let's move
the java world

A Performance Comparison of JPA Providers

Patrycja Wegrzynowicz

About Me

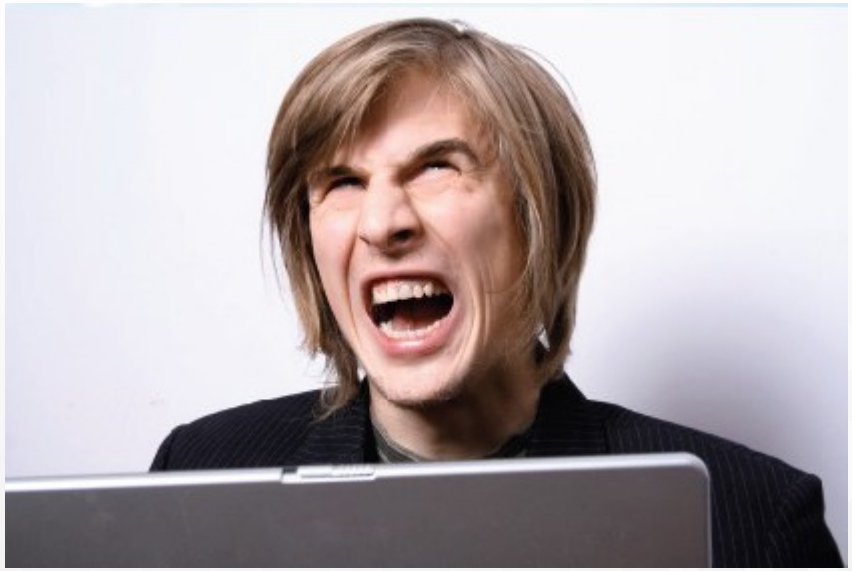
- 10+ years of professional experience as software developer, architect, and head of software R&D
- PhD in Computer Science
  - Patterns and anti-patterns, code analysis, language semantics, compiler design
- Speaker at JavaOne, Devoxx, OOPSLA, JavaZone, TheServerSide Symposium, Jazoon, others
- CTO of Yonita, Inc.
  - Bridge the gap between the industry and the academia
  - Automated detection and refactoring of software defects
  - Security, performance, concurrency, databases
- Twitter: @yonlabs

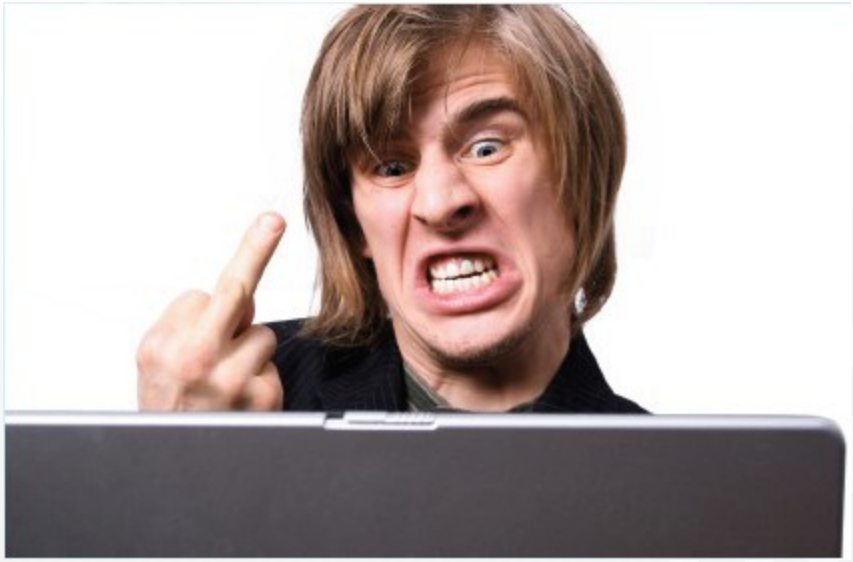
Today

- JPA Providers
  - Hibernate
  - EclipseLink
  - OpenJPA
  - DataNucleus
- Performance
  - Generated queries (logic)
  - Code performance (implementation)

Disclaimer

I do think those JPA Providers are great tools!





JPA Providers

JPA Provides Usage Poll

- Hibernate?
- EclipseLink?
- OpenJPA?
- DataNucleus?

