PRESENTED BY RED HAT

LEARN. NETWORK.
EXPERIENCE OPEN SOURCE.

www.theredhatsummit.com

Hibernate OGM: JPA on Infinispan

When PaaS Persistence Meets Java EE

Emmanuel Bernard
Platform Architect (but still does stuff)
JBoss by Red Hat

Sanne Grinovero
Software Engineer (but challenges architects)
JBoss by Red Hat

Before you leave

- Hibernate OGM
  - JPA for NoSQL
    - in particular key / value stores (Infinispan initially)
  - reuse mature projects
  - keep the good of the relational model
  - does queries too (gradual ramp up)
  - still early in the project

Who are we?

- Work for JBoss by Red Hat
- Emmanuel Bernard
  - Hibernate team
  - JCP
  - Les Cast Codeurs
  - Author
  - Links
    - emmanuelbernard.com
    - @emmanuelbernard
- Sanne Grinovero
  - Hibernate team
  - Infinispan team
  - Links
    - in.relation.to/Bloggers/Sanne
    - @sannegrinovero

(No)SQL tour

Relational databases

- Brought peace and order for 30 years
- Data structure abstraction
- Safety net
  - transaction, referential integrity, (simple) types
- Proven usefulness
  - tuning, backup, resilience

Relational databases

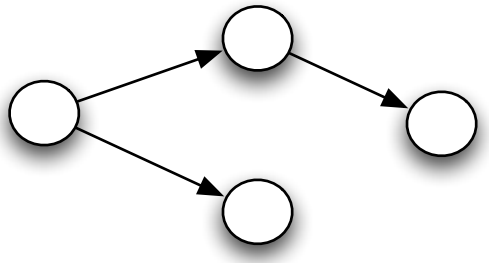
- (Some) limitations:
  - plan for scale is hard
  - data model changes are painful
- New needs
  - limitless data for later analysis
  - risk of being successful
  - Cloud

NoSQL alternatives

- ¬SQL
  - it's a big set :)
  - is 'new File("EBernardDB");' a NoSQL solution?
- Very different Goals
  - large dataset
  - high availability
  - low latency / higher throughput
- Specific data access pattern

NoSQL families

- Graph oriented databases
- Key / value stores
- Document based stores
- Column based



key	value
123	Address@23
126	“Booya”

1	Things	A	foo	B	bar	C	baz	
2	Things	C	bam	E	coh	People	A	Emmanuel
3	Languages	A	C	B	Java	C	Ceylon	

```
{ "user" : {
  "id": "124",
  "name": "Emmanuel",
  "addresses" : [
    { "city": "Paris", "country": "France" },
    { "city": "Atlanta", "country": "USA" }
  ]
}
```


Flexibility at a cost

- Programming model
  - one API per product :(
  - query (Map Reduce, specific DSL, ...)
  - no schema => app driven schema
- Physical data structure transpires
- Transaction / durability / consistency

