Java and/in DB2

GSE NL DB2 working group

Bussum - 28/10/2010

Gie Indesteege - ABIS Training & Consulting

Welcome

ABIS Training & Consulting

www.abis.be

abis
TRAINING & CONSULTING

Gie Indesteege

- trainer and consultant
- president of BeNeLux GSE working group EGL/RDZ

Contents

Java for enterprise critical applications !

- **business requirements: functional, performance, security, ... and**
- **consistent and correct data**

Java and DB2

- **connection between Java application and DB2**
- **Java functionality inside DB2**

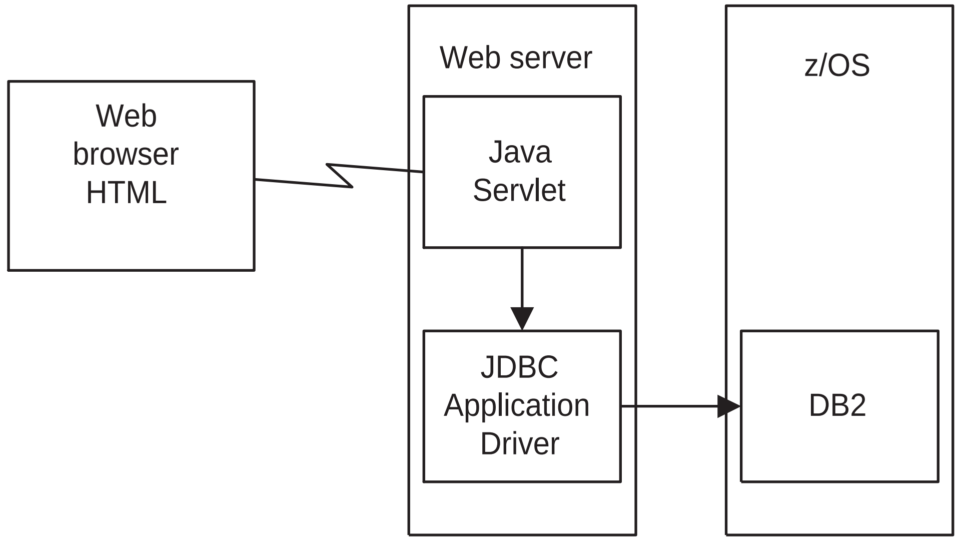
An overview of the architecture and possibilities of the Java - DB2 combination

Agenda

- **Java Database Connectivity**
- **Object Relational Mapping**
- **Java Persistency Architecture**
- **Persistence Frameworks**
- **Java in DB2: Stored Procedures and User Defined Functions**
- **Q & A**

