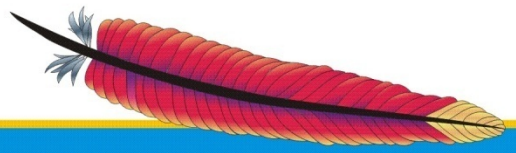
ApacheCon



Leading the Wave of Open Source

iBATIS

Apache iBatis

Getting Up to Speed

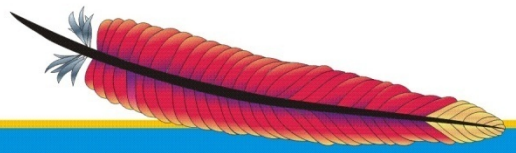
Carsten Ziegeler
cziegeler@apache.org
Day
Day Software

About Carsten Ziegeler

- Apache Software Foundation Member
  - Cocoon, Excalibur, Pluto, Felix, Incubator, Sling, Sanselan
  - PMC: Cocoon, Incubator, Portals, Felix, Excalibur (Chair)
- Senior Developer at Day Software
- Article/Book Author, Technical Reviewer
- JSR 286 spec group (Portlet API 2.0)

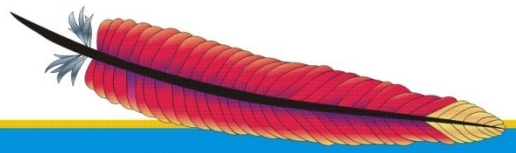
Apache iBatis

- **Quick Intro**
- Quick Start Guide
- The Data Mapping
- The Database Configuration
- Summary



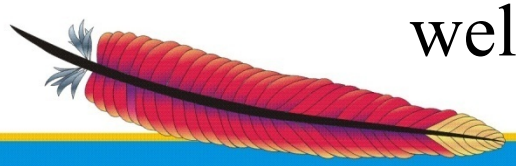
A Common Problem

- Reading from / writing to a database in Java
- Mapping of result sets to objects
- Mapping of objects to tables/columns
- Object Relational Mapping (ORM)



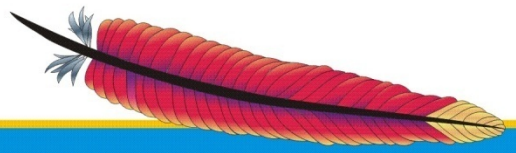
Why do I need iBatis?

- Several ORM solutions out there
  - Hibernate, Apache OpenJPA, Apache OJB
  - Very sophisticated, but "heavy" solutions
  - Usable for literally everything
  - Setting up, configuring, using takes time
- What about javax.sql/JDBC?
  - ….if you think that SQL and Java code mixes well



Why do I need iBatis?

- Apache iBatis fits in-between
  - Small library
  - Simple setup
  - Easy to use
  - Uses data maps



Use the Right Tool!

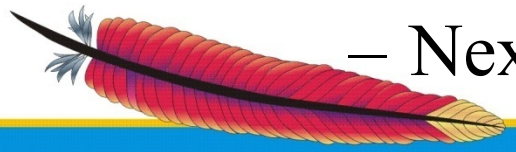
- Not every tool is suited for every use case!
  - Analyze requirements
  - Decide for the tool
  - Constantly check decision
- Apache iBatis is not suited for all ORM problems

Some Typical Scenarios for iBatis

- Reading dictionaries (like i18n translations)
- Performing complex and nested queries
  - Full SQL support
- Reading configuration
- Writing log statements
- Displaying query results

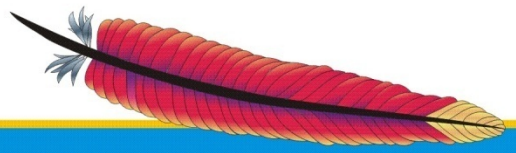
The Project

- Popular tool
  - Abstracting JDBC
  - Working at the SQL level
- Started by Clinton Begin
  - Part of Apache since 2005
  - Top level project
- Current version 2.x (> 3 years old)
  - Next major version in design discussions



