Groovy, Grails, and the JPA

POJOs, POGOs, and Annotations

Why?

- You have legacy JPA classes
- You need to share code with other JPA based projects
- You have to run somewhere JPA will but GORM won't
- You have an investment in existing JPA tooling

Why are you here?

- who is using Grails?
- who is using the JPA?
- will this combination help your company adopt Grails?

What is the JPA?

Java Persistence API

- one API to rule them all
- many vendors
    - Hibernate
    - OpenJPA
    - TopLink
    - DataNucleus

New in JPA 2.0

@JoinTable
@JoinColumn
@CollectionTable
@OrderColumn
@Embeddable -- now with nesting
@Access -- mixed modes (ie calculated values)
@IdClass -- derived complex identifiers
@EmbeddedId -- complex identifier objects

New locking modes

OPTIMISTIC (= READ)
OPTIMISTIC_FORCE_INCREMENT (= WRITE)

- my next entity manager flush forces a version increment even if the entity was not changed

PESSIMISTIC_READ
PESSIMISTIC_WRITE
PESSIMISTIC_FORCE_INCREMENT

OPTIMISTIC_FORCE_INCREMENT

my next entity manager flush forces a version increment even if the entity was not changed

Two Paths

Grails + JPA green field

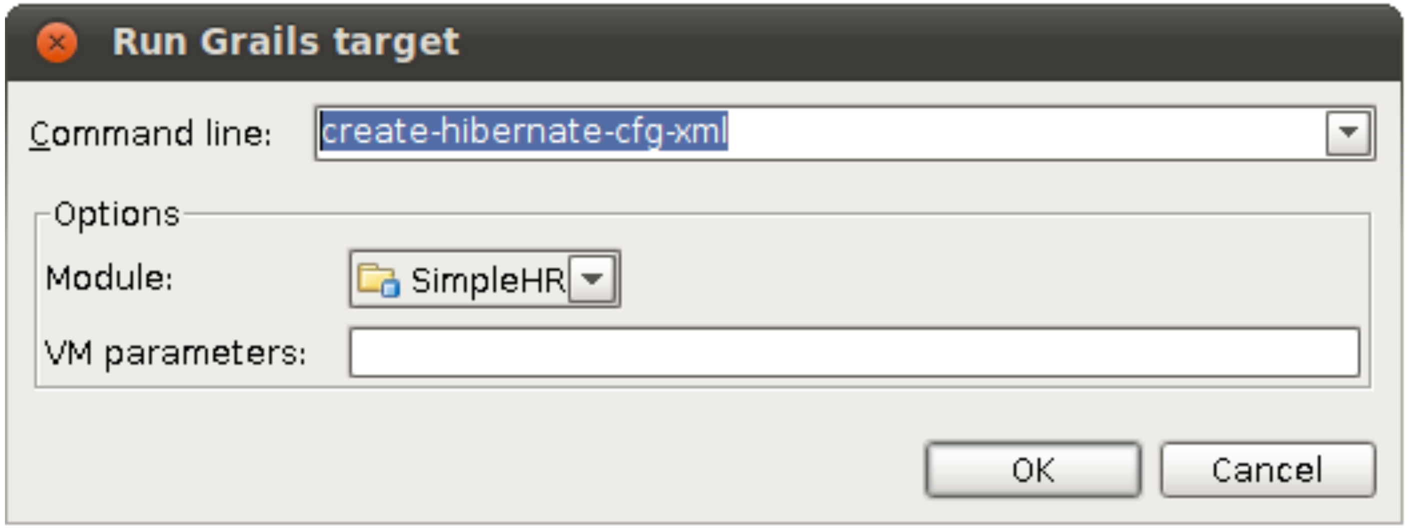
Grails + JPA brown field

Greenfield

start a grails project
"SimpleHR"