Java Database Connectivity

Connection between Java application and (local or remote) DB

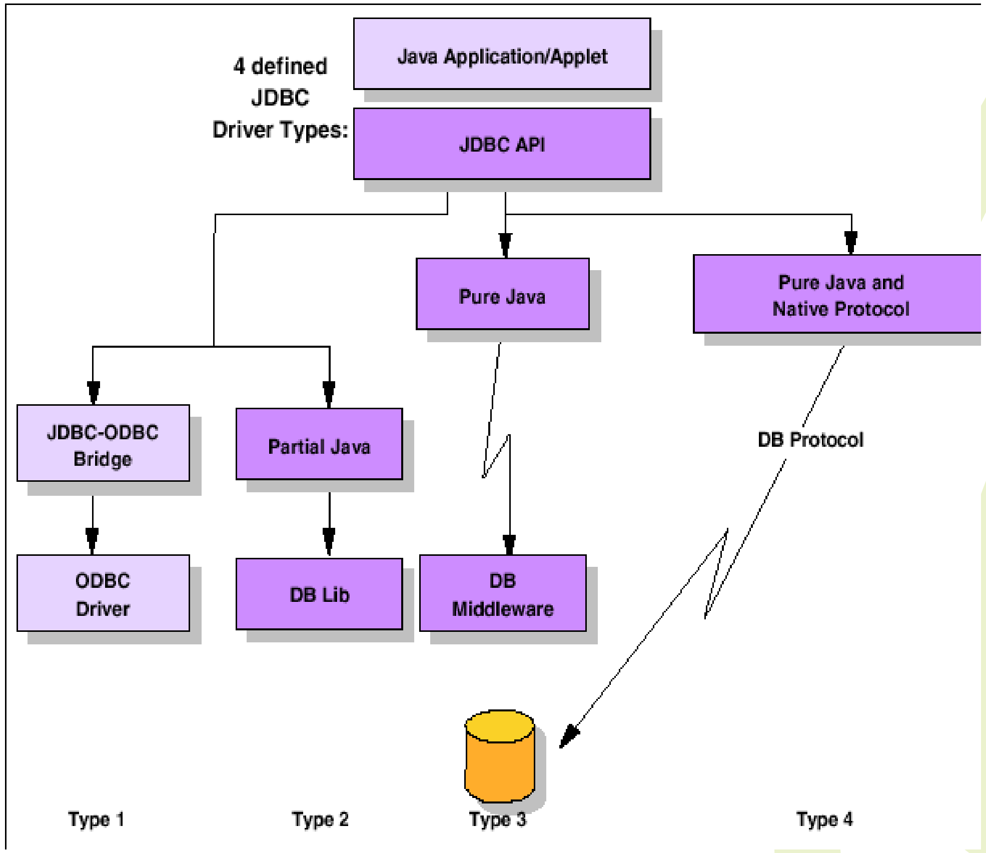


dynamic access to DB via standard JDBC API

Note: embedded (static) SQL possible via SQLJ

JDBC (Java Database Connectivity) API

- **Driver - types**
  - **1: JDBC - ODBC**
  - **2: app driver (*)**
    local connection
  - **3: net driver**
  - **4: universal driver (*)**
    remote connection



(*) DB2 for z/OS supports the IBM Data Server Driver for JDBC

JDBC API (cont.)

Set up connection via

- DriverManager
- Connection

Connection connection = DriverManager.getConnection(databaseUrl , userid , password);

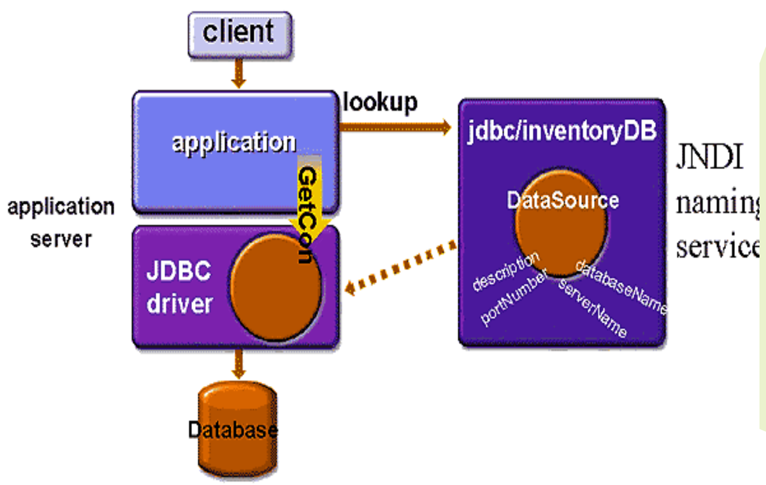
or use DataSource

DataSource ds = (DataSource)ctx.lookup("jdbc/inventoryDB");

connection = ds.getConnection();

via Java Naming and Directory Interface (JNDI)

Connecting to a Data Source



JDBC API (cont.)

Create SQL query via

- dynamic statement
  - Statement
  - PreparedStatement
  - CallableStatement (for Stored Procedures)

```
PreparedStatement preparedSQLStatement =
        connection.prepareStatement("select email from person where id = ?" );
```

- static statement (see SQLJ)

Execute query and analyse result via

```
preparedSQLStatement.setInt( 1, personNr );
ResultSet resultSet = preparedSQLStatement.executeQuery();
```

