JDBC vs iBATIS

Un caso di studio

Fabrizio Gianneschi

Atlantis S.p.A.

Cos'è iBATIS

- Framework open source per la gestione della persistenza in Java
- Disaccoppia il codice Java dall'SQL
- Elimina la necessità di usare JDBC
- NON è un object-relational mapper (come ad esempio *Hibernate*)
- Semplifica la vita agli sviluppatori ☺

10 Reasons to use iBatis

1. Works with any database that has a JDBC driver (no plugins required)
2. Configurable caching (including dependencies)
3. Local and Global transaction support and management (JTA)
4. Simple XML mapping document structure
5. Supports Map, Collection, List and Primitive Wrappers (Integer, String etc.)

10 Reasons to use iBatis

6. Supports JavaBeans classes (get/set methods)
7. Supports complex object mappings (populating lists, complex object models etc.)
8. Object models are never perfect (no changes required!)
9. Database designs are never perfect (no changes required!)
10. You already know SQL, why waste time learning something else?

Componenti

iBATIS si divide in due grandi componenti:

- **SQLMaps**
  Consente il salvataggio di oggetti su db relazionali senza utilizzare JDBC ed SQL nelle classi Java
- **DAO**
  E' un layer che nasconde i dettagli della persistenza fornendo comunque un'interfaccia condivisa col resto dell'applicazione

SQLMaps

SQLMaps in pillole

- Si basa su file XML di configurazione, in cui si danno dei *nomi* alle query utilizzate nell'applicazione
- I parametri d'ingresso delle query possono essere passati tramite classi semplici (Integer, String…) o HashMap
- In casi più complessi possono essere utilizzate direttamente classi applicative (es. VO, DTO)
- Situazioni particolari sul DB sono gestite attraverso delle mappature colonna/proprietà
- Idem per i risultati, disponibili come singolo oggetto o collezioni

Mapping semplice

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

…

<sqlMap namespace="Persona">

    <select id="findPersonaById“ resultClass =
          “com.gruppoatlantis.www.verbali.dto.PersonaDTO“>

        SELECT *
        FROM persone_view
        WHERE idpersona=#value#

    </select>

…
```

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

…

<sqlMap namespace="Persona">

  <resultMap id="personaMap"
      class="com.gruppoatlantis.www.verbali.dto.PersonaDTO">

    <result property="id" column="idpersona"/>
    <result property="cognome" column="cognome"/>
    <result property="nome" column="nome"/>
    <result property="matricola" column="matricola"/>
    <result property="livello" column="eta" jdbcType="INTEGER"
            nullValue="0" />
  </resultMap>

    <select id="findPersonaById" resultMap="personaMap">

       SELECT *
       FROM persone_view
       WHERE idpersona=#value#

    </select>

…
```

Mapping più complesso

SQLMaps in pillole

- Da Java, per eseguire una query basta utilizzare un'istanza della classe
  `com.ibatis.sqlmap.client.SqlMapClient`
- Si utilizzano quasi sempre i metodi
  `queryForList` e **`queryForObject`** specificando il nome della query da eseguire e un oggetto per i parametri.

  Ad esempio:

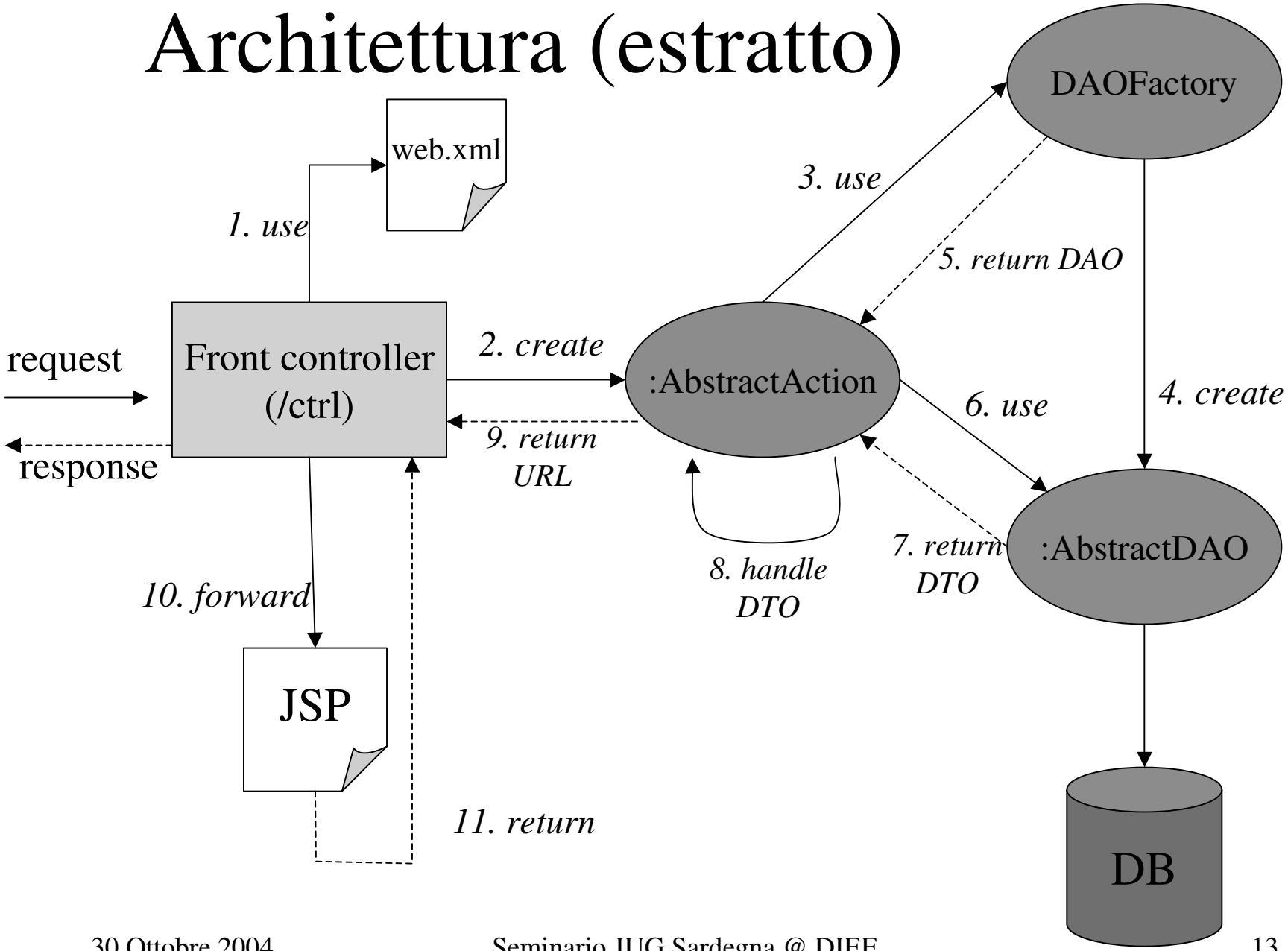
```
SQLMapClient sqlMap = …;
String queryName = "findPersonaById";
Integer id = new Integer(15);
PersonaDTO p = (PersonaDTO) sqlMap.queryForObject(queryName, id);
```

SQLMaps case study

Scenario

- Applicazione intranet realizzata in tecnologia **J2EE** (JSP, JSTL, Servlet), in esecuzione da Gennaio 2004 per la gestione dei verbali delle riunioni aziendali
- Accesso ai dati tramite DAO/**JDBC**
- Database PostgreSQL (14 tabelle, 3 viste, 9 sequence, 1 funzione)

Architettura (estratto)



LOC prima di iBATIS

AbstractDAO	26
AttivitaPostgresDAO	524
CategoriaPostgresDAO	151
MascheraDAO	221
OdgPostgresDAO	131
PersonaPostgresDAO	388
ProgettoPostgresDAO	112
RiunionePostgresDAO	514
RuoloPostgresDAO	75
StatoAttivitaPostgresDAO	124
TOTALE	**2266**

Metodo Java con JDBC (1/3)

```
public List findXyZ(String par1) {
   Connection conn = null;
   List retColl = null;
   try {
      conn = getConnection(); //implemented on superclass
      String query = "SELECT * FROM table WHERE a=?;";

      PreparedStatement ps = conn.prepareStatement(
          query, ResultSet.TYPE_SCROLL_INSENSITIVE,
          ResultSet.CONCUR_READ_ONLY);

      ps.setString(1 , par1); // sets query parameters

      ResultSet rs = ps.executeQuery();
/(… continua …)
```

Metodo Java con JDBC (2/3)

```
        retColl = new ArrayList(); // result collection

        while ( rs.next() ) {  // iterate over query results

            // an object for each row
            MyClass obj = new MyClass();
            obj.setField1( rs.getInt(1) );
            obj.setField2( rs.getDate(2) );
            …
            obj.setFieldN( …. );
            retColl.add(obj);
        }
        rs.close();  // release resources
        ps.close();
        return retColl;
//(… continua …)
```

Metodo Java con JDBC (3/3)

```
        //...

        } catch (SQLException ex) {
           // error handling
           ...
        } finally {
            // implemented on superclass
            closeConnection(conn);
        }

} // End of method
```