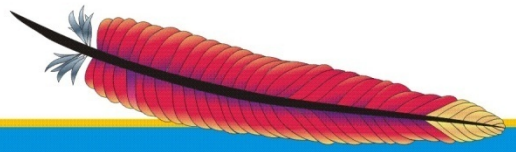
The Project

- Two frameworks
  - iBatis Data Mapper
  - iBatis Data Access Objects
- Implementations for different languages
  - Java, .Net, Ruby/Rails
- Focus on the Java Data Mapper



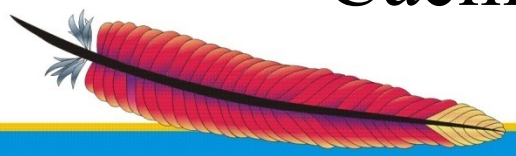
How Does it Work?

- SQL statements are defined in XML
- Statements can contain placeholders
- Placeholders are replaced on execution
- SQL query result is mapped to objects



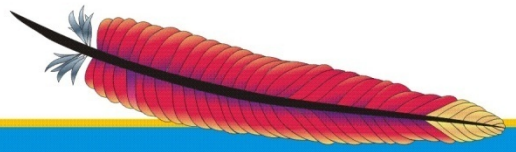
Some Advantages

- Retain the full power of SQL
  - Functions like avg, sum, count etc.
  - Mapping across tables
- Single point of configuration
- Short integration time
- Very good documentation
- Caching of query results



Caveats

- No transparent persistence
- No change detection
- Explicit mapping required
- No implicit operations (like cascaded updates, deletes etc.)