Error handling via

- SQLException
- SQLWarning

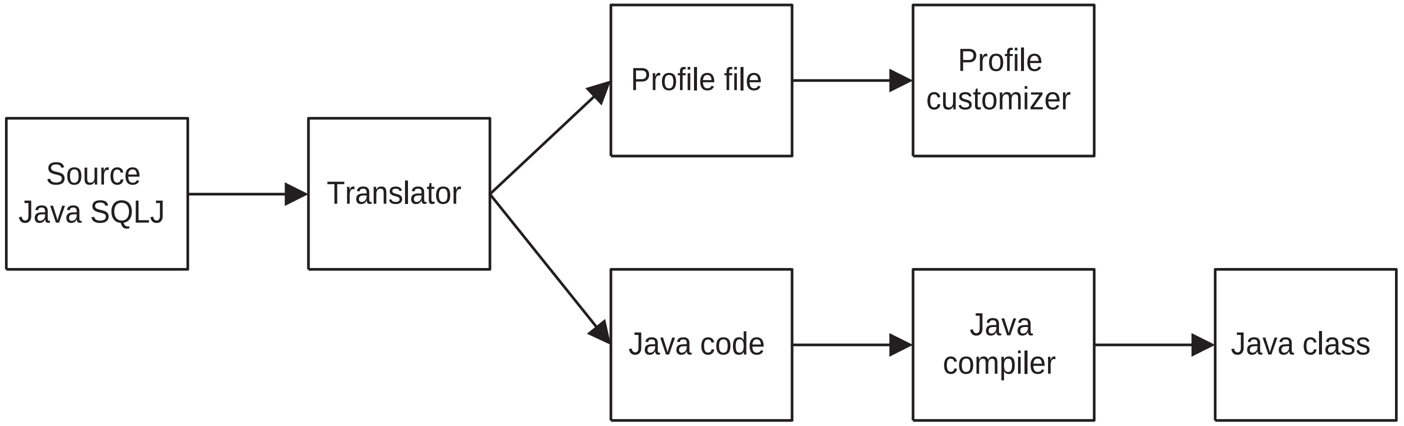
SQLJ

support for embedded **static** SQL

initially developed by IBM, Oracle, and Tandem

Example

```
#sql context EzSqljCtx;
#sql iterator EzSqljNameIter (String EMAIL);
...
#sql [ctx] iter = { SELECT email FROM person };
while (iter.next()) {
    System.out.println(iter.EMAIL());
    ...
    }
```



JDBC additional considerations

Datasource configuration via Java EE container

Transaction processing

- local vs distributed
- application vs database
- transaction isolation -> locking (UR, CS, RS, RR)
- commit/rollback

Resultset processing

- Scrollable resultsets
- Updateable resultsets
- Disconnected resultsets -> rowsets

Connection control

Metadata

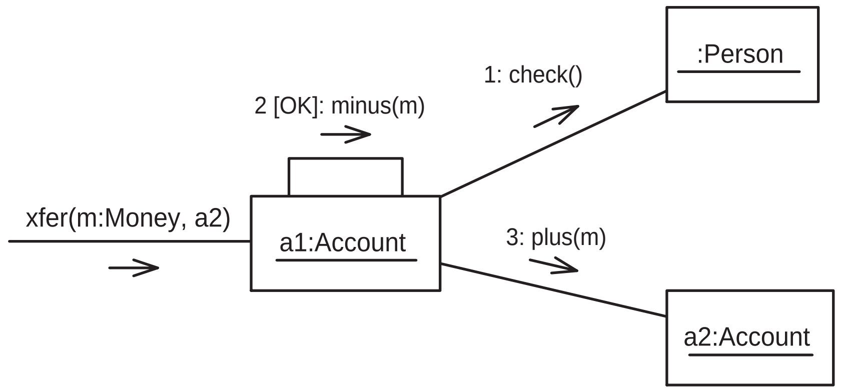
Agenda

- Java Database Connectivity
- **Object Relational Mapping**
- Java Persistency Architecture
- Persistence Frameworks
- Java in DB2: Stored Procedures and User Defined Functions

Object Relational Mapping

OO-applications are composed of objects which

- consist of **data** and **behaviour**
- are connected to each other
- send messages to each other



How to persist objects?

Object Relational Mapping (cont.)

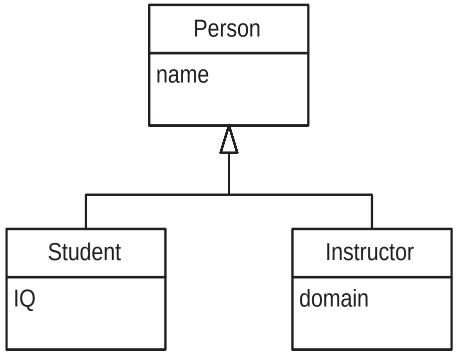
The big ‘impedance mismatch’:

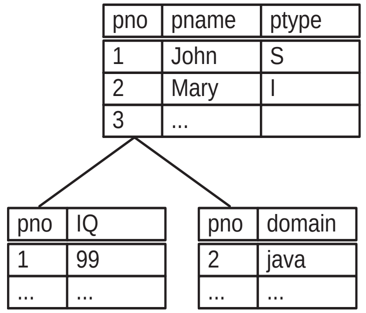
objects in memory << >> relational data on disk

OO	**RDBMS**
class	table
object	row
object identity	primary key
attribute	column
association	foreign key
accessors & other methods	SQL, triggers, stored procedures
hopping (traversal)	joining
generalization	combination

Object Relational Mapping - structural problems

- inheritance trees
  - tree to multiple tables
  - tree to single table
- identity field
  - primary key vs object ID
- mapping relationships
  - direction of relation?
  - uni/bidirectional?
  - one-to-many?
  - many-to-many?

