Easy ORM-ness with Objectify-Appengine

Meetu Maltiar

Inphina Technologies

2nd IndicThreads.com Conference On Cloud Computing

3,4 JUNE 2011

PUNE, INDIA

Overall Presentation Goal

Google Datastore Basics

Options available for managing persistence

Objectify-Appengine

Demo of an application using Objectify

Enough About Me

Senior Software Engineer at Inphina

Technologies that interest me:

- Cloud Computing
- Scala
- Hadoop

Datastore Basics

Entities

Operations

Keys

Transactions

Entities

An Entity is an object's worth of data in the datastore

In datastore an Entity is like HashMap-like object of type Entity

Datastore is conceptually a HashMap of keys to entities, and an Entity is conceptually a HashMap of name/value pairs

Operations

Get() an entity as a whole from datastore

Put() an entity as whole in datastore

Delete() an entity from datastore

Query() for entities matching criteria you define

Keys

In the datastore, entities are identified by the id (or name) and a kind, which corresponds to the type of Object you are storing.

So, to get Employee #111 from datastore, we need to call something like get_from_datastore ("Employee", 111)

Keys Continued

There is actually a third parameter required to identify an entity and it is called **parent**

Parent places the child in the same entity group as the parent

Parent (which can be null for the un-parented, root entity) is also required to uniquely identify an Entity.

Keys Continued

So, to get Employee #111 from datastore we need to call something equivalent to:

get_from_datastore ("Employee", 111, null)

or,

get_from_datastore ("Employee", 111, the_parent).

Instead of passing three parameters datastore wraps it in a single Object called Key.

Transactions

Data gets stored in gigantic form of thousands of machines

In order to perform an atomic transaction datastore requires that entities lie on same servers.

To give us more control over where our data is stored, the datastore has a concept of an *entity group*

To give us more control over where our data is stored, the datastore has a concept of an entity group

Transactions Continued

Within a Transaction we can access data from a single entity group

Choose entity groups carefully

Why not have everything in a single entity group?

Google restricts number of requests per second per entity group

Executing Transactions

When we execute get(), put(), delete() or query() in transaction

We must operate it on single entity group

All operations will either fail or succeed completely

If another process modifies the data before commit datastore operation will fail

Tools

Persistence Options

JPA/JDO

Google Datastore

Persistence Frameworks on GAE

- Objectify-Appengine
- Twig
- Simple Datastore
- Slim3

Google Datastore Challenges

Supports just four operations

It persists GAE-Specific entity classes rather than POJO's

Datastore Keys are untyped

JPA/JDO Challenges

Extra Cruft

- Fetch Groups
- Detaching
- Owned/Unowned relationships

Leaky Abstraction

- Keys
- Entity Groups
- Indexes

Developers Dilemma

Objectify

It lets you persist, retrieve, delete and query typed objects

All native datastore features are supported

It provides type safe key and query classes

Objectify Design Considerations

Minimally impacts cold start time. It is light weight

No external dependencies apart from GAE SDK

Working With Datastore

```
Entity ent = new Entity("car");
ent.setProperty("color", "red");
ent.setProperty("doors", 2);
service.put(ent);
```

Objectify ORMNess Objects!

```
Employee emp = new Employee();
emp.setFirstName("John");
emp.setLastName("Smith");
service.put(emp);
```

An Objectify Entity

```
public class Employee {
  @Id Long id;
   private String firstName;
   private String lastName;
}
```

get() Operation

```
Objectify ofy = Objectifyservice.begin();
Employee emp = ofy.get(Employee.class, 123);
Map<Long, Employee> employees =
        ofy.get(Employee.class, 123, 124, 125);
```

put() Operation

```
Objectify ofy = Objectifyservice.begin();
Employee emp = new Employee("John", "adams");
ofy.put(emp);
System.out.println("Id Generated is " + emp.getId());
List<Employee> employees = createEmployees();
ofy.put(employees);
```

delete() Operation

```
Objectify ofy = Objectifyservice.begin();
ofy.delete(Employee.class, 123);

Employee emp = // get Employee from some where
ofy.delete(emp);
```

query() Operation

```
Objectify ofy = Objectifyservice.begin();

List<Employee> employees =
    ofy.query(Employee.class).filter("firstName =", "John")
```

Demo

get()

put()

delete()

query()

Objectify Best Practices

Use a DAO to register entities

Automatic Scanning

- not advised adds to initialization time
- will require dependency jars apart from GAE
- will require changes in web.xml

GAE spins down the instance when not in use. When it comes up the request is slow because of added initialization time. This is called cold start.

Objectify Best Practices …

Use Batch Gets Instead of Queries

Use Indexes sparingly

By default every field of object will be indexed. It comes with heavy computational price.

Use @Unindexed on entity level and @Indexed at fields required for query

Avoid @Parent

In JPA “owned” entity relationship provides referential integrity checking and cascading deletes and saves. Not so here.

Happy Developer

Conclusion

JDO/JPA learning curve is steep due to App Engine's non-relational nature and unique concepts such as entity groups, owned and un-owned relationships.

Google Datastore is low level. It makes no pretense of being relational but also does not allow working with objects. It just stores the objects of type *com.google.appengine.api.datastore.Entity*

Objectify is light weight and it preserves the simplicity and transparency of low level API and does all work converting to and from POJOS to Entity Objects.

mmaltiar@inphina.com

www.inphina.com
http://thoughts.inphina.com

References

Objectify-Appengine
http://code.google.com/p/objectify-appengine/

Google IO 2011 Session on highly productive gwt
rapid development with app-engine objectify-requestfactory
and gwt-platform

Twitter mining with Objectify-Appengine
http://www.ibm.com/developerworks/java/library/j-javadev2-14
-