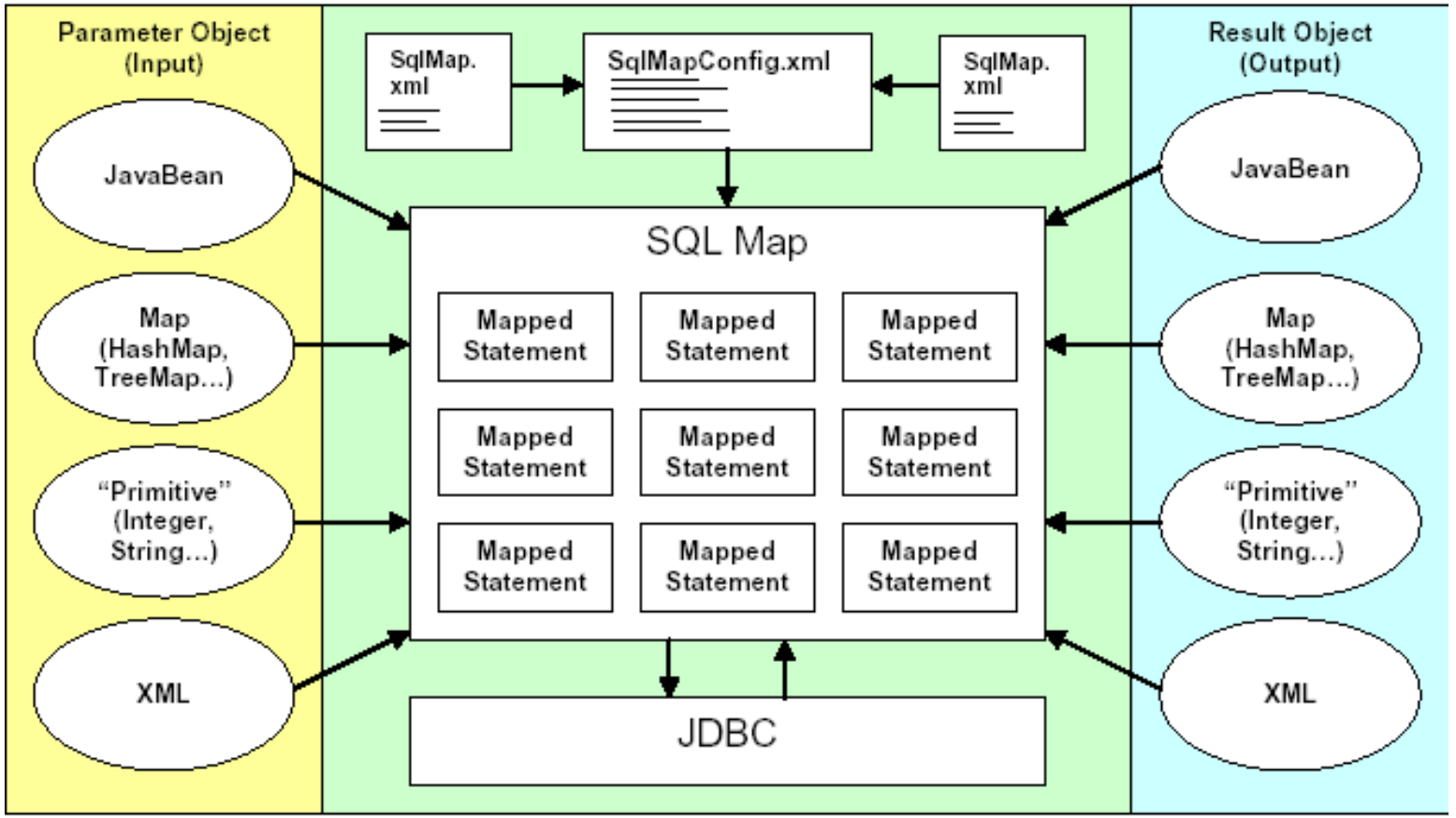


Installation

- Download and add to classpath
- Optional 3rd party jars for additional stuff
  - Refer to the documentation

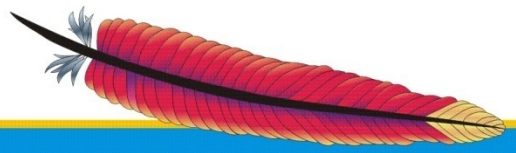
File Name	*Description*	*Required*
ibatis-common.jar	iBATIS Common Utilities	YES
ibatis-sqlmap.jar	iBATIS Data Mapper Framework	YES
ibatis-dao.jar	iBATIS Data Access Framework	NO

Data Mapper Concept



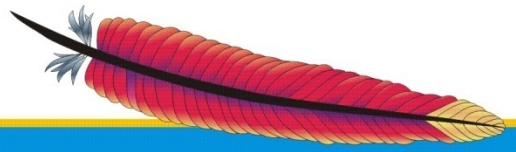
Apache iBatis

- Quick Intro
- **Quick Start Guide**
- The Data Mapping
- The Database Configuration
- Summary



Four Basic Steps

1. Write Java beans (optional)
2. Create mapping configuration
3. Specify the SQL configuration
4. Write the app code executing the statements

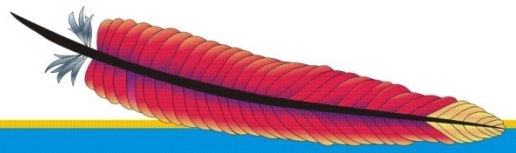


1. Write Java Beans (optional)

- Use primitive types for properties
- Alternative: maps

Person.java

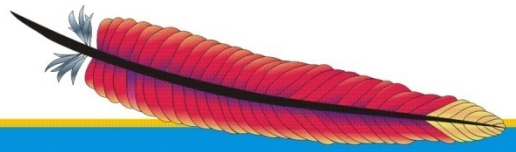
```
public class Person {
   private long personId;
   private String name;
   private java.util.Date dateOfBirth;

   // getters and setters:
   ...
```



2. Create Mapping Configuration

Person.xml

```
<sqlMap namespace="Person">

  <select id="getPerson" resultClass="samples.Person"
          parameterClass="long">
     SELECT id                  as personId
     ,      p_name              as name
     ,      date_of_birth       as dateOfBirth
     FROM person_t pers
     WHERE pers.id = #value#
  </select>
</sqlMap>
```