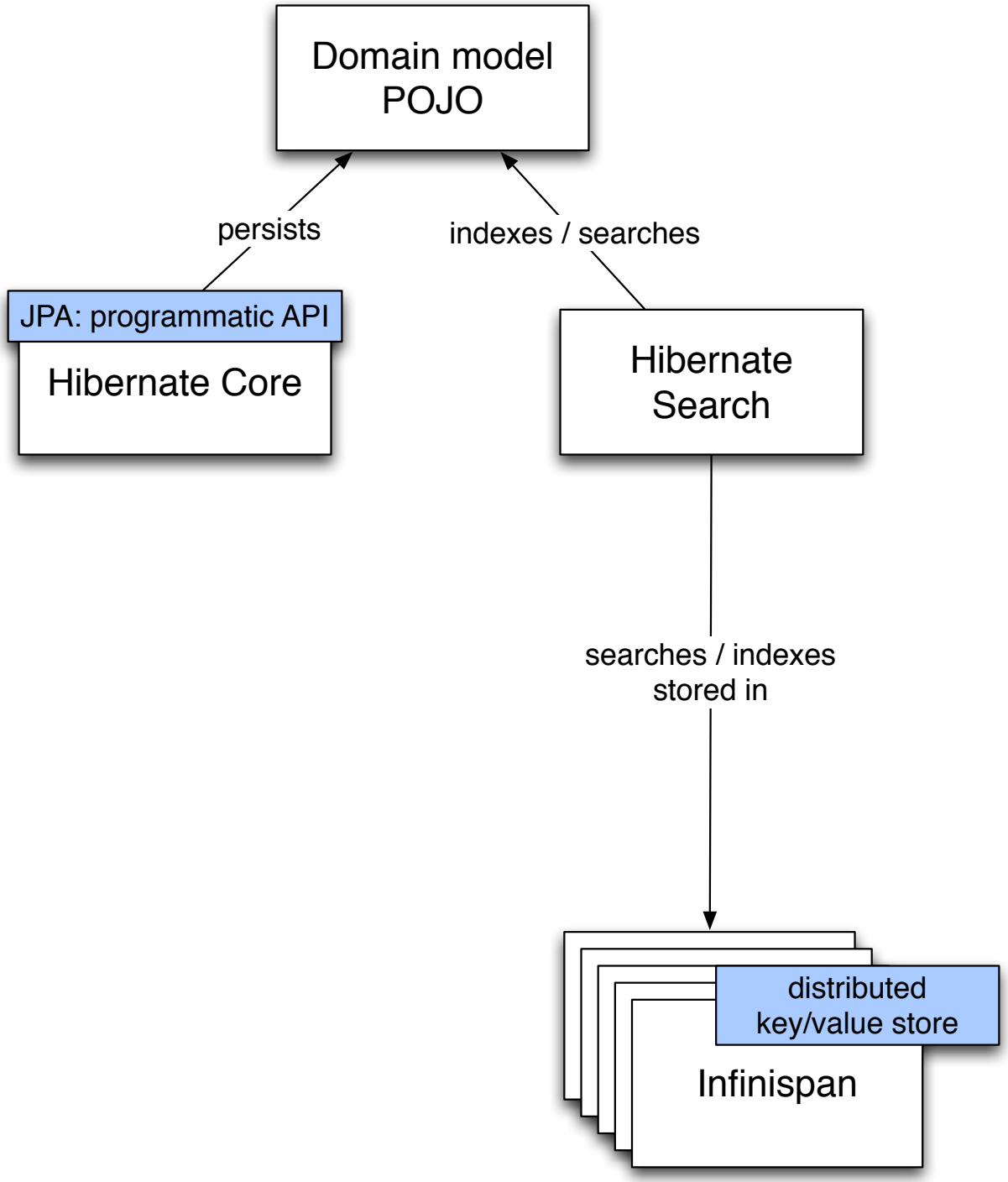
JPA for NoSQL

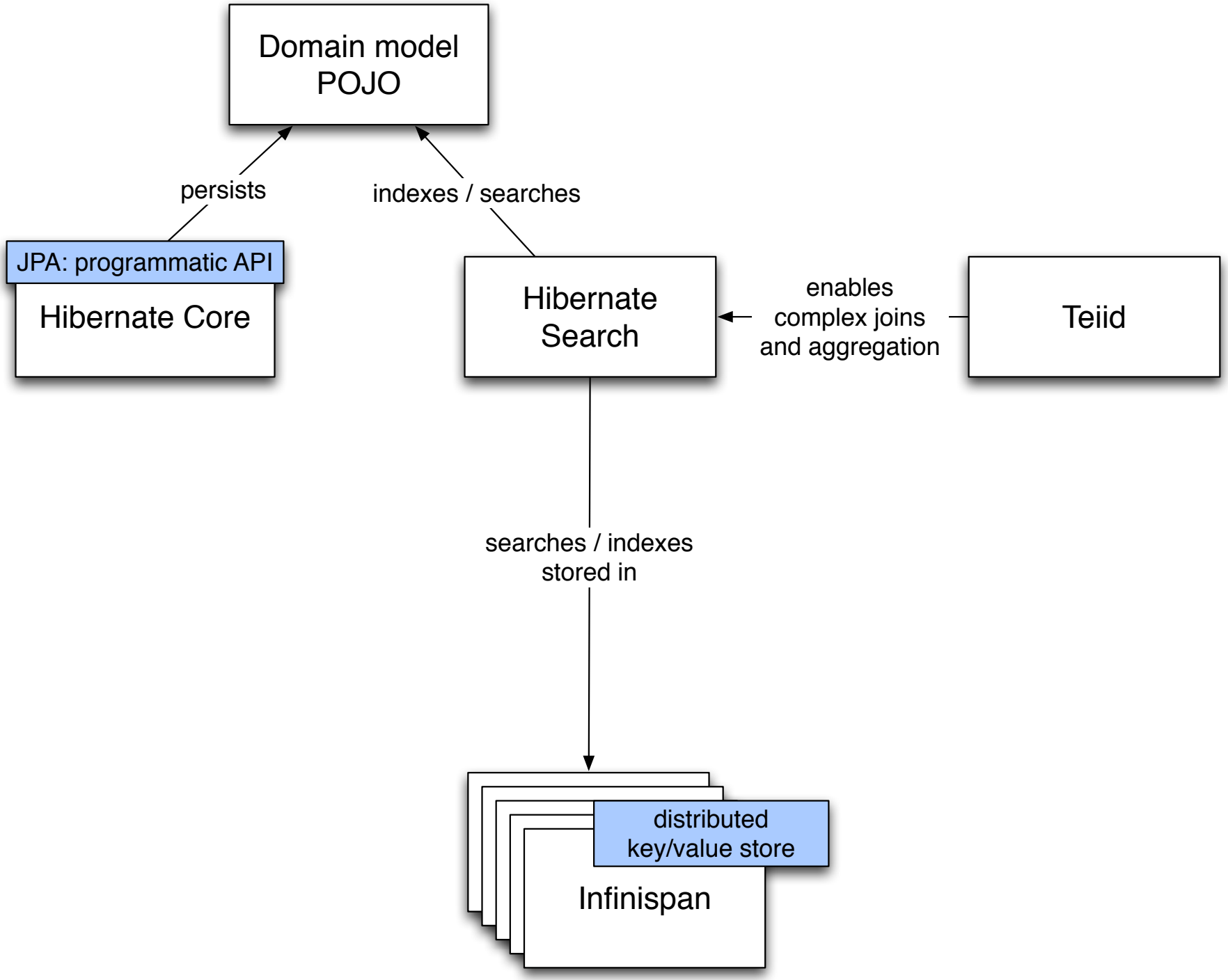
Goals

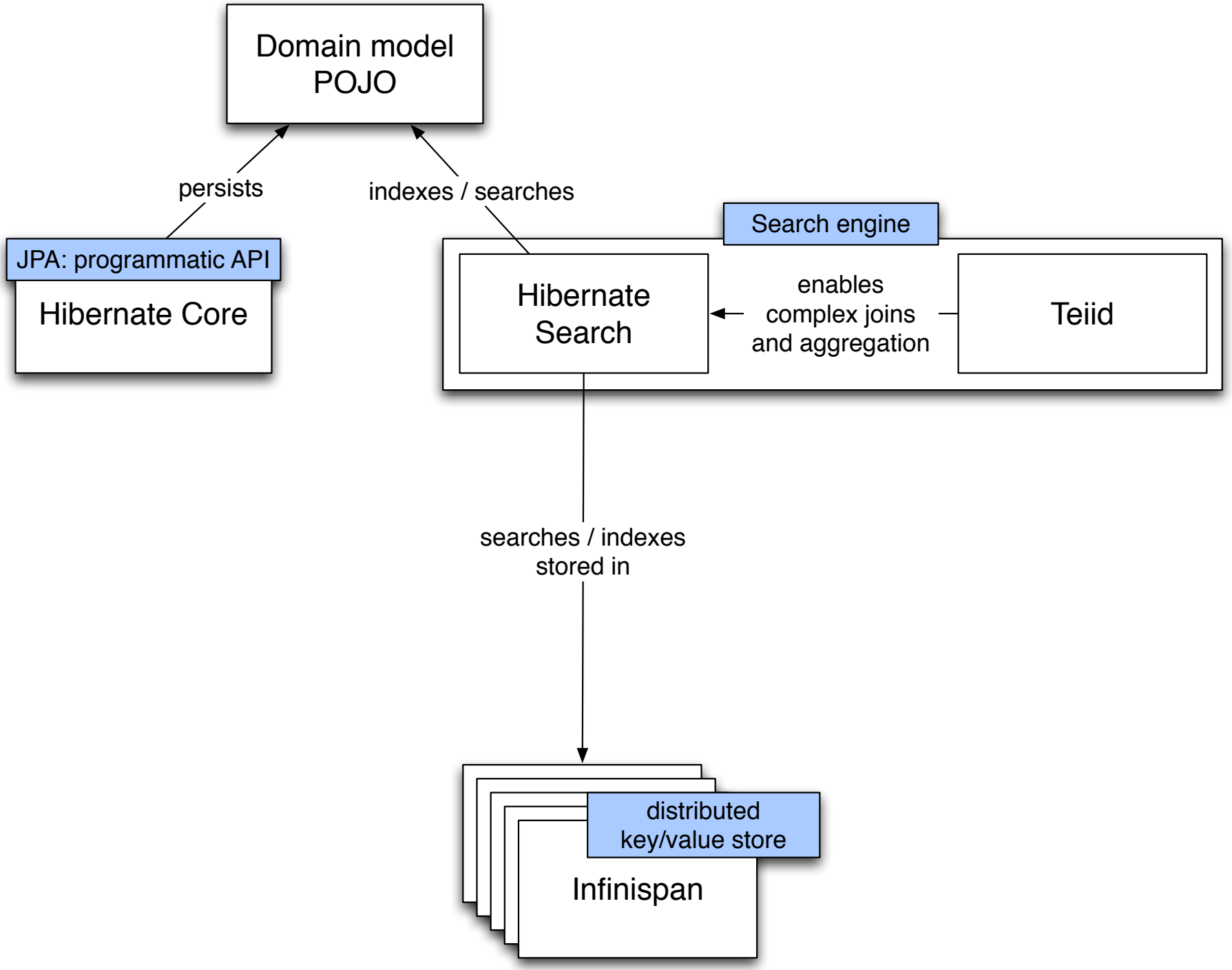
- Encourage new data usage patterns
  - Familiar environment
  - ease of use
  - easy to jump in (and out!)
- Push NoSQL exploration in enterprises
  - data scale up / down
  - NoSQL front end to a traditional RDBMS
- “PaaS on Java EE” initiative