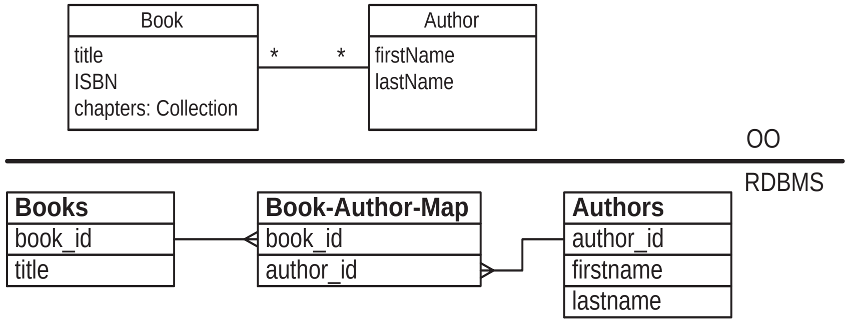




pno	pname	ptype
1	John	S
2	Mary	I
3	...	

pno	IQ
1	99
...	...

pno	domain
2	java
...	...



Books
book_id
title

Book-Author-Map
book_id
author_id

Authors
author_id
firstname
lastname

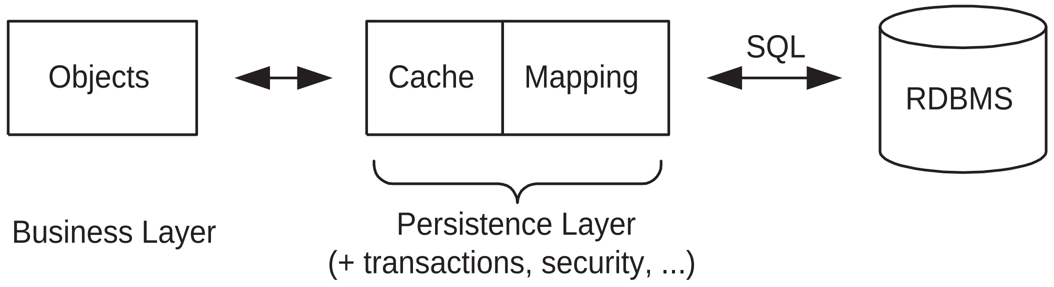
Object Relational Mapping - behavioural problems

- avoid to load complete database in memory
- how to load and save (persist) object in database
- concurrency problem
- keep track of changed objects for later storage in database

Object Relational Mapping - architectural problems

- class modelling of data tables
- where to code SQL (in which tier)?
- DB schema in Java source? in XML configuration?
- who dictates: Class model - Database schema?

Persistence mechanisms



- persist
  - data (object attributes) - types
  - relationships (links between objects)
  - metadata driven! (XML configuration or Annotations)
- synchronise (memory - disk)
  - caching
  - (container) service
- concurrency control
  - different users/applications must reach the same data at the same time
  - ... while keeping the data in a consistent state

Persistence mechanisms (cont.)

- transactional
  - set of actions that move data from one consistent state to another
  - key features: Atomicity, Consistency, Isolation, Durability
- query language:
  - retrieve data selective from the data store
  - SQL, OQL, xxxxQL
- identity support
  - avoid multiple copies of the same data
- security
  - unauthorized people must not see sensitive data
- performance
- ...