Metodo Java con iBATIS

```
public List findXyZ(String par1) {
    try{

        return sqlMap.queryForList("findXyZ",par1);

    }catch(SQLException ex){
       // error handling
       ...
    }
} // End of method
```

Confronto JDBC-iBATIS

JDBC

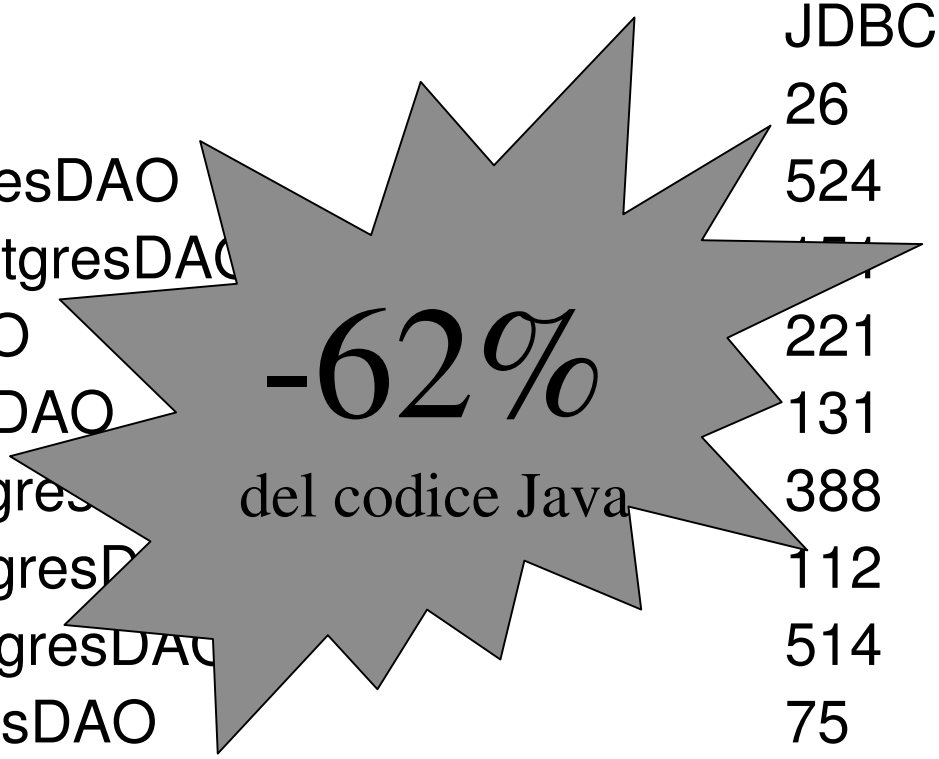
1. Ottenere la connessione al db
2. Creare lo statement per la query
3. Settare i parametri
4. Eseguire lo statement
5. Creare la collezione dei risultati
6. Per ogni riga ricevuta:
    1. Creare un oggetto
    2. Settare i campi dell'oggetto con le colonne della riga
    3. Aggiungere l'oggetto alla collezione
7. Rilasciare le risorse
8. Chiudere la connessione
9. Restituire la collezione

