



J-Spring

16 april 2008 Spant! - Bussum



## Speeding up your enterprise JDBC development with iBatis

Jamie Craane  
QNH Business Integration

Jamie.craane@qnh.nl



## Agenda

- **Introduction**
- Practical examples
- Advanced and undocumented features
- Testing and tool support
- Q&A



## Introducing iBatis (1)

Structured Query Language

1970

2038





## Introducing iBatis (2)

*“A layer of mappers that moves data between objects and a database while keeping them independent of each other and the mapper itself”*



## Introducing iBatis (3)



The person class...

Has to match the PERSON table



## Introducing iBatis (4)

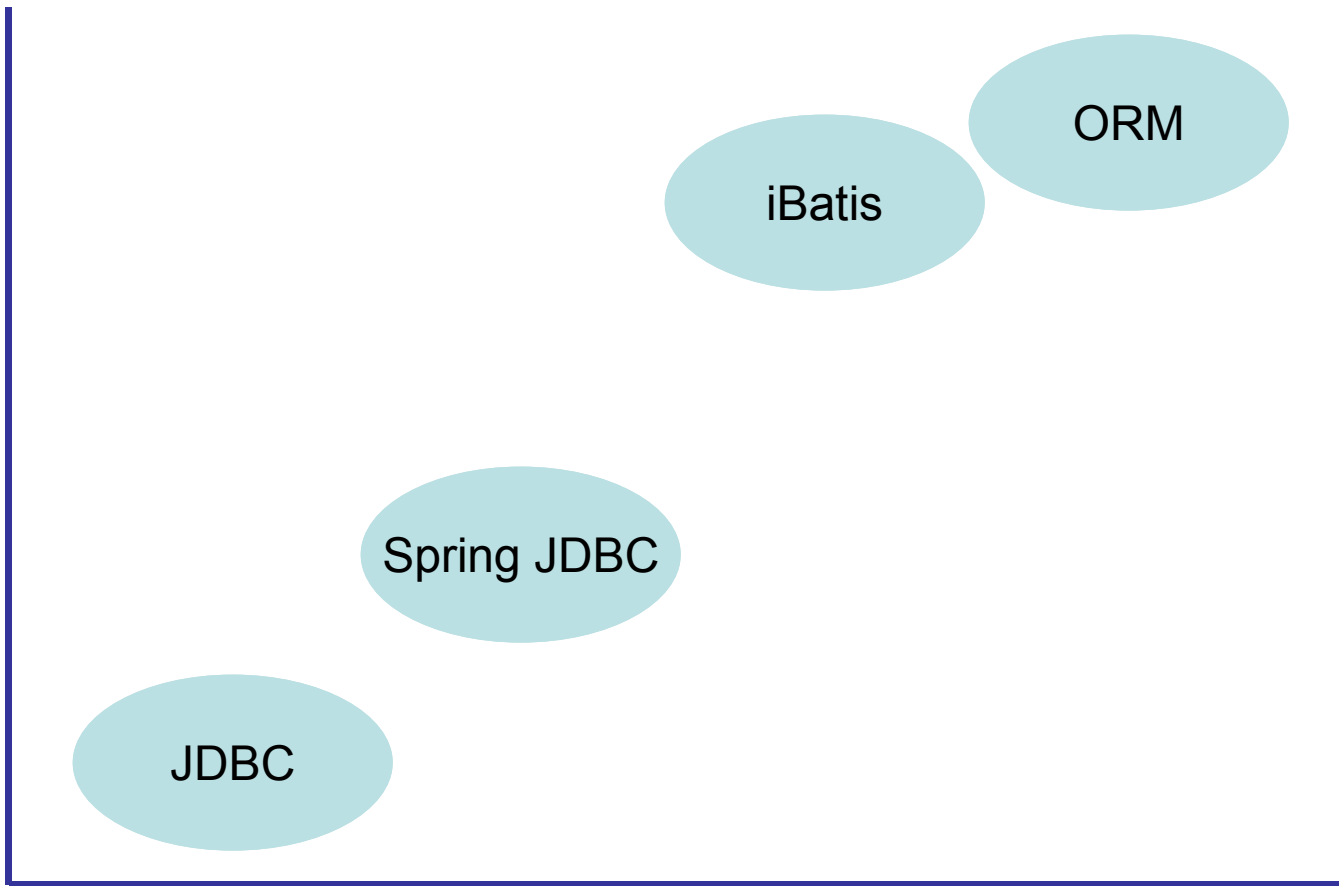
Person
-Id
-firstName
-lastName

```
Select
    ID,
    FIRST_NAME,
    LAST_NAME
FROM
    PERSON
WHERE
    ID = #identifier#
```