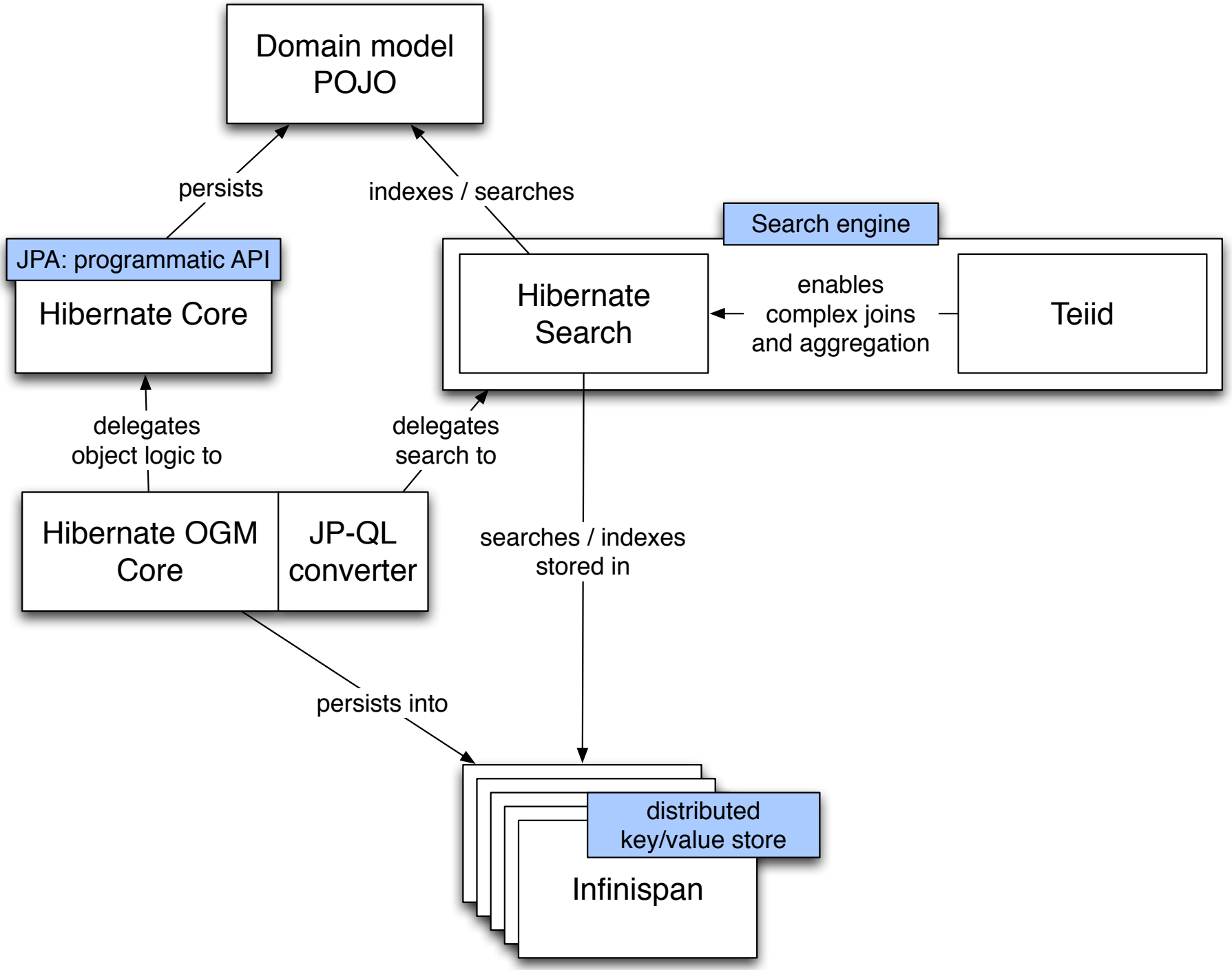
What it does

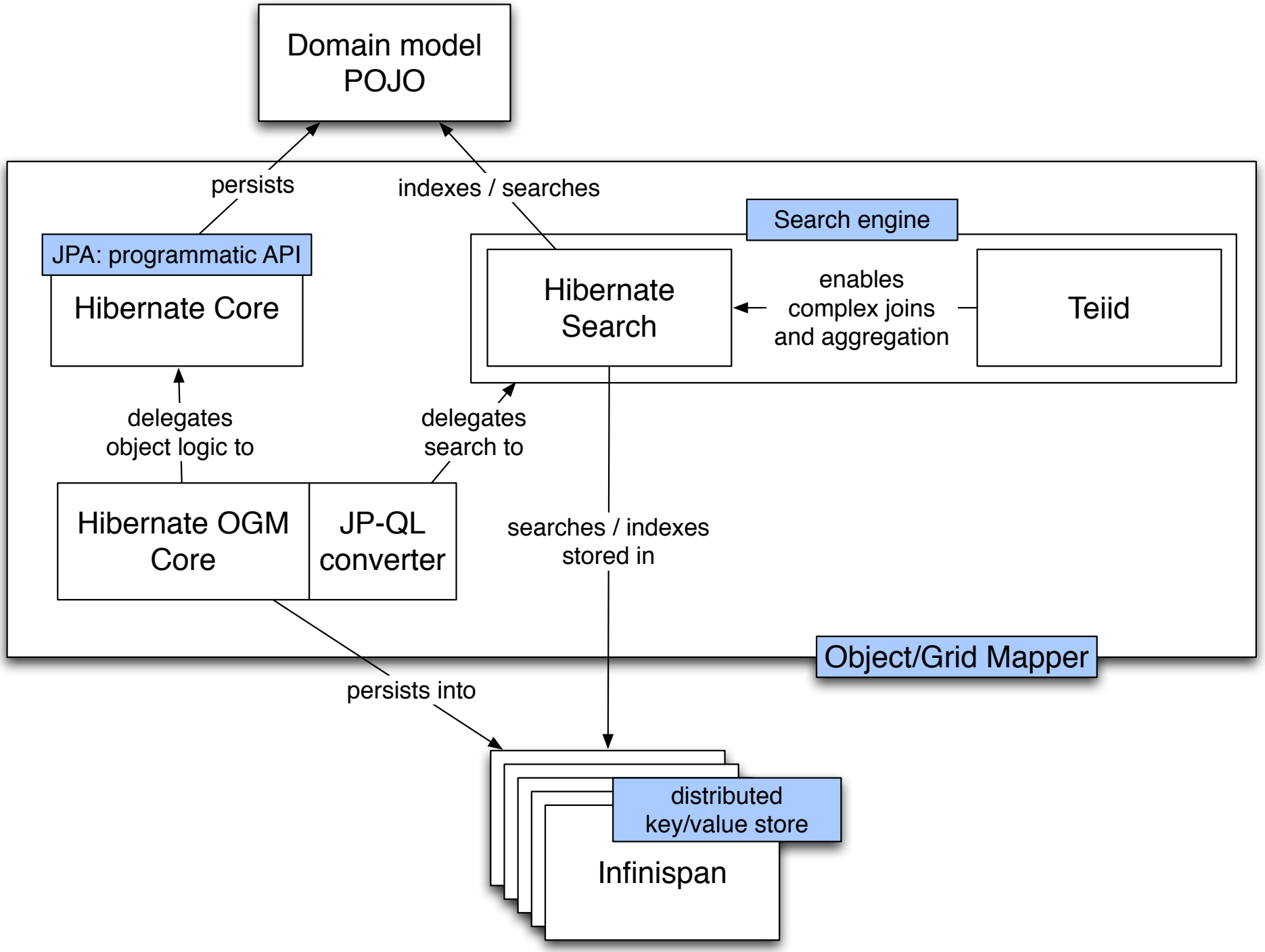
- JPA front end to key/value stores (Infinispan)
  - Object CRUD (incl polymorphism and associations)
  - OO queries (JP-QL)
- Reuses
  - Hibernate Core
  - Hibernate Search (and Lucene)
  - Infinispan
- Is not a silver bullet
  - not for all NoSQL use cases