JPA Providers in Comparison

- Hibernate
  - hibernate 4.1.2.Final
- EclipseLink
  - eclipselink 2.3.2
- OpenJPA
  - openjpa-all 2.2.0
  - openjpa-maven-plugin 1.2
- DataNucleus
  - datanucleus-core, -api-jpa 3.0.10
  - datanucleus-rbms, -api-jdo 3.0.8
  - maven-datanucleus-plugin 3.0.2

JPA Providers Test Environment

- Mac OSX 10.6.8
- 1.86Ghz Core 2 DUO
- 2Gb RAM
- JDK 1.6.0_26-b023-384
- MySQL 5.5.16 Community Server

Simple Operations – User with 2 fields

Inserts

	Hibernate	EclipseLink	OpenJPA	DataNucleus
1000 inserts	2621ms	1936ms	2383	4400
1000 finds	1297ms	577ms	1584	363
1000 updates	2039ms	2396ms	1485	314
1000 deletes	2279ms	2687ms	3382	2698

Hibernate Puzzle: Heads of Hydra

Heads of Hydra

```
@Entity
public class Hydra {
	private Long id;

	private List<Head> heads = new ArrayList<Head>();


	@Id @GeneratedValue

	public Long getId() {...}

	protected void setId() {...}

	@OneToMany(cascade=CascadeType.ALL)

	public List<Head> getHeads() {

			return Collections.unmodifiableList(heads);

	}

	protected void setHeads() {...}

}

// new EntityManager and new transaction: creates and persists the hydra with 3 heads

// new EntityManager and new transaction
Hydra found = em.find(Hydra.class, hydra.getId());
```

How Many Queries in 2nd Tx?

(a) 1 select
(b) 2 selects
(c) 1+3 selects
(d) 2 selects, 1 delete, 3 inserts
(e) None of the above

During commit hibernate checks whether the collection property is dirty (needs to be re-created) by comparing Java identities (object references).

IllegalArgumentException is wrapped by PersistentException.

Another Look

```
@Entity
public class Hydra {
	private Long id;

	private List<Head> heads = new ArrayList<Head>();


	@Id @GeneratedValue

	public Long getId() {...}

	protected void setId() {...}

	@OneToMany(cascade=CascadeType.ALL)

	public List<Head> getHeads() {

			return Collections.unmodifiableList(heads);

	}

	protected void setHeads() {...}

}
// new EntityManager and new transaction: creates and persists the hydra with 3 heads

// new EntityManager and new transaction
// during find only 1 select (hydra)
Hydra found = em.find(Hydra.class, hydra.getId());
// during commit 1 select (heads),1 delete (heads),3 inserts (heads)
```

Head of Hydra Comparison

Hibernate	EclipseLink	OpenJPA	Datanucleus
2 selects 1 delete 3 inserts	Both TXs in the same JVM: none 2nd TX in a new JVM: 1 select	IllegalAccessError (not able to enhance the class, in both modes: runtime & build-time enhancing)	Both TXs in the same JVM: none 2nd TX in a new JVM: 1 select

Heads of Hydra Revisited

```
@Entity
public class Hydra {
    @Id @GeneratedValue

    private Long id;

    @OneToMany(cascade=CascadeType.ALL)

    private List<Head> heads = new ArrayList<Head>();


    public Long getId() {...}

    protected void setId() {...}

    public List<Head> getHeads() { return heads; }

    public void setHeads(List<Head> heads) { this.heads = heads; }

}

// new EntityManager and new transaction: creates and persists the hydra with 3 heads

// new EntityManager and new transaction
Hydra found = em.find(Hydra.class, hydra.getId());
found.setHeads(new ArrayList<Head>(found.getHeads());
```

Head of Hydra Revisited Comparison

Hibernate	EclipseLink	OpenJPA	Datanucleus
2 selects 1 delete 3 inserts	Both TXs in the same JVM: none 2nd TX in a new JVM: **2 selects**	2 selects 1 delete 3 inserts	Both TXs in the same JVM: **1 select** 2nd TX in a new JVM: **2 selects**

Execution Times – 1000x

	Hibernate	EclipseLink	OpenJPA	Datanucleus
SQL logging	8170ms	2858ms	14863ms	10657ms (inner joins)
No logging	4310ms	676ms	8786ms	4958ms

EclipseLink Puzzle: Plant a Tree

Plant a Tree

```
@Entity
public class Forest {
        @Id @GeneratedValue

        private Long id;

        @OneToMany

        private Collection<Tree> rees = new HashSet<Tree>();


        public void plantTree(Tree tree) {

                        return trees.add(tree);

        }

}

// new EntityManager and new transaction: creates and persists a forest with 10.000 trees

// new EntityManager and new transaction
Tree tree = new Tree(“oak”);
em.persist(tree);
Forest forest = em.find(Forest.class, id);
forest.plantTree(tree);
```

How Many Queries in 2nd Tx?

(a) 1 select, 2 inserts
(b) 2 selects, 2 inserts
(c) 2 selects, 1 delete, 10.000+2 inserts
(d) 2 selects, 10.000 deletes, 10.000+2 inserts
(e) none of the above

```
@Entity
public class Forest {
        @Id @GeneratedValue

        private Long id;

        @OneToMany

        private Collection<Tree> trees = new HashSet<Tree>();


        public void plantTree(Tree tree) {

                        return trees.add(tree);

        }

}

// new EntityManager and new transaction: creates and persists a forest with 10.000 trees

// new EntityManager and new transaction
Tree tree = new Tree("oak");
em.persist(tree);
Forest forest = em.find(Forest.class, id);
forest.plantTree(tree);
```

How Many Queries in 2nd Tx?

