Java EE Server and Containers



What is JCA

- The Java Connector Architecture (JCA)
- standard specifies how the Java Enterprise
- Edition platform connects to Enterprise
- Information Systems (EIS)
- – Database systems
- – Messaging system
- – Mainframe transaction processing
- Standard is created in JCP

JCA concepts

- JCA defines the contract between the JCA container and the EIS
    - Scalable, Transactional, Secure, Client API
- Resource Adapter (EIS vendors)
    - Protocol specific communication
    - Multiple AS vendors
- Resource Adapter (AS vendors)
    - Support multiple EISs

JCA concepts

- Resource adapters can be divided into
  - Outbound, Inbound and Bidirectional
- Transactions
  - NoTransaction, LocalTransaction, XATransaction
- Security
  - javax.security.auth.Subject

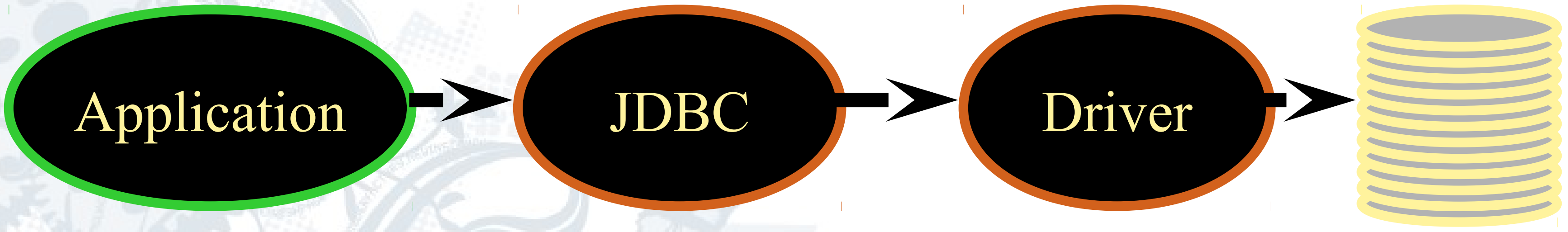
JCA concepts (Examples)

- JDBC resource adapter (outbound)
  - Wraps JDBC API and adds connection pooling
- Timer resource adapter (inbound)
  - Enables Quartz to invoke MessageEndpoints
- JMS resource adapter (bidirectional)
  - Wraps JMS API (outbound part)
  - Enables message invocation on MDBs (inbound)