Agenda

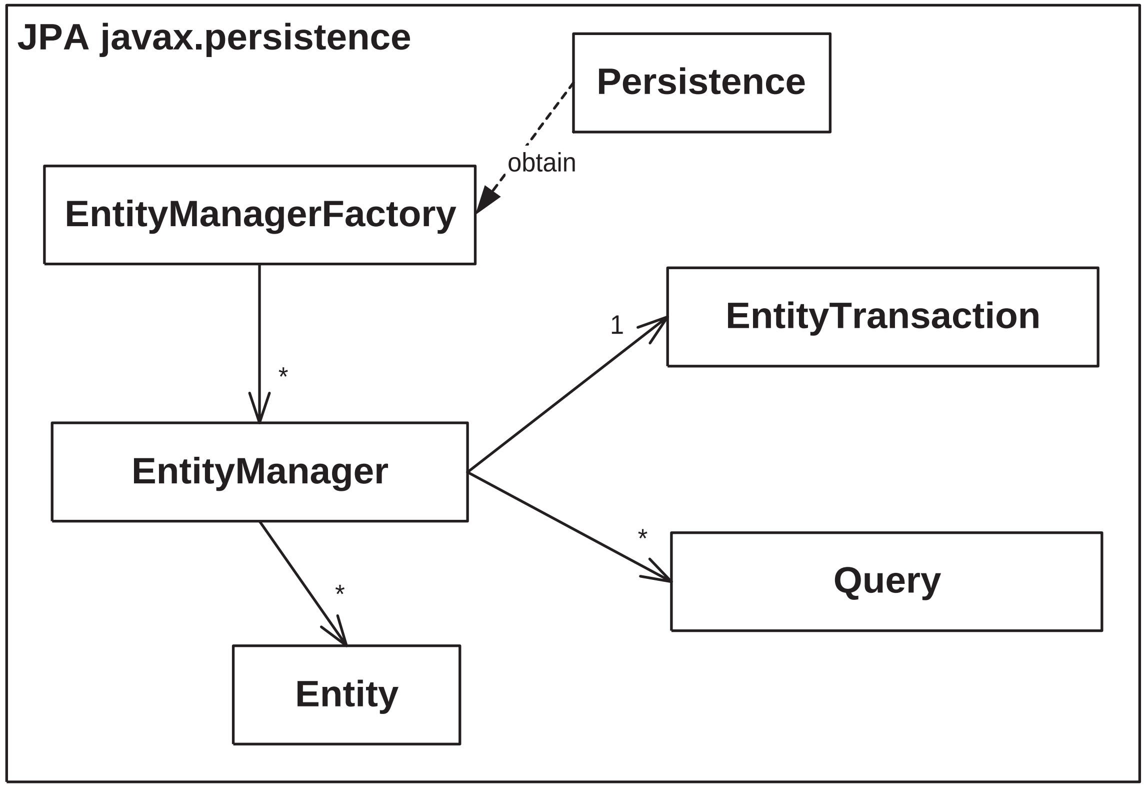
- Java Database Connectivity
- Object Relational Mapping
- Java Persistency Architecture
- Persistence Frameworks
- Java in DB2: Stored Procedures and User Defined Functions

Java Persistency Architecture (JPA)

since Java EE 5.0

- **transparent persistency framework for POJOs (Plain Old Java Objects) -> persistent entity objects**
- **Object-relational mapping framework**
  - **persistent fields**
  - **entity relationships**
  - **entity inheritance**
- **entity manager -> control of persistence unit**
- **configuration by exception (XML and/or annotations)**
- **transactional support**
- **native query support (Java Persistence Query Language - JPQL)**
- **detached entities -> serializable**
- **possibility to work inside or outside of a container**

Java Persistency Architecture (JPA) - Architecture



Java EE containers provide services for factory and transaction management

Java Persistence Architecture terminology

- Persistence: get EntityManagerFactory instances (vendor-neutral)
- EntityManagerFactory: factory for EntityManagers
- EntityManager: manages a set of persistent objects

  acts also as factory for Query instances
- Entity: persistent object that represents datastore records
- EntityTransaction: (associated with 1 EntityManager) group operations on persistent data into consistent units of work
- Query: interface to find persistent objects that meet certain criteria

  uses both the Java Persistence Query Language (JPQL) and the Structured Query Language (SQL)

JPA configuration

data source + persistent entities -> persistence.xml

```
<persistence-unit name="DemoJPA" transaction-type="JTA">
    <jta-data-source>jdbc/MyDB</jta-data-source>
    <class>be.abis.entities.Course</class>
</persistence-unit>
```

object/relational mapping + default schema -> orm.xml or via annotations in Entity

```
import javax.persistence.*;
@Entity
@Table (name = "TUTCOURSES")
@NamedQuery (name = "findAllCoursesByName",
     query = "SELECT c FROM Course c WHERE c.cltitle
     LIKE :courseName ORDER BY c.cltitle")
public class Course implements Serializable {
    @Id
    private String cid;
    @Column(name = ”CSTITLE”)
    private String courseTitle;
    ...
```