Person
-ID
-FIRST_NAME
-LAST_NAME



## Positioning iBatis





## A simple example

```
<sqlMap>
  <select id="getPerson" resultClass="Person"
parameterClass="integer">
    select
      id,
      first_name as firstName,
      last_name as lastName
    from
      person
    where
      id = #value#
  </select>
</sqlMap>
```

```
Person p = (Person) sqlMapClient.queryForObject("getPerson",
1);
```



## Agenda

- Introduction
- **Practical examples**
- Advanced and undocumented features
- Testing and tool support
- Q&A



## Fragments (1)

```
<select id="getProducts" resultMap="getProductsMap">
  select id, description, manufactureDate, name, price
  from product
</select>
```

```
<select id="getProduct" resultMap="getProductsMap"
parameterClass="int">
  select id, description, manufactureDate, name, price
  from product where id = #value#
</select>
```



## Fragments (2)

```
<sql id="getProductData">  
    select id, description, manufactureDate, name, price  
    from product  
</sql>
```

```
<select id="getProducts" resultMap="getProductsMap">  
    <include refid="getProductData"/>  
</select>
```

```
<select id="getProduct" resultMap="getProductsMap"  
parameterClass="int">  
    <include refid="getProductData"/> where id = #value#  
</select>
```



## Dynamic Mapped Statements

### Search Products

Description	<input type="text"/>
Manufacture date	<input type="text"/>
Price	<input type="text"/>
Name	<input type="text"/>

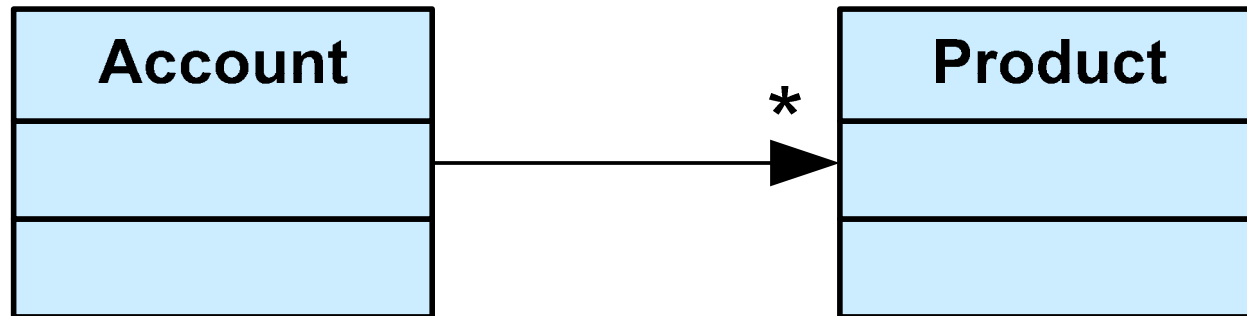


## Dynamic Mapped Statements

```
<select id="getProducts" resultMap="getProductsMap"
parameterClass="...Product">
  select id, description, manufactureDate, name, price, stock
from product
  <dynamic prepend="where">
    <isNotEmpty property="description" prepend="and">
      description = #description#
    </isNotEmpty>
    <isNotNull property="manufactureDate" prepend="and">
      manufacturedate = #manufactureDate#
    </isNotNull>
  </dynamic>
</select>
```



## Managing Relations (1)





## Managing Relations (2)

```
<select id="getAccount" parameterClass="integer"
    resultMap="getAccountMap">
    select id, name from account where id = #value#
</select>
```

```
<resultMap id="getAccountMap" class="...Account">
    <result property="products" select="getProduct" column="id"/>
</resultMap>
```

```
<select id="getProduct" parameterClass="integer"
    resultMap="getProductMap">
    select id, name, price from product_rel where account_id =
#value#
</select>
```

```
<resultMap id="getProductMap" class="...Product">
    <result property="id" column="id"/>
</resultMap>
```