JDBC...a standard API abstracting DB real API

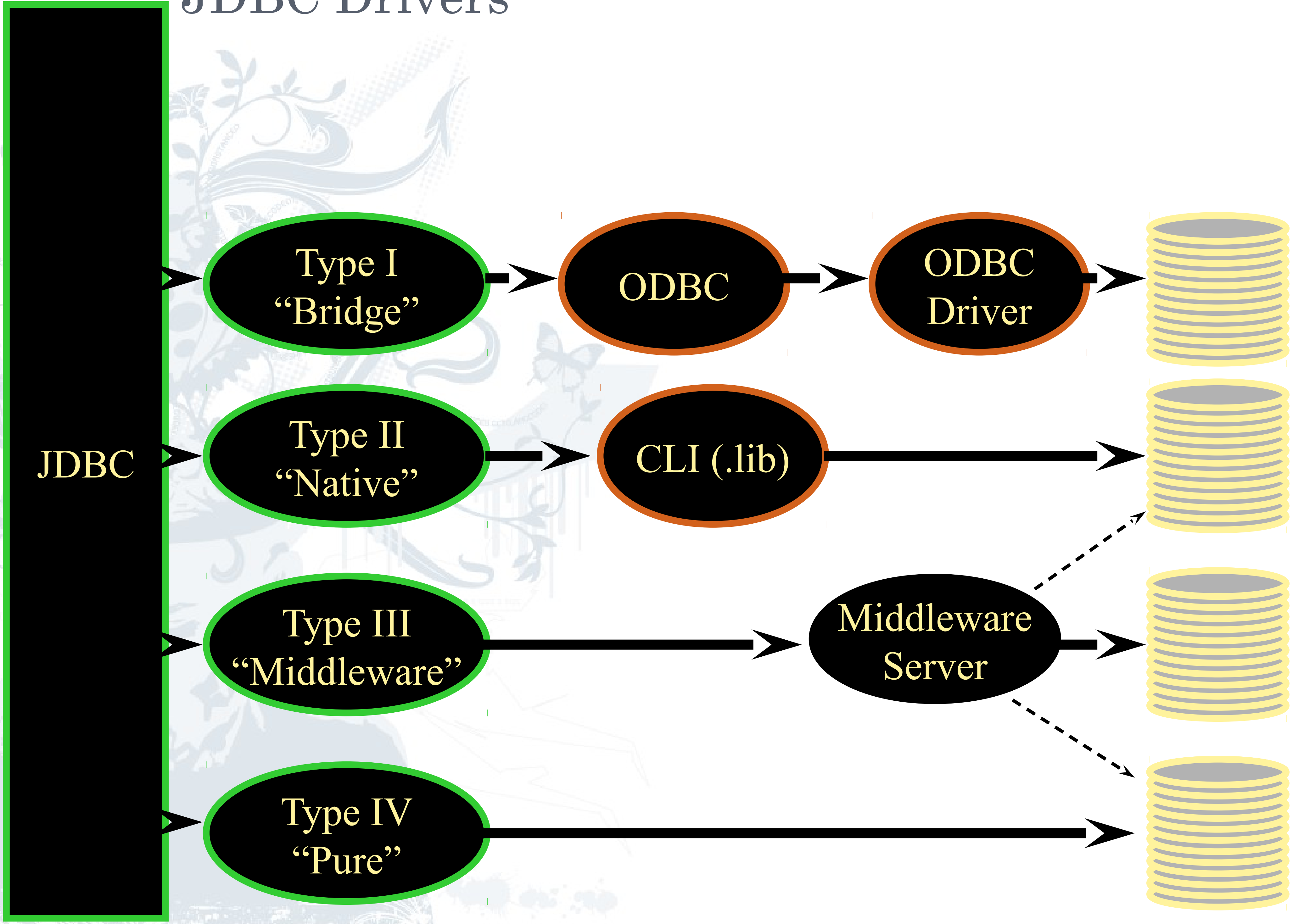
- Is an API spec. whose implementation comes in the form of jdbc drivers.
- JDBC Driver
    - Is a bridge s/w between java application and database s/w.
    - Is a java class that implements java.sql.Driver interface.

JDBC Architecture



- Java code calls JDBC library
- JDBC loads a *driver*
- Driver talks to a particular database
- Can have more than one driver -> more than one database

JDBC Drivers



Developing standard JDBC application

- Load the JDBC Driver class and register with DriverManager.
- Establish the connection with database s/w.
- Prepare Statement object
- Execute the query.
- Get result and process the result
- Close the connection.

JDBC driver in a EE server (JBoss)

- Just hot deploy the sql driver jar for example postgresql-9.1-901-1.jdbc4.jar
- It will load Driver and register it to DriverManager.
- It's ready tobe used by JDBC resource adapter aka DataSource

JDBC resource adapter

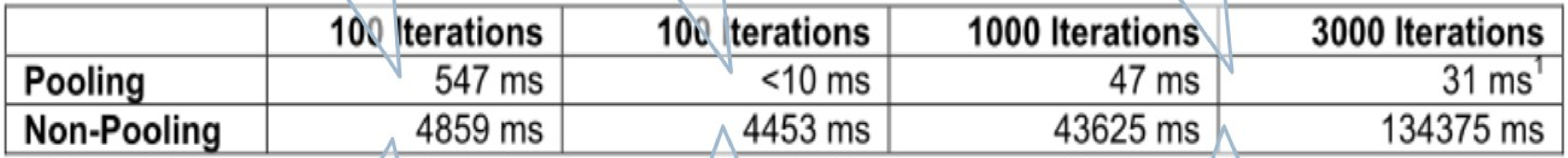
- Add datasource concept. It's basically add:
  - connection pooling ability
  - Transactional support (local and XA)
  - Security support (EE security subject and role)
  - Connection validation, pool flush strategy, pool statistics, connection recovery and reautentication....