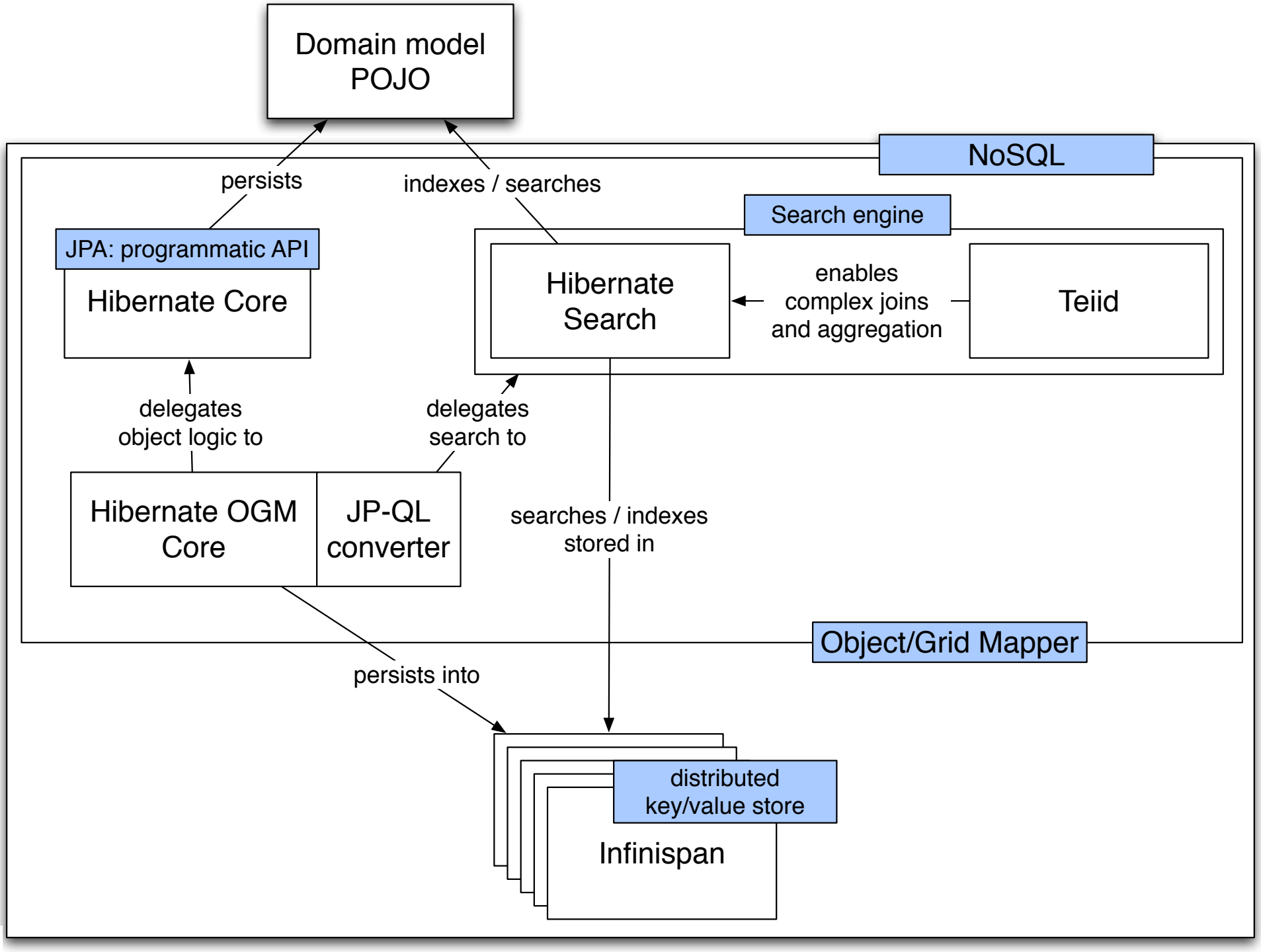



Domain model
POJO
persists
indexes / searches
JPA: programmatic API
Hibernate Core
Hibernate
Search
enables
complex joins
and aggregation
Teiid
searches / indexes
stored in
distributed
key/value store
Infinispan









Concepts

Schema or no schema?

- Schema-less
  - very fast development cycle
  - app deal with old and new structure or migrate all data
  - need strict development guidelines
- Schema
  - strong documentation
  - share with other apps
  - tooling

Entities as serialized blobs?