3. Specify the SQL Configuration

sql-map-config.xml

```
<sqlMapConfig>

  <transactionManager type="JDBC">
    <dataSource type="SIMPLE">
      <property ...
      ...
    </dataSource>
  </transactionManager>

  <sqlMap resource="samples/maps/Person.xml"/>

</sqlMapConfig>
```

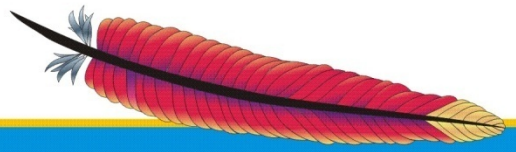
4. Write the App Code

DAO.java

```
public Person getPerson(long id) {
    Person p = null;
    try {
        // Configure iBatis
        Reader reader = Resources.getResourceAsReader
                    ("samples/sql-map-config.xml");
        SqlMapClient sqlMap =
                SqlMapClientBuilder.buildSqlMapClient(reader);

        // execute
        p = (Person)sqlMap.queryForObject("getPerson", id);
    } catch (Exception e) {
        // do something useful here
    }
    return p;
}
```

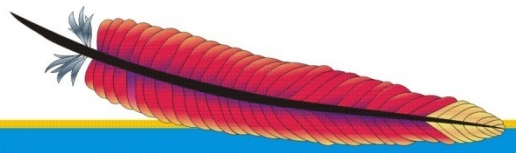
Apache iBatis

- Quick Intro
- Quick Start Guide
- **The Data Mapping**
- The Database Configuration
- Summary



The Data Mapping

- Define mapping between Java objects and SQL
- XML configuration file(s)
- iBatis offers many possibilities/features
  - The following is just an overview
  - Check out the docs
  - (Often maps are sufficient)



Mapping Elements

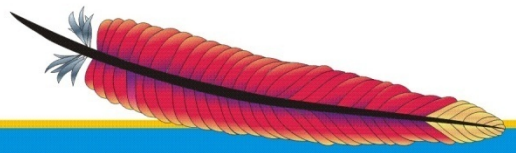
- Mapping Java beans to SQL statements
- More than one config file possible

```
<sqlMap>

  <typeAlias/> *
  <parameterMap/> *

  <resultMap/> *

  <statement/> *
  <select | insert | update | delete | procedure/> *
</sqlMap>
```

Data Mapper Concept

- Mapped statements
  - SQL
  - Parameter maps (input)
  - Result maps (output)

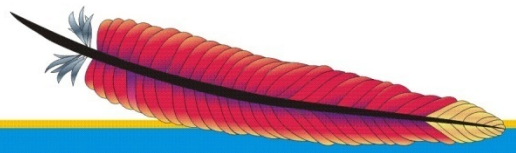
```
<select id="getAllUsers" resultMap="User">
    SELECT id, username, password, active
    FROM user_t
</select>
```



Querying the Database

- Unique identifier
- Result and parameter class
- SQL statements (it's XML!)

```
<select id="getOlderThan" resultClass="Person" parameterClass="date">
  <![CDATA[
    SELECT id as personID
           ,  ...
    FROM person_t pers
    WHERE date_of_birth > #value#
  ]]>
</select>
```



Parameters: Primitive Types

- Primitive types can directly be used for input
  - e.g. String, Integer, Date, etc.
  - #value# references the actual value

```
<statement id="getPerson" parameterClass="java.lang.Integer">
          SELECT * FROM product_t
  WHERE product_id = #value#
</statement>
```

```
sqlMapClient.queryForObject("getPerson", new Integer(1));
```

Parameters: Beans

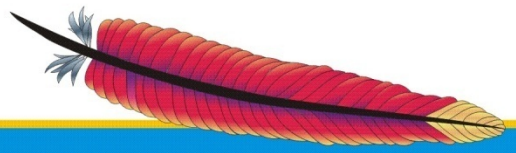
- Mapping description: bean -> input
  - Properties
  - Optional type definition: Java and JDBC
  - Optional null values
  - Optional type handler for conversion

```
<parameterMap id="parameterMapName" [class="someDomainClass"]>
   <parameter property="propertyName" [jdbcType="VARCHAR"]
       [javaType="string"] [nullValue="-1"]
       [typeHandler="someHandler"]/>
   <parameter .../>
</parameterMap>
```