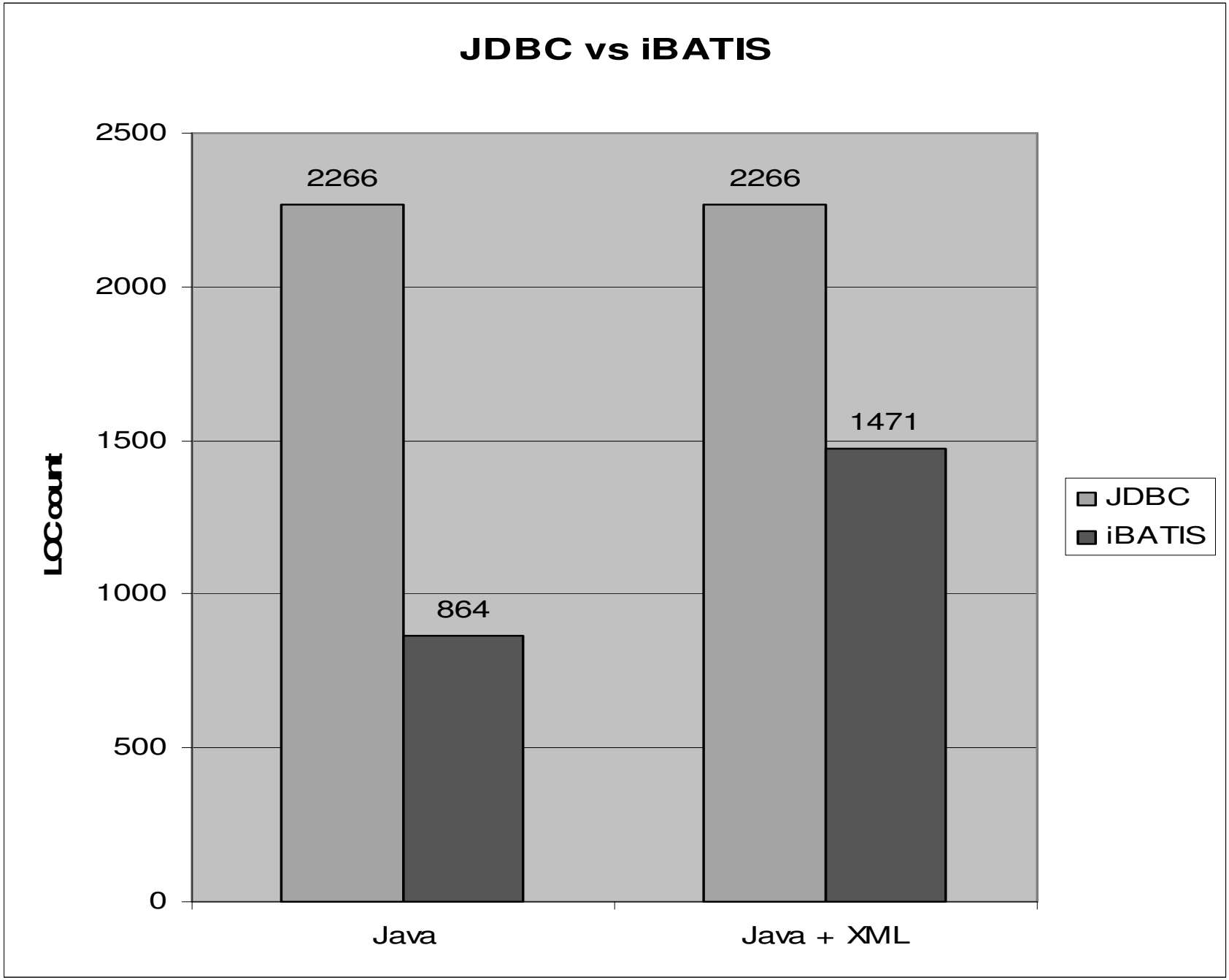
iBATIS

1. Ottenere l'oggetto SqlMap
    a) Preparare eventuali parametri complessi (opzionale)
2. Invocare la query passandole i parametri
3. Restituire la collezione dei risultati

LOC count

	JDBC	iBATIS
AbstractDAO	26	0
AttivitaPostgresDAO	524	199
CategoriaPostgresDAO	[illegible]	66
MascheraDAO	221	73
OdgPostgresDAO	131	51
PersonaPostgres[illegible]	388	165
ProgettoPostgresD[illegible]	112	26
RiunionePostgresDAO	514	202
RuoloPostgresDAO	75	34
StatoAttivitaPostgresDAO	124	48
TOTALE	**2266**	**864**




JDBC vs iBATIS
2500
2000
1500
1000
500
0
LOC count
2266
864
2266
1471
Java
Java + XML
JDBC
iBATIS

Performance

- Comparabile con l'approccio classico abilitando alcune opzioni
  (http://raibledesigns.com/page/rd?anchor=persistence_options_with_existing_sql)
- Possibilità di Caching
- Lazy loading degli oggetti
- E' possibile invocare stored procedure
- L'SQL è sempre visibile per ulteriori ottimizzazioni

Conclusioni

- L'introduzione di iBATIS nel progetto in esame è risultata estremamente vantaggiosa in termini di LOC e chiarezza generale
- La logica di business non ha richiesto praticamente nessuna modifica, così come le query SQL
- Il tempo di apprendimento è stato molto breve (< 1 gg per il primo DAO funzionante)

Confronto iBATIS - Hibernate

	iBATIS	Hibernate
Consigliabile all'inizio del progetto	👍 👍	👍
Consigliabile a progetto avviato	👍 👍	👎
OR mapping	👍	👍 👍 👍
Semplicità	👍 👍	👍
Funzionalità	👍 👍	👍 👍 👍
Documentazione	👍	👍 👍
Community	👍	👍 👍

Riferimenti

Ufficiali:

- http://www.ibatis.com
- http://www.ibatis.com/common/sqlmaps.html
- http://opensource.thoughtworks.com/projects/ibatis.jsp

Articoli e impressioni:

- http://www.sixty4bit.com/mt/archives/2003/12/17/persistence_layer_review.html
- http://www.developer.com/db/article.php/3346301