Use of JPA in application

Define persistence context

```
@PersistenceContext(unitName = "DemoJPA")
    private EntityManager em;
```

Use entity manager for persistence

```
// looking for existing course entity
Course crs = em.find(Course.class, courseNr);
// make modifications to course entity
crs.setTitle(“Java Persistence Architecture”);
// save to database
em.persist(crs);
```

Agenda

- Java Database Connectivity
- Object Relational Mapping
- Java Persistency Architecture
- Persistence Frameworks
- Java in DB2: Stored Procedures and User Defined Functions

Persistence Frameworks

alternative for or collaborating with JPA

Examples:

- Enterprise Java Beans of type Entity (part of J2EE 1.4) -> obsolete
  - Bean Managed Persistence (BMP) or
    Container Managed Persistence (CMP)
  - O/R mapping of persistent fields and relations in
    (XML) deployment descriptor
  - EJBQL
  - container responsible for persistence, synchronisation,
    transaction, security, ...
- Java Data Objects (JDO) - JSR 243 - Apache project

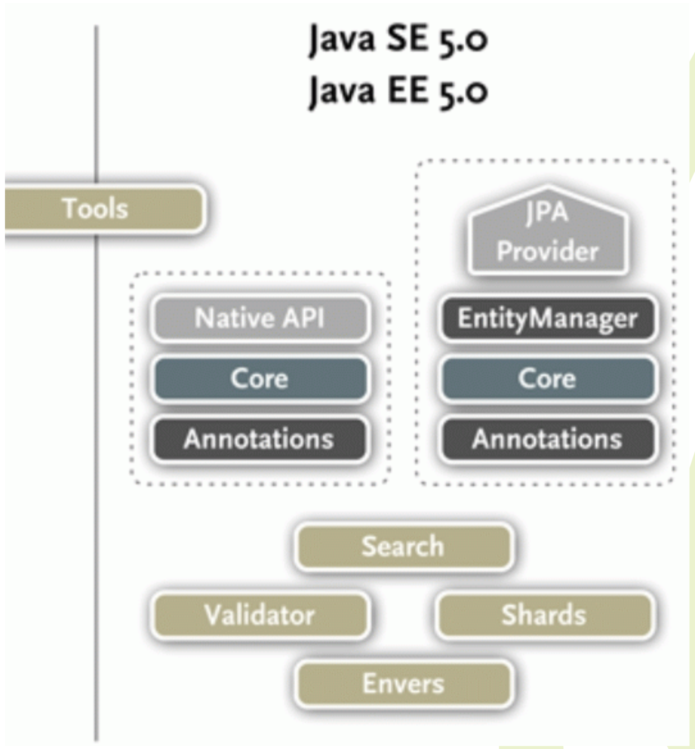
  http://db.apache.org/jdo/index.html
  - any datastore, not only RDBMS
  - ‘superset’ of JPA
  - JDOQL

Persistence Frameworks (cont.)

