**iBATIS vs Hibernate vs JPA**

**iBATIS :**

* Maps the ResultSet from JDBC API to your POJO Objets.
* Makes use of SQL which could be database dependent
* Works very well for stored procedures, works very well for reporting applications, etc
* Simpler, Faster development time
* Flexible
* Enables the data model and the object model to be independent of each other.

**Hibernate:**

* Maps your Java POJO objects to the Database tables
* Makes use of HQL which is relatively independent of databases and it is easier to change db in Hibernate.
* If you are using stored procedures, well you can do it in Hibernate but it is little difficult in comparision of iBATIS
* Generates SQL for you which means you don't spend time on SQL
* Provides much more advance cache
* Highly scalable
* HQL also supports many advanced features of pagination and dynamic profiling that SQL has never supported.

**JPA:**

* JPA uses metadata annotations and/or XML descriptor files to configure the mapping between Java objects in the application domain and tables in the relational database.
* Defines an SQL-like query language, JPQL (Java Persistence Query Language), which is different from EJB-QL (EJB Query Language), the language used by entity beans.
* Is the standard object-relational mapping and persistence management interface for the Java EE 5 platform
* Entity Class Annotation (@Entity, @Table, @Column, @NamedQuery)
* JPA also supports SQL through the createNativeQuery() method of the EntityManager.
* Hibernate is one of the most popular frameworks that implements JPA.

use iBATIS if:

* You need complete control of the SQL or the SQL queries need to be fine-tuned.
* your environment is driven by relational data model.
* you have to work existing and complex schema's.

use Hibernate if:

* Your environment is driven by object model and wants generates SQL automatically.

When not to use JPA:

* Caching, this is not clearly defined in JPA but is well supported by Hibernate.
* JPA is defined to work with relational databases only. If your persistence solution needs to be extended to other types of data stores, like XML databases, then JPA is not the answer to your persistence problem.

**Persistence solutions compared**

|  |  |  |  |
| --- | --- | --- | --- |
| **Features** | **iBATIS** | **Hibernate** | **JPA** |
| **Simplicity** | Best | Good | Good |
| **Complete ORM solution** | Average | Best | Best |
| **Adaptability to data model changes** | Good | Average | Average |
| **Complexity** | Best | Average | Average |
| **Dependence on SQL** | Good | Average | Average |
| **Performance** | Best | Best | N/A \* |
| **Portability across different relational databases** | Average | Best | N/A \* |
| **Portability to non-Java platforms** | Best | Good | Not Supported |
| **Community support and documentation** | Average | Good | Good |

**Spring support**

**iBatis**

**extends**org.springframework.orm.ibatis.support.SqlMapClientDaoSupport

"iBATIS ,Hibernate ,JPA" These 3 are diffeerent technologies like iBATIS for SQL, Hibernate for java, JPA for Caching.. You haven choosen the 3 effective technologies,,if you find out any source for these 3.. Share with us...