Connectio pooling

The first time, the connections must be created

Second time, reuse connections

Negligible overhead



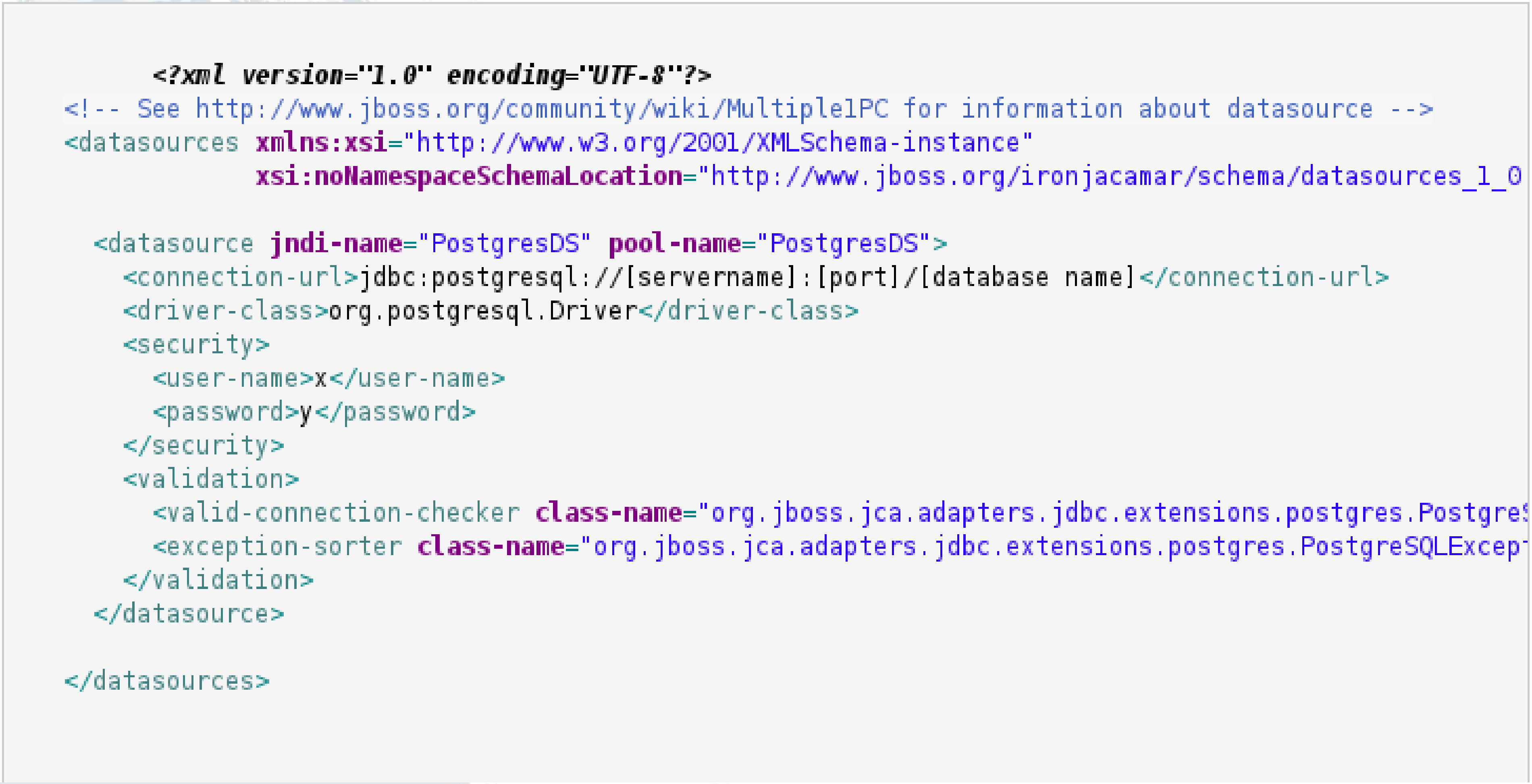
	100 Iterations	100 Iterations	1000 Iterations	3000 Iterations
Pooling	547 ms	<10 ms	47 ms	31 ms[1]
Non-Pooling	4859 ms	4453 ms	43625 ms	134375 ms