- Serialize objects into the (key) value
  - store the whole graph?
  - maintain consistency with duplicated objects
  - guaranteed identity a == b
  - concurrency / latency
  - structure change and (de)serialization

OGM's approach

- Keep what's best from relational model
  - as much as possible
- Decorrelate object structure from data structure
  - object model evolution
- Data stored as (self-described) tuples
- Core types limited
  - portability

- Store metadata for queries
  - Lucene index
- CRUD operations are key lookups

Infinispan

- In-memory key / value store + cache store
- Data grid
  - memory >> network >> local disk access
- Transactional
  - JTA / XAResource
- Distributed
  - virtual memory = (sum of all servers)/redundancy

Hibernate OGM's data structure

Storage - Entities

- Each entity in a unique key
  - table name
  - id column names and values
- Value is Map<String,Object>
  - String: column name
  - Object: simple type (serializable)

Storage - Associations

- Cannot store exactly like relational DBs (key lookup)
  - still tuple based
- Simulate navigation to association
  - one key per navigation
    - table name
    - fk column names and values (for a given side)
- Value is the list of tuples
- Focus on speedy reads
  - association writes involve two key lookups

key	value
tbl_user,userId_pk,1	{userId_pk=1,name="Emmanuel"}
tbl_user,userId_pk,2	{userId_pk=2,name="Caroline"}
tbl_address,addressId_pk,3	{addressId_pk=3,city="Paris"}
tbl_address,addressId_pk,5	{addressId_pk=5,city="Atlanta"}
tbl_user_address,userId_fk,1	{ {userId_fk=1, addressId_fk=3}, {userId_fk=1, addressId_fk=5} }
tbl_user_address,userId_fk,2	{ {userId_fk=2, addressId_fk=3} }
tbl_user_address,addressId_fk,5	{ {userId_fk=1, addressId_fk=5} }
tbl_user_address,addressId_fk,3	{ {userId_fk=1, addressId_fk=3}, {userId_fk=2, addressId_fk=3} }

Queries

- Hibernate Search indexes entities
- Store Lucene indexes in Infinispan
- JP-QL to Lucene query

- Works for simple-ish queries

```
select a from Animal a where a.size > 20
> animalQueryBuilder
  .range().onField("size").above(20).excludeLimit()
  .createQuery();

select u from Order o join o.user u
where o.price > 100 and u.city = "Paris"
> orderQB.bool()
  .must(
    orderQB.range().onField("price")
      .above(100).excludeLimit().createQuery() )
  .must(
    orderQB.keyword("user.city").matching("Paris")
      .createQuery() )
  .createQuery();
```

Demo

Future

- Performance
  - support denormalization
  - high performance sequence generator
  - parallel key fetching when possible
- More JP-QL support
  - support joins and aggregation
    - an embedded version of Teiid
- API for massive change
  - Map/Reduce like
  - using some higher order DSL (JP-QL?)

- Support other key/value stores
- Support other NoSQL approaches
  - document, column based, graph?
- Mix NoSQL and RDBMS
  - keep entities on both NoSQL and RDBMS
  - lookups resolved on NoSQL
  - simple queries on NoSQL
  - complex queries on RDBMS
- Better Lucene index storage on Infinispan

Conclusion

- Hibernate OGM
  - JPA for NoSQL (key / value stores and Infinispan initially)
  - reuse mature projects
  - keep the good of the relational model
  - does queries too
- Status
  - alpha quality
  - refining the core data structure
  - quite promising and exciting

More info

- Documentation
  - home: http://www.hibernate.org/subprojects/ogm
  - wiki: http://community.jboss.org/en/hibernate/ogm
  - doc: http://docs.jboss.org/hibernate/ogm/3.0/
  - Any good JPA book ;)
- Code
  - come and contribute! It's fun stuff
  - https://github.com/hibernate/hibernate-ogm
- Q&A

LIKE US ON FACEBOOK

www.facebook.com/redhatinc

FOLLOW US ON TWITTER

www.twitter.com/redhatsummit

TWEET ABOUT IT

#redhat

READ THE BLOG

summitblog.redhat.com

GIVE US FEEDBACK

www.redhat.com/summit/survey