Parameters: Beans

- Properties are applied in order of definition

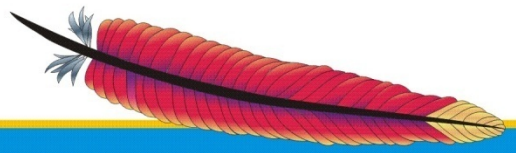
```
<parameterMap id="personMap"
              class="samples.Person">
   <parameter property="name"/>
   <parameter property="dateOfBirth"/>
   <parameter property="address"/>
</parameterMap>

<insert id="insertPerson" parameterMap="personMap">
   INSERT INTO person_t
   (name, date_of_birth, address)
   VALUES (?, ?, ?)
</insert>
```



Parameters: Inline Beans

- Less verbose form of parameter maps
  - Inline property names

```
<insert id="insertPerson" parameterMap="samples.Person">
   INSERT INTO person_t
   (name, date_of_birth, address)
   VALUES (#name#, #dateOfBirth#, #address#)
</insert>
```



Parameters: Inline Beans

- Support for types and null values

```
<insert id="insertPerson" parameterMap="samples.Person">
   INSERT INTO person_t
   (name, date_of_birth, address)
   VALUES (#name#, #dateOfBirth#, #address:VARCHAR:NO_ADDR#)
</insert>
```

Parameters: Maps

- Alternative to custom beans: maps
  - java.util.HashMap etc.
  - #key# references the value stored for *key*

```
<statement id="getAddresses" parameterClass="java.util.Map">
          SELECT * FROM addresses_t
          WHERE person_id = #personID#
          AND category_information = #category#
</statement>
```

```
params.put("personID", new Integer(1));
params.put("category", "Business");
sqlMapClient.queryForList("getAddresses", params);
```

Result Maps

- Mapping result sets to beans
  - Properties/columns
  - Optional type definition: Java and JDBC
  - Optional null values
  - Optional type handler for conversion

```
<resultMap id="resultMapName" class="someDomainClass"
           [extends="parent-resultMap"]>
  <result property="propertyName" column="COLUMN_NAME"
            [javaType="int"] [jdbcType="NUMERIC"]
            [nullVlaue="-1"]
[typeHandler="someHandler"]/>
  <result .../>
</resultMap>
```

Result Maps: Inline

- Direct mapping

```
<select id="getPerson" parameterClass="long"
                       resultClass="samples.Person">
  SELECT id AS personID
  , first_name AS firstName
  , email
  , ...
  FROM person_t pers
  WHERE pers.id = #value#
</select>
```

```
public class Person {
  private long personID;
  private String firstName;
  private String email;
  ...
```

Primitive Results

- Primitive types can directly be used
  - e.g. String, Integer, Date, etc.
  - Result map or inline

```
<resultMap id="get-result" resultClass="java.lang.Integer">
   <result property="value" column="NUMBER_OF_CUSTOMERS"/>
</resultMap>
```

```
<statement id="getPersonCount" resultClass="int"
   SELECT count(*) AS value
   FROM person_t
</statement>
```

Map Results

- Alternative to custom beans: maps
  - java.util.HashMap etc.

```
<resultMap id="result-map" resultClass="java.util.HashMap">
   <result property="id" column="ID"/>
   <result property="title" column="TITLE"/>
   <result property="speaker" column="SPEAKER_NAME"/>
</resultMap>

<select id="getAllSessions" resultMap="result-map">
 SELECT id, title, speaker_name
 FROM sessions_t
</select>
```

```
sqlmap.queryForList("getAllSessions")

{id=1, title=iBatis, speaker=Ziegeler}
{id=2, title=JCR, speaker=Ziegeler}
{id=3, title=Portals, speaker=Ziegeler}
```