10x slower

No improvement

Linear increase

Deploying a datasource and using it

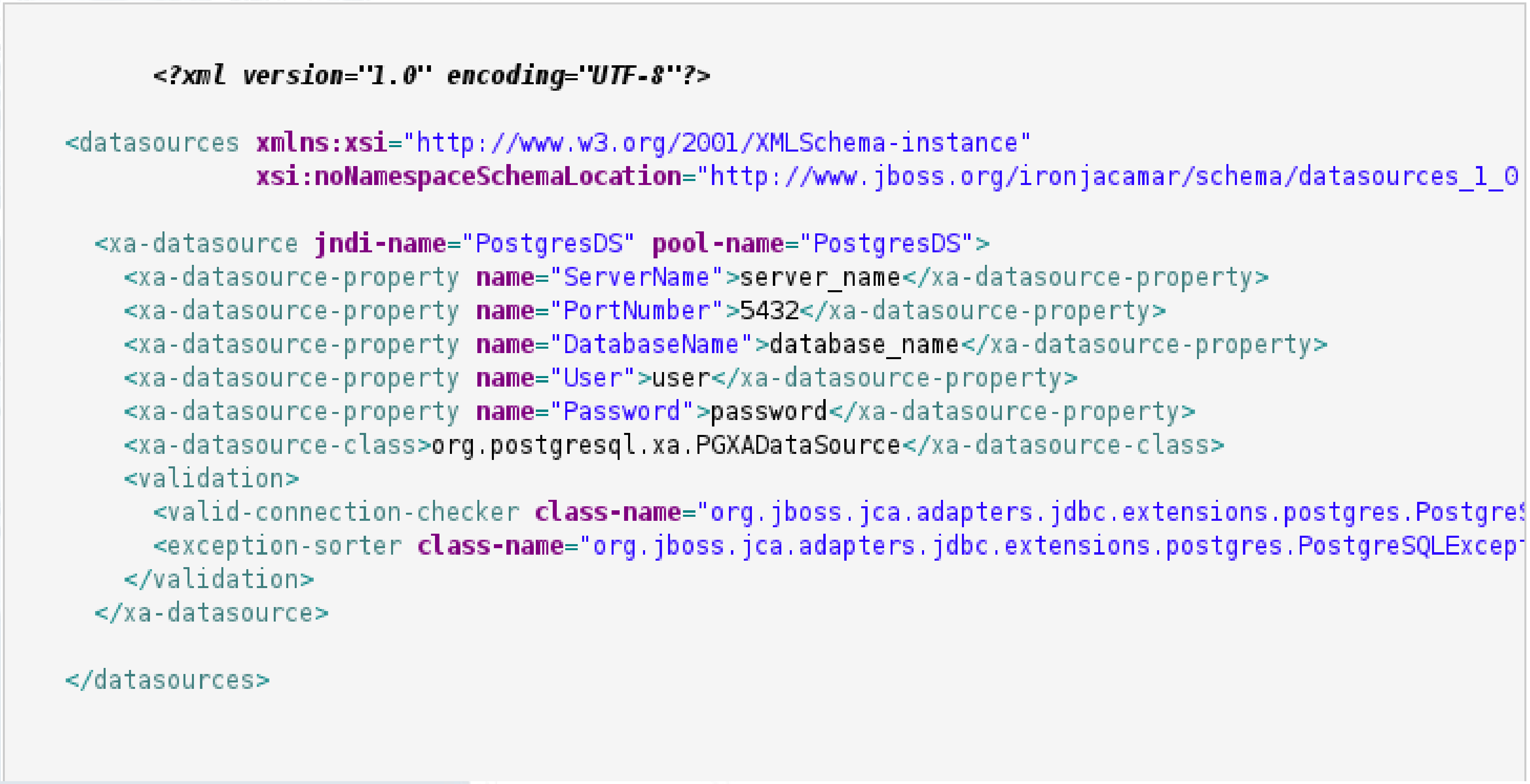


```
     <?xml version="1.0" encoding="UTF-8"?>
<!-- See http://www.jboss.org/community/wiki/Multiple1PC for information about datasource -->
<datasources xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             xsi:noNamespaceSchemaLocation="http://www.jboss.org/ironjacamar/schema/datasources_1_0

  <datasource jndi-name="PostgresDS" pool-name="PostgresDS">
    <connection-url>jdbc:postgresql://[servername]:[port]/[database name]</connection-url>
    <driver-class>org.postgresql.Driver</driver-class>
    <security>
      <user-name>x</user-name>
      <password>y</password>
    </security>
    <validation>
      <valid-connection-checker class-name="org.jboss.jca.adapters.jdbc.extensions.postgres.Postgre
      <exception-sorter class-name="org.jboss.jca.adapters.jdbc.extensions.postgres.PostgreSQLExcep
    </validation>
  </datasource>

</datasources>
```