- **Hibernate (part of JBoss community)**

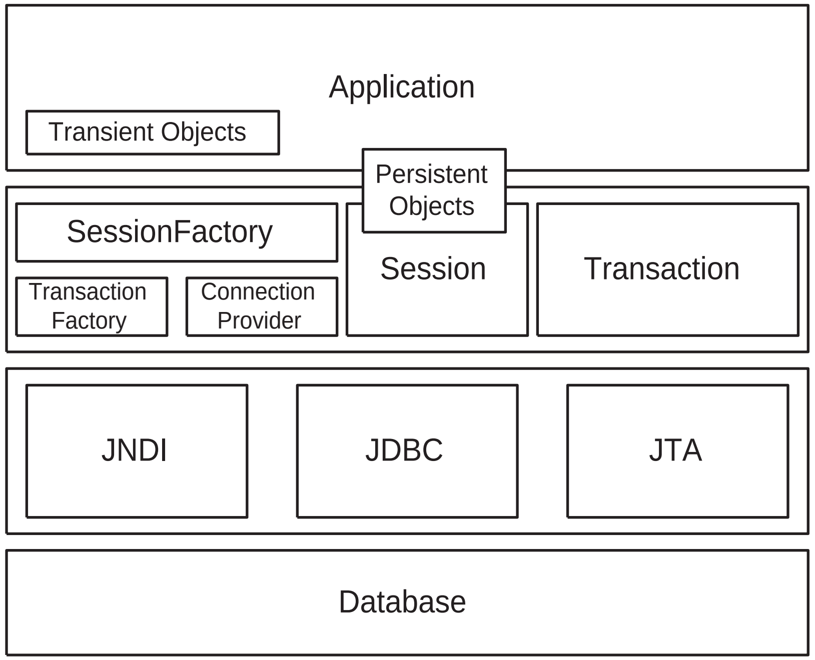
  www.hibernate.org

  - **transparent (open source) persistency framework for POJOs (Plain Old Java Objects)**
  - **Object-relational mapping**
  - **API with a lot of interfaces**
  - **additional tools**
  - **Possibility to work inside or out-side of a container**
  - **HQL**
    - Query by Criteria
    - Query by Example
    - polymorphic queries
  - **datastore independent**



Persistence Frameworks (cont.)

Hibernate architecture



- **Other (open source) persistence frameworks**

  see http://java-source.net/open-source/persistence

Agenda

- Java Database Connectivity
- Object Relational Mapping
- Java Persistency Architecture
- Persistence Frameworks
- **Java in DB2: Stored Procedures and User Defined Functions**

Java in DB2: Stored Procedures and User Defined Functions

1. Calling a stored procedure

- Call independent from how the stored procedure is built
- Call dependent on INPUT and OUTPUT specifications
- use JDBC CallableStatement

Example

```
CallableStatement callableStmt = connection.prepareCall("CALL STPROC( ? , ? )");
int companyId = 10023;
callableStmt.setInt(1 , companyId);
callableStmt.registerOutParameter(2, Types.CHAR);
callableStmt.execute();

String companyName = callableStmt.getString(2);
System.out.println("Name of company " + companyId + ": " + companyName);
```

STPROC runs in stored procedure address space, controlled by WLM

Java in DB2: Stored Procedures and User Defined Functions

2. Create a DB2 stored procedure (or User Defined Function)

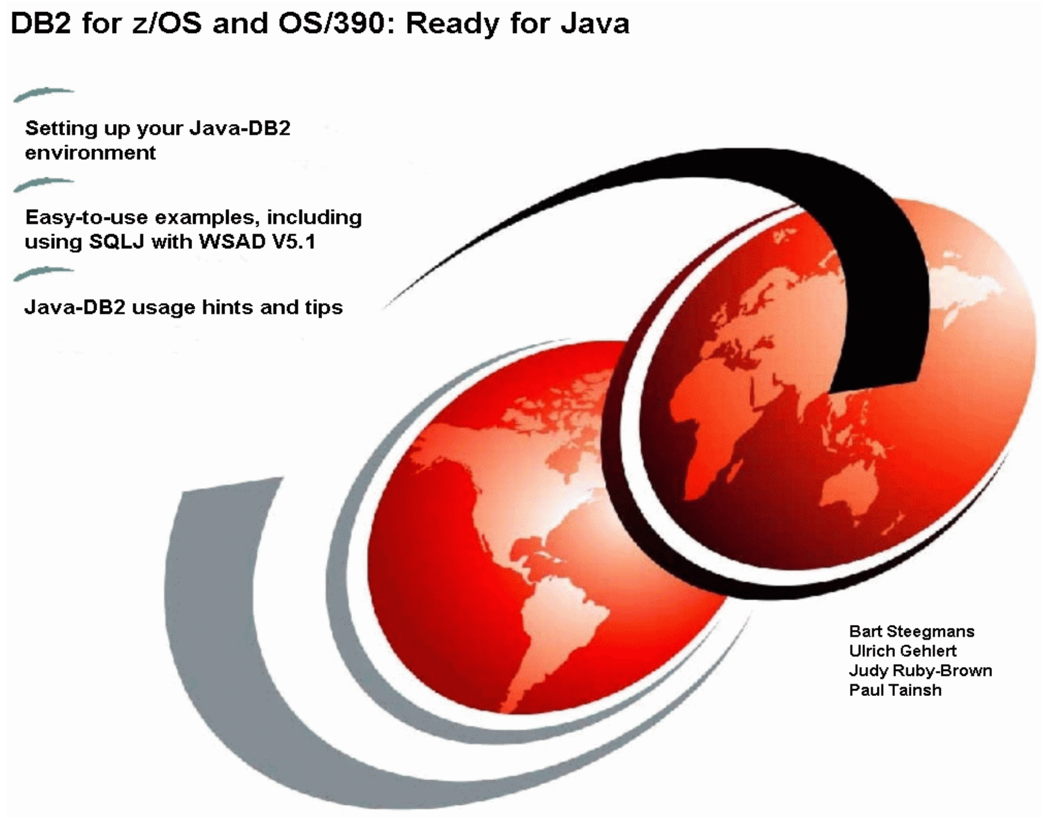
- **parameter style Java**
  - declare OUT and INOUT parameters as arrays
- **SQL access level**
  - CONTAINS SQL, READS SQL DATA or MODIFIES SQL DATA
- **use JDBC or SQLJ**
- **multiple result sets can be returned**