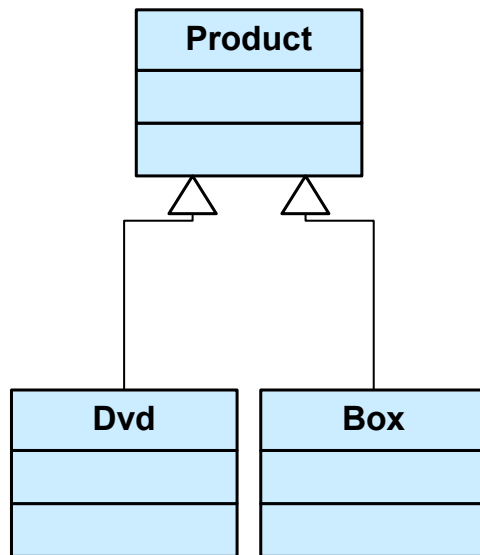
## Managing Relations (3)



But beware of the N+1 select problem!



## Inheritance (1)



ID
TYPE
PRICE
TITLE
WIDTH

```
<select id="getProduct" resultMap="productMap">
  select id, type, price, title, width from product
  where id = #value#
</select>
```



## Inheritance (2)

```
<resultMap id="productMap" class="Product">
  <result property="id" column="id"/>
  <discriminator javaType="string" column="type">
    <subMap value="dvd" resultMap="dvdResultMap"/>
    <subMap value="box" resultMap="boxResultMap"/>
  </discriminator>
</resultMap>
```

```
<resultMap id="dvdResultMap" class="Dvd" extends="productMap">
  <result property="title" column="title"/>
</resultMap>
```

```
<resultMap id="boxResultMap" class="Box" extends="productMap">
  <result property="width" column="width" nullValue="0"/>
</resultMap>
```



## Custom Type Handler (1)

- For persisting custom types
- Implement `TypeHandlerCallback`
- iBatis 2.3.1b has standard enum support



## Custom Type Handler (2)

```
public void setParameter(ParameterSetter setter, Object  
parameter) throws SQLException;
```

```
public Object getResult(ResultGetter getter) throws  
SQLException;
```

```
public Object valueOf(String s);
```

**In sqlmap-config.xml**

```
<typeHandler javaType="boolean" callback="YesNoTypeHandler"/>
```



## Caching (1)

- 4 types of build-in cache implementations:
  - MEMORY
  - LFU
  - FIFO
  - OSCACHE
- Implement CacheController for custom cache-strategy



## Caching (2)

```
<cacheModel id="productsCache" type="MEMORY" readOnly="true">  
  <flushInterval hours="1"/>  
  <flushOnExecute statement="insertProduct"/>  
  <property name="cache-size" value="10" />  
</cacheModel>
```

```
<select id="getProducts" resultMap="getProductsMap"  
  cacheModel="productsCache">  
  select id, name, description, price from product  
</select>
```



## Inserting data (1)

- iBatis can use database generated keys
- No support for optimistic locking



## Inserting data (2)

```
<insert id="insertAccount" parameterClass="Account">  
  <selectKey keyProperty="accountId" resultClass="int">  
    select nextVal('account_accountid_seq')  
  </selectKey>  
  insert into  
    account(accountId, username, password)  
  values  
    (#accountId#, #username#, #password#)  
</insert>
```

```
Account account = New Account();  
account.setUsername("bla");  
Account.setPassword("alb");
```

```
Integer accountId = (Integer)  
sqlMapClient.insert("insertAccount", account);
```



## iBatis and Spring

- Spring does have iBatis support
- SqlMapClientTemplate
- Similar as Jdbc- and HibernateTemplate