Deploing a datasource and using it



```
     <?xml version="1.0" encoding="UTF-8"?>

<datasources xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             xsi:noNamespaceSchemaLocation="http://www.jboss.org/ironjacamar/schema/datasources_1_0

  <xa-datasource jndi-name="PostgresDS" pool-name="PostgresDS">
    <xa-datasource-property name="ServerName">server_name</xa-datasource-property>
    <xa-datasource-property name="PortNumber">5432</xa-datasource-property>
    <xa-datasource-property name="DatabaseName">database_name</xa-datasource-property>
    <xa-datasource-property name="User">user</xa-datasource-property>
    <xa-datasource-property name="Password">password</xa-datasource-property>
    <xa-datasource-class>org.postgresql.xa.PGXADataSource</xa-datasource-class>
    <validation>
      <valid-connection-checker class-name="org.jboss.jca.adapters.jdbc.extensions.postgres.Postgre
      <exception-sorter class-name="org.jboss.jca.adapters.jdbc.extensions.postgres.PostgreSQLExcep
    </validation>
  </xa-datasource>

</datasources>
```

Developing a JDBC application with Datasource facilities

- ~~Load the JDBC Driver class and register with DriverManager.~~
- ~~Establish the connection with database s/w~~ -> get connection from datasource pooling
- Prepare Statement object
- Execute the query.
- Get result and process the result
- ~~Close the connection.~~

Monitor a datasource and read statistics

- JBoss provide some statistics about use of connection pooling for JCA and for standard JDBC datasources
- It's a pluggable architecture to leave opportunity to EIS vendor to implement their own

