Project notes

- You need to have a JDK installed, not a JRE
- Polls: How many have gotten as far as compiling JSPs?
Office hours

• John
  • Monday 1:30-2:30
  • Tuesday 3-4
  • Friday 2-3

• Bill
  • Monday 11-noon
  • TuTh 1:45-2:15
Helping each other

• Project questions should be posted to forum rather than emailed to TA and myself
  • allow others to help out, benefit from responses
  • use common sense
• If I’m talking with a student during office hours, ask if you can join us
  • I can often multitask
Web places for help

• 433 Forum

• http://stackoverflow.com

• http://www.coderanch.com/forums

• http://groups.google.com/group/google-appengine-java

• http://gae-java-persistence.blogspot.com/

• Add your own to forum posting
Play with it

• Got a question? Try it out yourself
• The only way to learn a technology

• Note: for local appengine deployments, access local admin console
  • http://localhost:8888/__ah/admin
Datastore

- Automatic conversion between POJOs and a persistent store
  - typically, one table per class
    - one row per instance
    - one column per field
Java datastore mechanisms

• Direct access to datastore
• JDO - Java Data Objects
• JPA - Java Persistence API
• JDO and JPA provide similar capabilities
Making a class persistent

@PersistenceCapable(identityType = IdentityType.APPLICATION)

public class Employee {
    @PrimaryKey
    @Persistent(valueStrategy = IdGeneratorStrategy.IDENTITY)
    private Key key;

    @Persistent private String firstName;
    @Persistent private String lastName;
    @Persistent private Date hireDate;
    ...
}

Making a instance persistent

Employee employee = new Employee("Alfred", "Smith", new Date());

PersistenceManager pm = PMF.get().getPersistenceManager();

try {
    pm.makePersistent(employee);
} finally {
    pm.close();
}
Modifying a persistent instance

- If you get an object from a persistence manager, it keeps track of changes to the object
- written back when persistence manager is closed
Getting an instance

- Employee e = pm.getObjectById(Employee.class, user.getEmail());
- Works for keys that are Long, unencoded strings and Keys
public void updateEmployeeTitle(User user, String newTitle) {

    PersistenceManager pm = PMF.get().getPersistenceManager();

    try {
        Employee e = pm.getObjectById(Employee.class, user.getEmail());
        if (titleChangeIsAuthorized(e, newTitle)) e.setTitle(newTitle);
        else throw new UnauthorizedTitleChangeException(e, newTitle);
    } finally {
        pm.close();
    }
}
Deleting a persistent object

• Just call pm.deletePersistent(e);
JDOQL

Query query = pm.newQuery(Employee.class);
query.setFilter("lastName == lastNameParam");
query.setOrdering("hireDate desc");
query.declareParameters("String lastNameParam");

List<Employee> results = (List<Employee>)
query.execute("Smith");
query.execute("Smith");
Query value literals

Query query = pm.newQuery(Employee.class,
   "lastName == 'Smith' order by hireDate desc");

• Don’t use value literals with untrusted data
SQL / JDOQL injection

HI, THIS IS YOUR SON'S SCHOOL. WE'RE HAVING SOME COMPUTER TROUBLE.

OH, DEAR - DID HE BREAK SOMETHING? IN A WAY-

DID YOU REALLY NAME YOUR SON Robert'); DROP TABLE Students;-- ?

OH, YES. LITTLE BOBBY TABLES, WE CALL HIM.

WELL, WE'VE LOST THIS YEAR'S STUDENT RECORDS. I HOPE YOU'RE HAPPY.

AND I HOPE YOU'VE LEARNED TO SANITIZE YOUR DATABASE INPUTS.
indexes

- Datastore needs indexes to support queries
- Need index on lastName and height to execute
- `select from Person where lastName == "Smith" && height < 72`
- In execution on localhost, new indexes automatically added
  - added to xml file, automatically created on server
- if you test, don’t need to add them manually
Extents

- Used to iterate over all instances of a class
- Will automatically fetch additional instances from datastore
  - can exceed limit of 1000 responses on queries
- Need to be closed
Fields

• **Core types** (primitives, Strings, etc)
• Keys
• **Serialized objects**
• Relationships
Keys

• Every persistent object must have a key
  • automatically generated Long
  • application defined String
• Key - built in datastore type
• Key encoded as String
Serialized objects

- Any object that can be serialized
  - e.g., collections of serializable objects
  - such as a map from String to Integer
Relationships

- One to one
- One to many
- Can be owned (hierarchical)
- can be bidirectional
Owned one-to-one

- parent/child relationship
  - child cannot exist without the parent
  - Can be bidirectional: child notes which fields identifies parent
Owned one to many

- Collection of children
  - some complications/inefficiencies with lists
    - insertions/deletions that change position are ugly
Dependent children and Cascade delete

• A child can be a dependent child
  • cannot exist without parent
• Deletion of parent forces deletion of child
• assignment of child forces deletion of previous child
entity groups

• Each persistent object belongs to an entity group
• Objects without parents belong to their own entity group
• Objects owned by their parent belong to their parents entity group
• parent can’t change
Unowned relationships

- Not automatic
- One to many and many to many relationships
- Store Keys for objects you want to reference
- have to look up objects
Transactions

- A transaction is a set of queries and updates
- Performed completely and atomically, or not at all
Example transaction

Employee e = new Employee();

ContactInfo ci = new ContactInfo();
e.setContactInfo(ci);

Transaction tx = pm.currentTransaction();

try {
    tx.begin();
    pm.makePersistent(e);
    tx.commit();
}

Ensures that either Employee and ContactInfo both are made persistent, or neither are.

Even more important when doing things like transferring funds between two accounts, or updating a record and a log entry.
Transactions can’t span entity groups

- A transaction can’t span entity groups
- A limitation of Google datastore
- Database transactions can span any set of operation
  - but supporting that is trick
Owned one-to-many

- A parent has a collection of children
- See http://code.google.com/appengine/docs/java/datastore/dataclasses.html#Collections
Projects

• Project 1, due today (26 deployments so far)
• Project 2, Idea book:
  • signed in users can vote + or -
  • each user gets one vote per idea, can change vote
  • list ideas by net votes
• Due Feb 9th