Setup for JPA

```
$ grails create-hibernate-cfg-xml
```

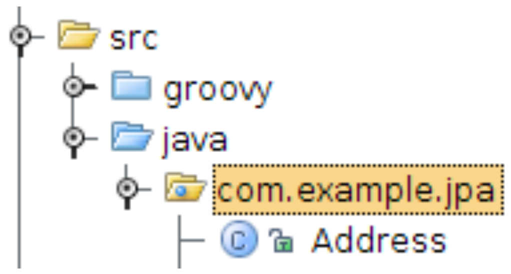


Command line: create-hibernate-cfg-xml

Options
Module: SimpleHR
VM parameters:

Our first JPA class

com.example.jpa.Address



Mapping the Java entity

```
@Entity
public class Address {
    private Long id;
    private Long version;
    private String line1;
    private String line2;
    private String city;
    private String state;
    private String zip;

    @Id
    @GeneratedValue
    public Long getId() {
        return id;
```

Version

```
@Version
public Long getVersion() {
    return version;
}
```

Columns

```
@Column
public String getLine1() {
    return line1;
}
```

now test it

```
class AddressTests extends GrailsUnitTestCase{
    protected void setUp() {
        super.setUp()
    }

    protected void tearDown() {
        super.tearDown()
    }

    void testSimplePersist() {
        def address = new Address()
        assert address.save()
    }
}
```

Did it work?

why not?

hibernate.cfg.xml

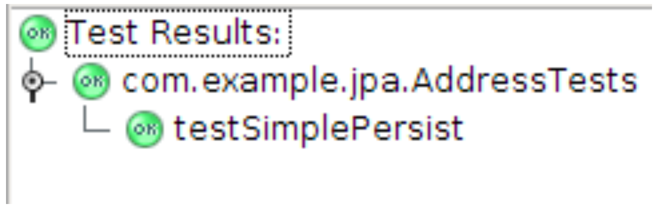
```
<hibernate-configuration>

<session-factory>
        <mapping package="com.example.jpa"/>
        <mapping class="com.example.jpa.Address"/>
</session-factory>

</hibernate-configuration>
```

Test it again

all clear?



```
Test Results:
com.example.jpa.AddressTests
  testSimplePersist
```

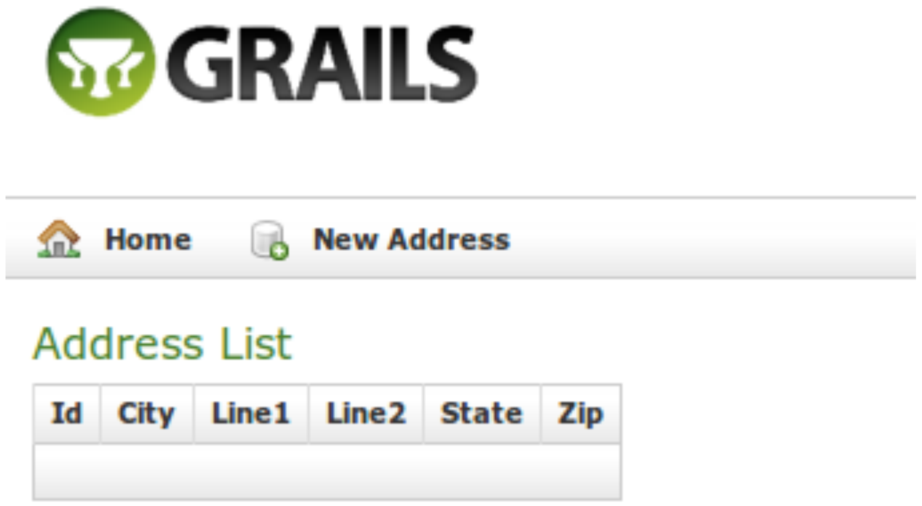
hibernate mapping

each class needs to make it into
hibernate.cfg.xml

views and controllers?

```
$ grails generate-all com.example.jpa.Address
```

Run App



Address List

Id	City	Line1	Line2	State	Zip

no data? that's no fun.