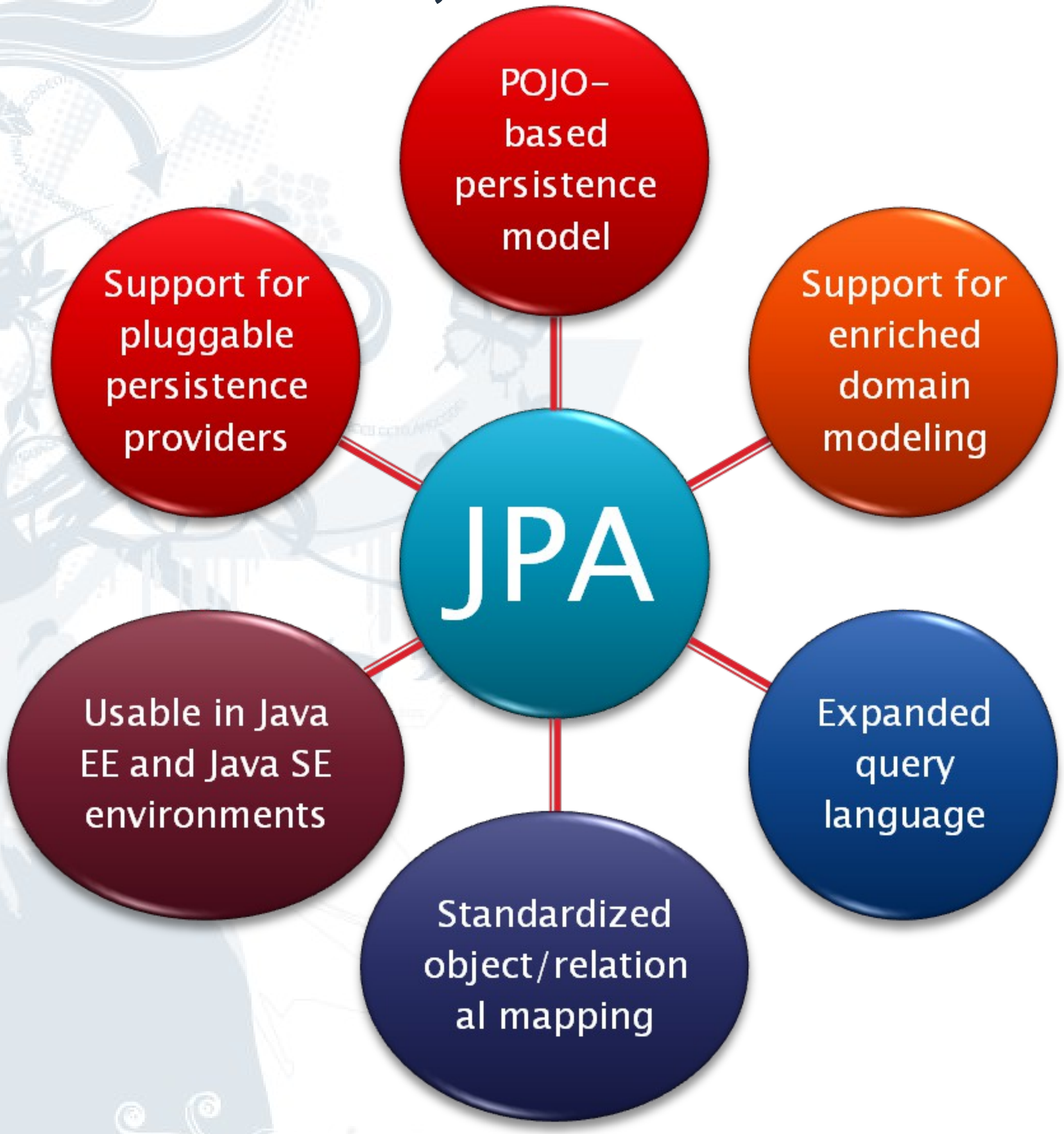
Why IronJacamar

- Can be used standalone
- Have tools for generationg skeleton of an RA implmentation (mvn, ant and eclipse)
- Have validators to validate your implementation against most important points of spec
- It's open source....

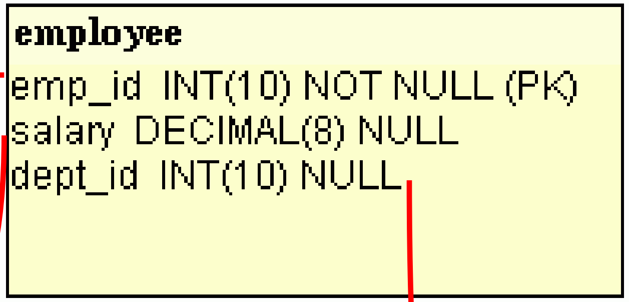
Higher layer of DB interaction
ORM, JPA and EJB

The Java Persistence API is the standard object/relational mapping and persistence management interface of the Java EE 5.0 platform. As part of the EJB 3.0 specification effort, it is supported by all major vendors of the Java industry.