## iBatis and Spring

```
<bean id="productDao" class="ProductDaoImpl">  
    <property name="sqlMapClient" ref="sqlMapClient"/>  
</bean>
```

```
<bean id="transactionManager"  
class="...DataSourceTransactionManager">  
    <property name="dataSource" ref="dataSource"/>  
</bean>
```

```
<bean id="sqlMapClient" class="...SqlMapClientFactoryBean">  
    <property name="dataSource" ref="dataSource"/>  
    <property name="configLocation"  
value="classpath:sqlmaps/sqlmap-config.xml"/>  
</bean>
```

```
<bean id="dataSource" class="...DriverManagerDataSource"/>
```

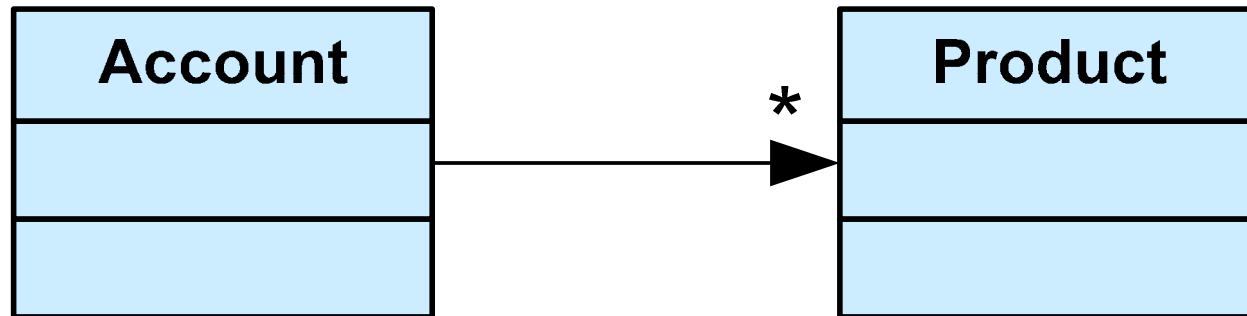


## Agenda

- Introduction
- Practical examples
- **Advanced and undocumented features**
- Testing and tool support
- Q&A



## Managing Relations (1)





## Use lazy loading

```
<sqlMapConfig>  
  <settings lazyLoadingEnabled="true"/>  
  
  ...  
</sqlMapConfig>
```



## Use a sql-join

```
<select id="getAccountWithProducts" parameterClass="integer"
resultMap="getAccountMap">
    select a.id as account_id, a.name as account_name, p.id as
product_id, p.name as product_name, p.price
from account a join product_rel p on a.id = p.account_id
where a.id = #value#
</select>

<resultMap id="getAccountMap" class="Account" groupBy="id">
    <result property="id" column="account_id"/>
    <result property="products" resultMap="getProductMap"/>
</resultMap>

<resultMap id="getProductMap" class="Product">
    <result property="id" column="product_id"/>
</resultMap>
```



## RowHandler (1)

- Powerful custom row processing
- Useful for processing large result sets
- Implement RowHandler interface



## RowHandler (2)

### Using the RowHandler

```
ProductRowHandler rh = new ProductRowHandler();  
sqlMapClient.queryWithRowHandler("getProducts", null, rh);
```

```
public class ProductRowHandler implements RowHandler {  
    ...  
  
    public void handleRow(Object o) {  
        Product product = (Product) o;  
        this.total = this.total.add(product.getPrice());  
        this.products.add(product);  
    }  
  
    ...  
}
```



## Naming conventions

- Statements `getProducts`
- Parameter maps `productsParam`
- Result maps `productsMap`
- Configuration file `sqlMapConfig.xml`
- SQL mapping files `Product.xml`



## Agenda

- Introduction
- Practical examples
- Advanced and undocumented features
- **Testing and tool support**
- Q&A



## Testing and tool support

- Test often requires a different database
- Put db-specific SQL in separate sql maps  
`Products-hsqld.xml`  
`Products-oracle.xml`
- Include the statements as fragments
- Put the Hsqldb version on the test classpath



## Testing and tool support

- IntelliJ iBatis plugin
- Abator (also as Eclipse plugin)



## Conclusions

- **iBatis “embraces” SQL**
- **iBatis does not make any assumptions about the database and object model**
- **iBatis is exceptionally well suited for existing databases**
  
- **Full fledged ORM solution may be a better choice when you have total control over your database...forever!**

.nl.  
jug



Q&A





Go to the QNH stand!