A bit of cheating

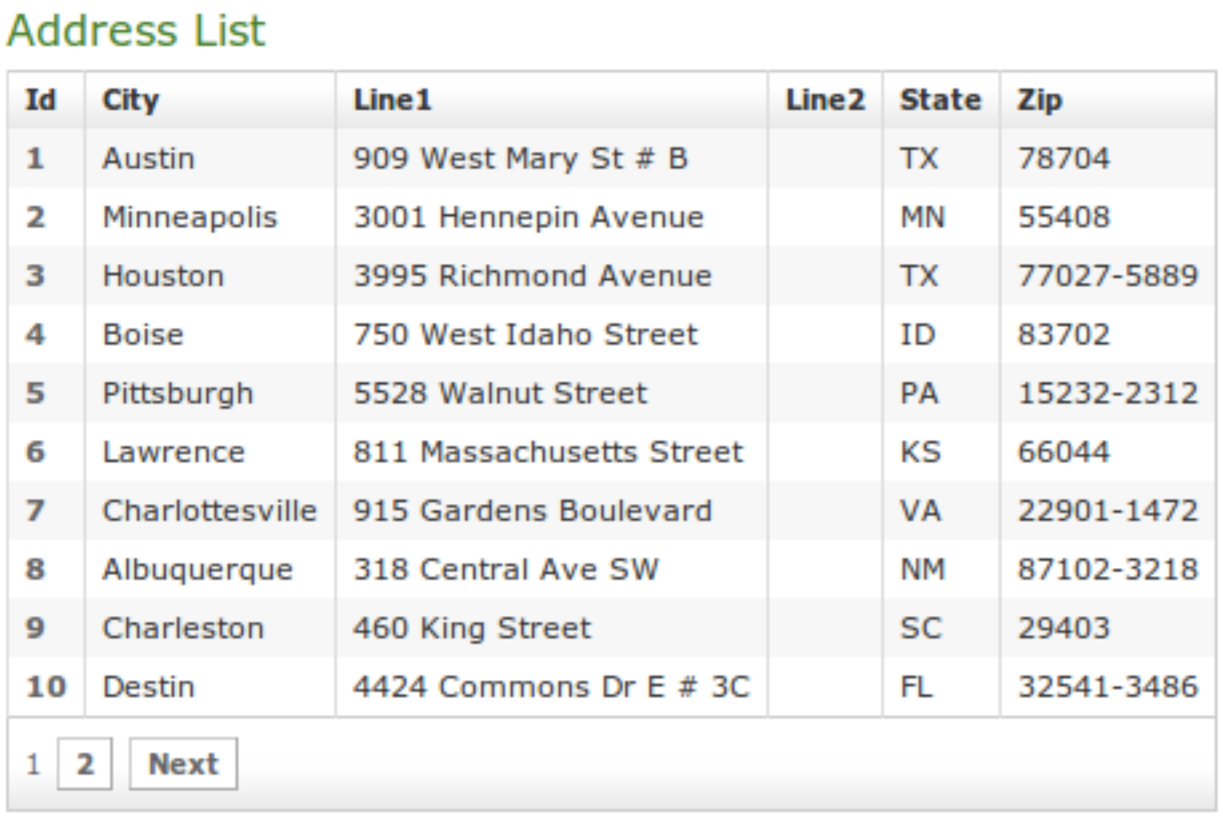
```
class BootStrap {

    def init = { servletContext ->
        if(Address.count() < 1) {
            log.warn "count is less than one"
            new AddressUtil().populateAddresses() // go to this
class
        }
    }
//...
```

a quick and dirty parser

```
file.text.split(/\n|\r/).each { String line ->
    def cols = ['line1', 'city', 'state', 'zip']
    def vals = line.split(/,/).toList()
    def stateZip = vals[2].trim().split(/\s/).toList()
    vals << stateZip.pop()
    vals[2] = stateZip.pop()
    def params = [:]
    4.times {
        params[cols[it]] = vals[it].trim()
    }
    new Address(params).save() // uncomment
}
```