Prepare stored procedure (or UDF)

- **compile Java class**
- **bind package**
- **put class in CLASSPATH**
  or create JAR file
- **define JAR to DB2 (DB2_INSTALL_JAR utility)**
- **register stored procedure (or UDF) to DB2 with DDL**

Literature

- **DB2 9.1 for z/OS application programming guide and reference for Java - SC18-9842-03**
- **IBM Redbook - SG24-6435-00**


DB2 for z/OS and OS/390: Ready for Java

Setting up your Java-DB2 environment

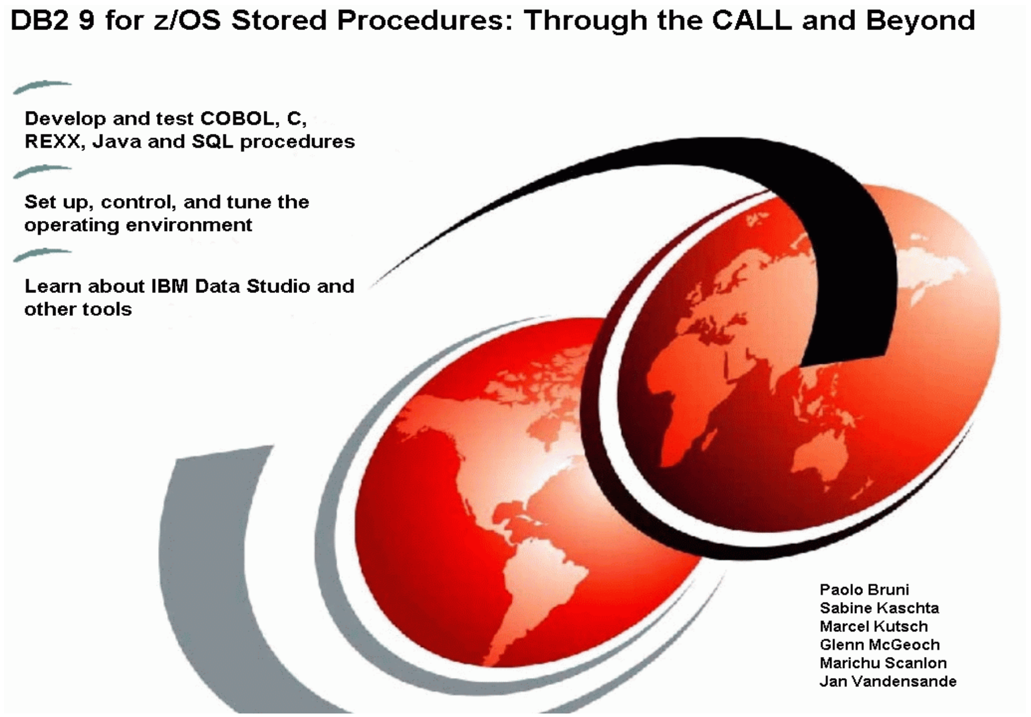
Easy-to-use examples, including using SQLJ with WSAD V5.1

Java-DB2 usage hints and tips

Bart Steegmans
Ulrich Gehlert
Judy Ruby-Brown
Paul Tainsh

Literature (cont.)

IBM Redbook - SG24-7604-00



DB2 9 for z/OS Stored Procedures: Through the CALL and Beyond

Develop and test COBOL, C, REXX, Java and SQL procedures

Set up, control, and tune the operating environment

Learn about IBM Data Studio and other tools

Paolo Bruni
Sabine Kaschta
Marcel Kutsch
Glenn McGeoch
Marichu Scanlon
Jan Vandensande

Q & A

Q&A

Thank you

Gie Indesteege

Trainer and Consultant

gindesteege@abis.be

abis

TRAINING & CONSULTING

thanks you