Map Results Inline

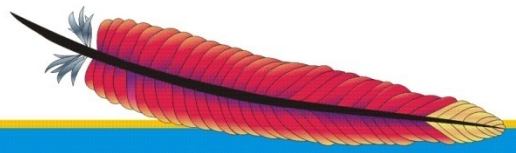
- Inline definition

```
<select id="getAllSessions" resultClass="java.util.HashMap">
 SELECT id, title, speaker_name   speaker
 FROM sessions_t
</select>
```

```
sqlmap.queryForList("getAllSessions")

{id=1, title=iBatis, speaker=Ziegeler}
{id=2, title=JCR, speaker=Ziegeler}
{id=3, title=Portals, speaker=Ziegeler}
```

Parameters and Results

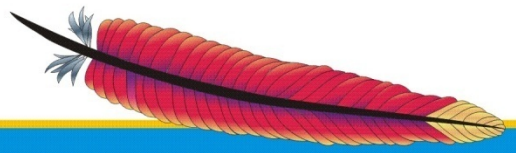
- Primitive types
- Maps
- Beans
- Inline or extra definition
- Quick Start: Maps + primitive types inline



Using the Data Mapper

- Lightweight API
  - Configuration
  - Executing Queries
  - Updating (inserts and deletes)

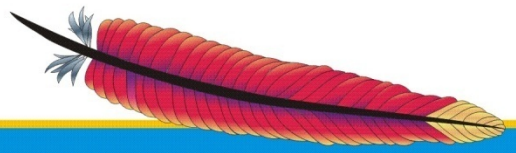
```
Reader reader = Resources.getResourceAsReader
                ("samples/sql-map-config.xml");

SqlMapClient sqlMap =
             SqlMapClientBuilder.buildSqlMapClient(reader);
```



Queries

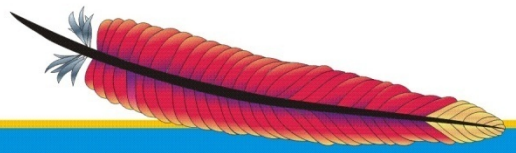
- Single objects: sqlMap.queryForObject()
- Lists: sqlMap.queryForList()
- Maps: sqlMap.queryForMap()
  - Key can be any mapped property
- Each query is run with:
  - Configured map identifier
  - Parameters (optional)

```
sqlmap.queryForList("getAllSessions")

{id=1, title=iBatis, speaker=Ziegeler}
{id=2, title=JCR, speaker=Ziegeler}
{id=3, title=Portals, speaker=Ziegeler}
```



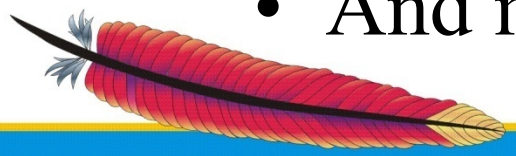
Updates, Inserts and Deletes

- sqlMap.update("updatePerson", person);
- sqlMap.insert("insertPerson", person);
- sqlMap.delete("deletePerson", 5);



Additional Features

- Mapping relationships
- Joins
- Lazy loading
- Transactions, Batches
- Stored Procedures
- Caching
- And more…



Fragments

- Define fragments for reuse
  - Avoid duplicate definitions

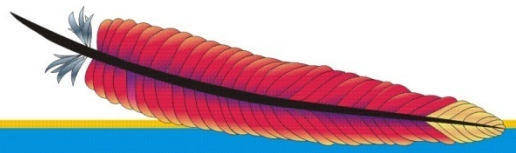
```
<sql id="base-select">
    SELECT id as personID
    ,      ...
    FROM person_t pers
</sql>

<select id="getPerson" resultClass="person" parameterClass="long">
  <include refid="base-select"/>
  WHERE pers.id = #value#
</select>
<select id="getPersons" resultClass="person">
  <include refid="base-select"/>
  ORDER BY date_of_birth   ASC
</select>
```

Dynamic SQL Statements

- Dynamic statements (if …)
- Using parameters:

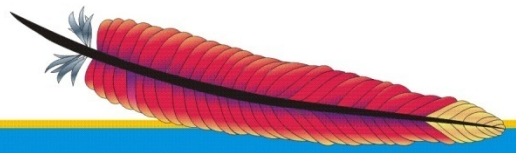
```
<select id="findPersons" resultMap="person"
        parameterClass="java.util.Map">
  SELECT id, first_name, last_name FROM person_t pers
  WHERE category = #category#
  $whereclause$
  ORDER BY $orderlist$
</select>
```

```
Map paramMap = new HashMap();
paramMap.put("category", "Business"));
paramMap.put("whereclause", "and id > 10");
paramMap.put("orderlist", "date_of_birtsh DESC");

sqlmap.queryForList("findPersons", paramMap);
```



Apache iBatis

- Quick Intro
- Quick Start Guide
- The Data Mapping
- **The Database Configuration**
- Summary



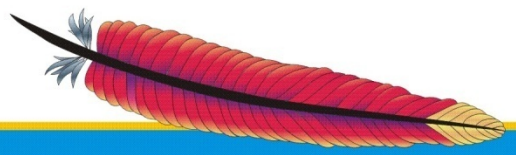
Data Base Configuration

- Central XML configuration
  - Properties
  - Data source

```
<sqlMapConfig>
  <properties/>
  <settings/>
  <typeAlias/> *

  <transactionManager>
     <dataSource/>
  </transactionManager>

  <sqlMap/> *
</sqlMapConfig>
```

Configuration Properties

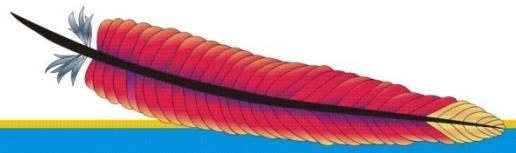
- Properties can be loaded from
  - The classpath: <properties resource=""/>
  - URL: <properties url=""/>
- Values can be referenced with Ant style syntax: ${key}



Transaction Manager

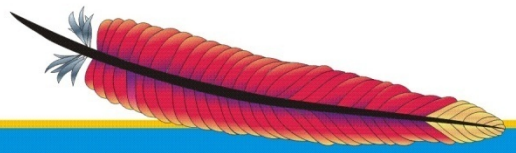
- Transaction manager
  - JDBC
  - JTA
  - EXTERNAL

```
<transactionManager type="JDBC">
     <dataSource …
```



Data Source

- Data source factory and properties
  - SIMPLE – based on iBATIS connection pooling implementation
  - DBCP – uses Jakarta DBCP
  - JNDI – retrieves data source from a JNDI context
  - Own factories possible



Data Source Sample

- Data source factory and properties

```
<dataSource type="SIMPLE">
  <property value="${driver}" name="JDBC.Driver" />
  <property value="${url}" name="JDBC.ConnectionURL" />
  <property value="${username}" name="JDBC.Username" />
  <property value="${password}" name="JDBC.Password" />
</dataSource>
```

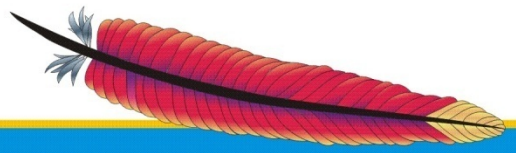
SQL Maps Configuration

- All SQL Map files must be referenced
  - Classpath
  - URL

```
<sqlMap resource="samples/Person.xml" />
<sqlMap url="file:///d:/ibatis/Person.xml" />
```

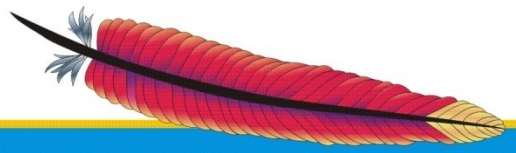
Apache iBatis

- Quick Intro
- Quick Start Guide
- The Data Mapping
- The Database Configuration
- **Summary**



Apache iBatis

- Easy and quick start
- Mapping Java beans to SQL at the SQL level
- XML configuration files for mapping and db config
- Very good documentation!
- Check it out!



ApacheCon

Thanks
Q&A

Visit our booth for infos, discussions **and jobs!**