Now we have data



Address List

Id	City	Line1	Line2	State	Zip
1	Austin	909 West Mary St # B		TX	78704
2	Minneapolis	3001 Hennepin Avenue		MN	55408
3	Houston	3995 Richmond Avenue		TX	77027-5889
4	Boise	750 West Idaho Street		ID	83702
5	Pittsburgh	5528 Walnut Street		PA	15232-2312
6	Lawrence	811 Massachusetts Street		KS	66044
7	Charlottesville	915 Gardens Boulevard		VA	22901-1472
8	Albuquerque	318 Central Ave SW		NM	87102-3218
9	Charleston	460 King Street		SC	29403
10	Destin	4424 Commons Dr E # 3C		FL	32541-3486

Groovy JPA

```
@Entity
class Person {
    @Id @GeneratedValue
    Long id
    @Version
    Long version
    @Column
    String name
    @OneToOne
    @JoinColumn(name="address_id")
    Address address
    String toString() { name }
}
```

Department.groovy

```
public class Department {
    Long id
    Long version
    String name
    Person manager

    Set<Person> staff
    String toString() { name }
}

// how are you going to annotate this?
```

hibernate.cfg.xml

```
<session-factory>
            <mapping package="com.example.jpa"/>
            <mapping class="com.example.jpa.Address"/>
            <mapping class="com.example.jpa.Person"/>
            <mapping class="com.example.jpa.Department"/>
</session-factory>
```

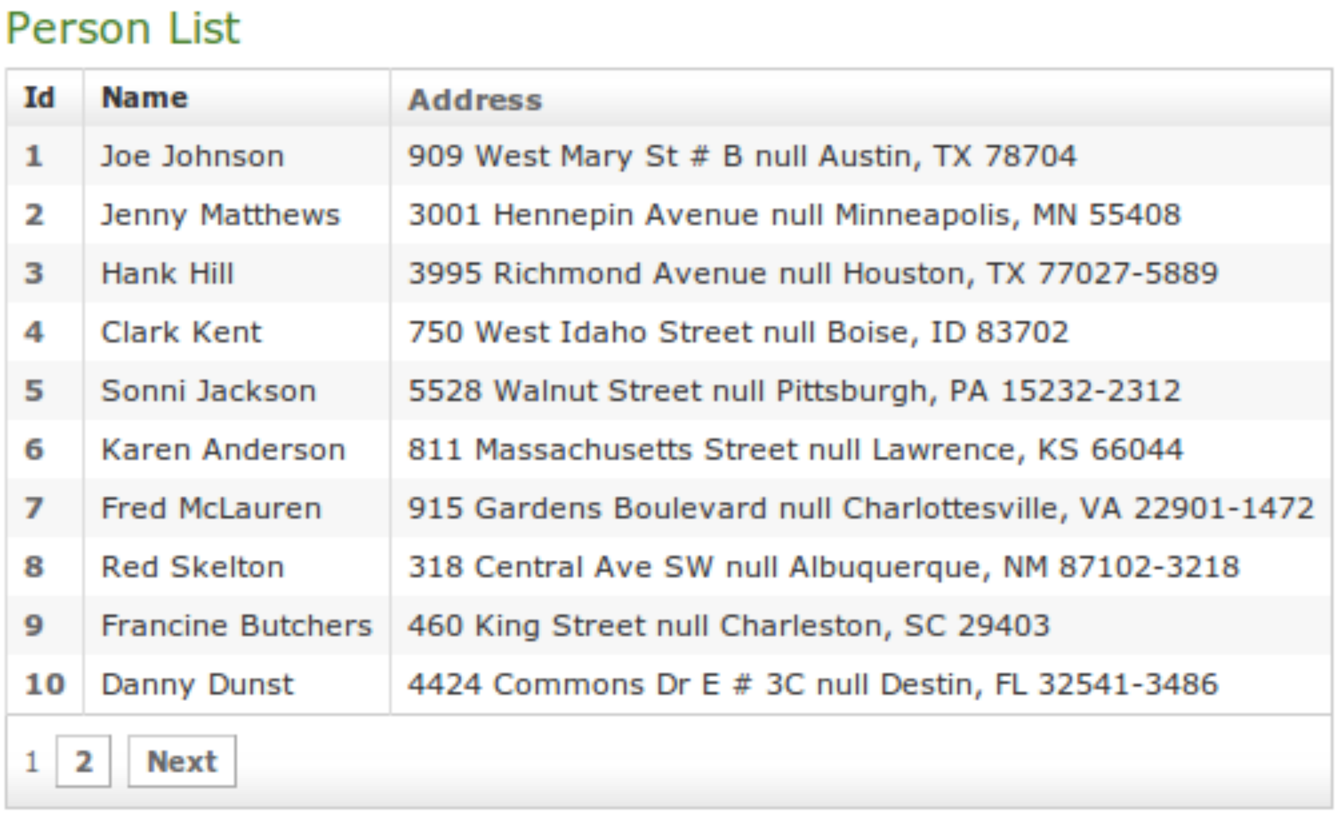
More Data?

```
if(Person.count() < 1) {
    new AddressUtil().populatePeople()
}
if(Department.count() < 1) {
    new AddressUtil().populateDepartments()
}
```