Higher layer of DB interaction ORM, JPA and EJB

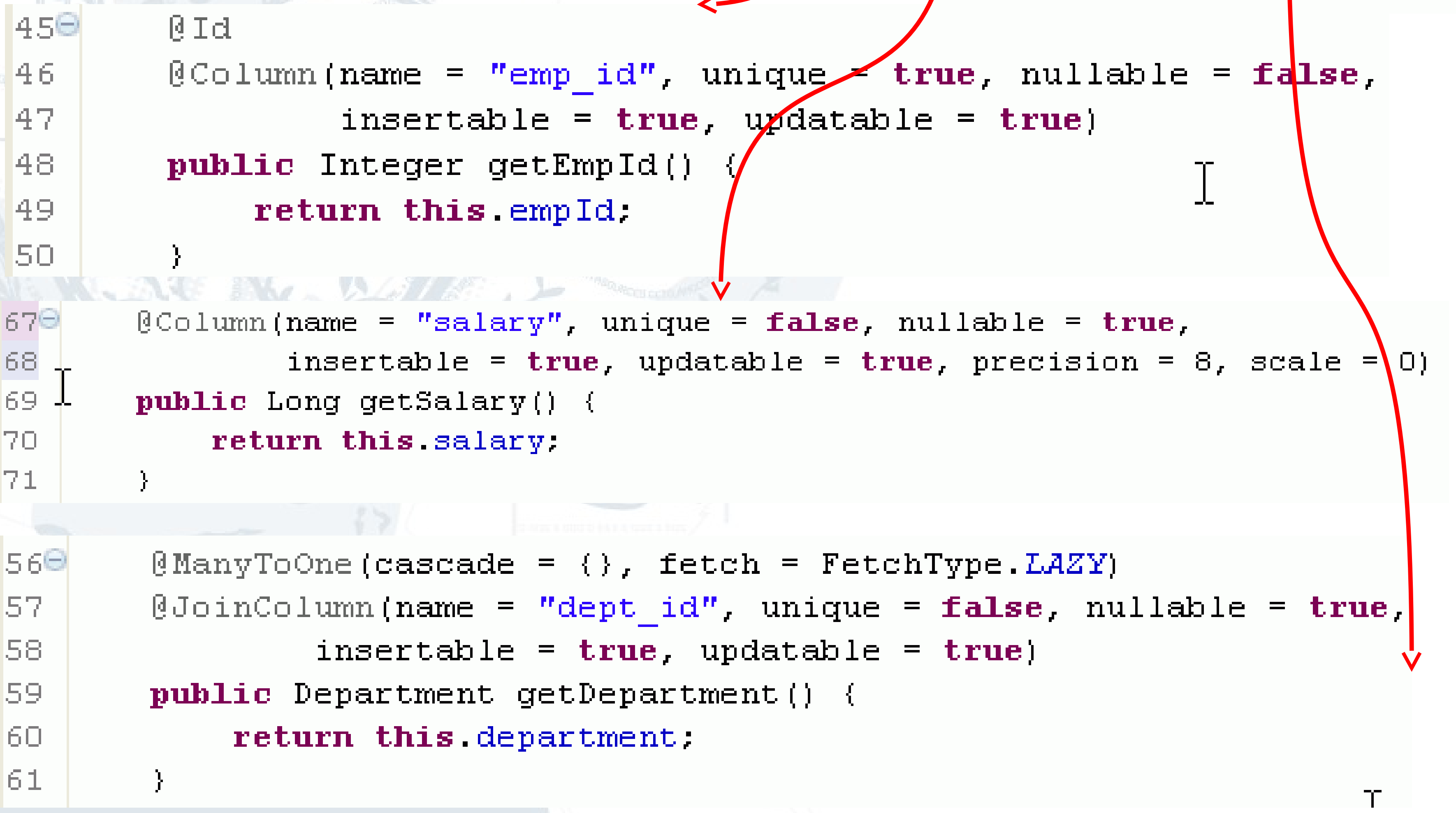

JPA
POJO-based persistence model
Support for enriched domain modeling
Expanded query language
Standardized object/relational mapping
Usable in Java EE and Java SE environments
Support for pluggable persistence providers

ORM in practice



employee

```
emp_id  INT(10) NOT NULL (PK)
salary  DECIMAL(8) NULL
dept_id  INT(10) NULL
```



```
    @Id
    @Column(name = "emp_id", unique = true, nullable = false,
              insertable = true, updatable = true)
    public Integer getEmpId() {
        return this.empId;
    }
```

```
    @Column(name = "salary", unique = false, nullable = true,
              insertable = true, updatable = true, precision = 8, scale = 0)
    public Long getSalary() {
        return this.salary;
    }
```

```
    @ManyToOne(cascade = {}, fetch = FetchType.LAZY)
    @JoinColumn(name = "dept_id", unique = false, nullable = true,
              insertable = true, updatable = true)
    public Department getDepartment() {
        return this.department;
    }
```

Contribute...

- http://www.jboss.org/ironjacamar
- http://www.jboss.org/as7

Q & A