(a) 1 select, 2 inserts

(b) 2 selects, 2 inserts

(c) 2 selects, 1 delete, 10.000+2 inserts

(d) 2 selects, 10.000 deletes, 10.000+2 inserts

(e) None of the above

Plant a Tree Comparison

Hibernate	EclipseLink	OpenJPA	Datanucleus
2 selects 1 delete 10.000+2 inserts	Both TXs in the same JVM: 2 inserts Different JVMs: 2 selects 2 inserts	3 selects 1 update (seq table) 2 inserts	3 selects 1 update (seq table) 2 inserts

Execution Times

	Hibernate	EclipseLink	OpenJPA	Datanucleus
SQL logging	4202ms	644ms (1017ms)	1391ms	264ms
No logging	4120ms	605ms (970ms)	1290ms	171ms

Plant a Tree Revisited

```
@Entity
public class Orchard {
        @Id @GeneratedValue

        private Long id;

        @OneToMany

        private List<Tree> trees = new ArrayList<Tree>();


        public void plantTree(Tree tree) {

                        return trees.add(tree);

        }

}

// creates and persists a forest with 10.000 trees

// new EntityManager and new transaction
Tree tree = new Tree("apple tree");
em.persist(tree);
Orchard orchard = em.find(Orchard.class, id);
orchard.plantTree(tree);
```

STILL BAG SEMANTIC

Use OrderColumn or IndexColumn for list semantic.

Plant a Tree Comparison

Hibernate	EclipseLink	OpenJPA	Datanucleus
2 selects 1 delete 10.000+2 inserts	Both TXs in the same JVM: 2 inserts Different JVMs: 2 selects 2 inserts	3 selects 1 update (seq table) 2 inserts	3 selects 1 update (seq table) 2 inserts

@OrderColumn, insert last

Hibernate	EclipseLink	OpenJPA	Datanucleus
2 selects 2 inserts	Both TXs in the same JVM: 2 inserts Different JVMs: 2 selects 2 inserts	3 selects 1 update (seq table) 2 inserts	2 selects 1 update (seq table) 2 inserts

Execution Times

	Hibernate	EclipseLink	OpenJPA	Datanucleus
SQL logging	1747ms	797ms (1337ms)	1516ms	625ms
No logging	1583ms	687ms (1240ms)	1497ms	621ms

@OrderColumn, insert first

Hibernate	EclipseLink	OpenJPA	Datanucleus
2 selects 2 inserts 10.000 updates	Both TXs in the same JVM: 2 inserts 10.000 updates Different JVMs: 2 selects 2 inserts 10.000 updates	2 selects 1 delete 10.000+2 inserts	2 selects 1 update (seq table) 2 inserts

Execution Times

	Hibernate	EclipseLink	OpenJPA	Datanucleus
SQL logging	4909ms	4968ms (5957ms)	5417ms	737ms
No logging	4800ms	3831ms (4580ms)	4783ms	690ms

OneToMany Mapping

Semantic	Java Type	Annotation
Bag semantic	java.util.Collection java.util.List	@ElementCollection || @OneToMany || @ManyToMany
Set semantic	java.util.Set	@ElementCollection || @OneToMany || @ManyToMany
List semantic	java.util.List	(@ElementCollection || @OneToMany || @ManyToMany) && (@OrderColumn || @IndexColumn)

Plant a Tree

```
@Entity
public class Forest {
        @Id @GeneratedValue

        private Long id;

        @OneToMany

        private Set<Tree> trees = new HashSet<Tree>();


        public void plantTree(Tree tree) {

                        return trees.add(tree);

        }

}

// new EntityManager and new transaction: creates and persists a forest with 10.000 trees

// new EntityManager and new transaction
Tree tree = new Tree("oak");
em.persist(tree);
Forest forest = em.find(Forest.class, id);
forest.plantTree(tree);
```

1. Collection elements loaded into memory
2. Possibly unnecessary queries
3. Transaction and locking schema problems: version, optimistic locking

Set - Queries

Hibernate	EclipseLink	OpenJPA	Datanucleus
2 selects 2 inserts	Both TXs in the same JVM: 2 inserts Different JVMs: 2 selects 2 inserts	3 selects 1 update (seq) 2 inserts	1 select 1 update (seq) 2 inserts

Set - Execution Times

	Hibernate	EclipseLink	OpenJPA	Datanucleus
SQL logging	1753ms	661ms (1301ms)	1676ms	704ms
No logging	1727ms	627ms (1233ms)	1510ms	658ms

Conclusion

- EclipseLink
  - Reasonable policies
  - Efficient implementation
- Hibernate
  - Needs smarter policies for collections
- Datanucleus
  - Very smart plicies for collections!
- OpenJPA
  - Moderate

Contact

- email: patrycja@yonita.com
- twitter: @yonlabs
- http://www.yonita.com