Run

```
$ grails generate-all com.example.jpa.Person
$ grails generate-all com.example.jpa.Department
```

Grails Joy


Person List

Id	Name	Address
1	Joe Johnson	909 West Mary St # B null Austin, TX 78704
2	Jenny Matthews	3001 Hennepin Avenue null Minneapolis, MN 55408
3	Hank Hill	3995 Richmond Avenue null Houston, TX 77027-5889
4	Clark Kent	750 West Idaho Street null Boise, ID 83702
5	Sonni Jackson	5528 Walnut Street null Pittsburgh, PA 15232-2312
6	Karen Anderson	811 Massachusetts Street null Lawrence, KS 66044
7	Fred McLauren	915 Gardens Boulevard null Charlottesville, VA 22901-1472
8	Red Skelton	318 Central Ave SW null Albuquerque, NM 87102-3218
9	Francine Butchers	460 King Street null Charleston, SC 29403
10	Danny Dunst	4424 Commons Dr E # 3C null Destin, FL 32541-3486

Recap JPA

- JPA classes in src/java
  - where else can they live?
  - is the directory important?
  - how does Hibernate know about the class?
- How does GORM know about the class?
  - what does this tell us about GORM?
  - How do Controllers and GSP know about the class?

Solve a problem?

webflow for people to ask for vacation time.
(live coding)

thus ends the simple part of our work shop

Brown Field

impossible.sql

ORM impossible?

Clients tell me:

we can't use ORM because

- it's impossible to map our database.
- we want to use our own queries
- our schema is too complex
- our data is too denormalized

really?

I doubt it.

ImpossibleHR

Some of the worst database designs I've ever encountered, together in one single horrifying place.

impossible

<show impossible.sql>

ImpossibleHR

How are we going to map this?

violations of *all* normal forms!

Embedded Entities

```
public class Application {
    private Long id;
  private Long version;
    private Date startedDate = new Date();
  private Date editedDate = new Date();
  private Name name;

    private Country residence;
    private Citizenship citizenship;
    private Address address;
    private Address mailingAddress;
```

JP-QL

@SqlResultSetMapping
@EntityResult
@FieldResult
@ColumnResult

Impossible?

maybe we *should* just migrate the data

Open Software Integrators

Shawn Hartsock

Senior Consultant

shartsock@osintegrators.com
hartsock@acm.org
http://twitter.com/hartsock
http://www.linkedin.com/in/hartsock