Yelp Data Challenge

https://github.com/apptsunami/yelpdatachallenge

Phase 1

5/18/2013

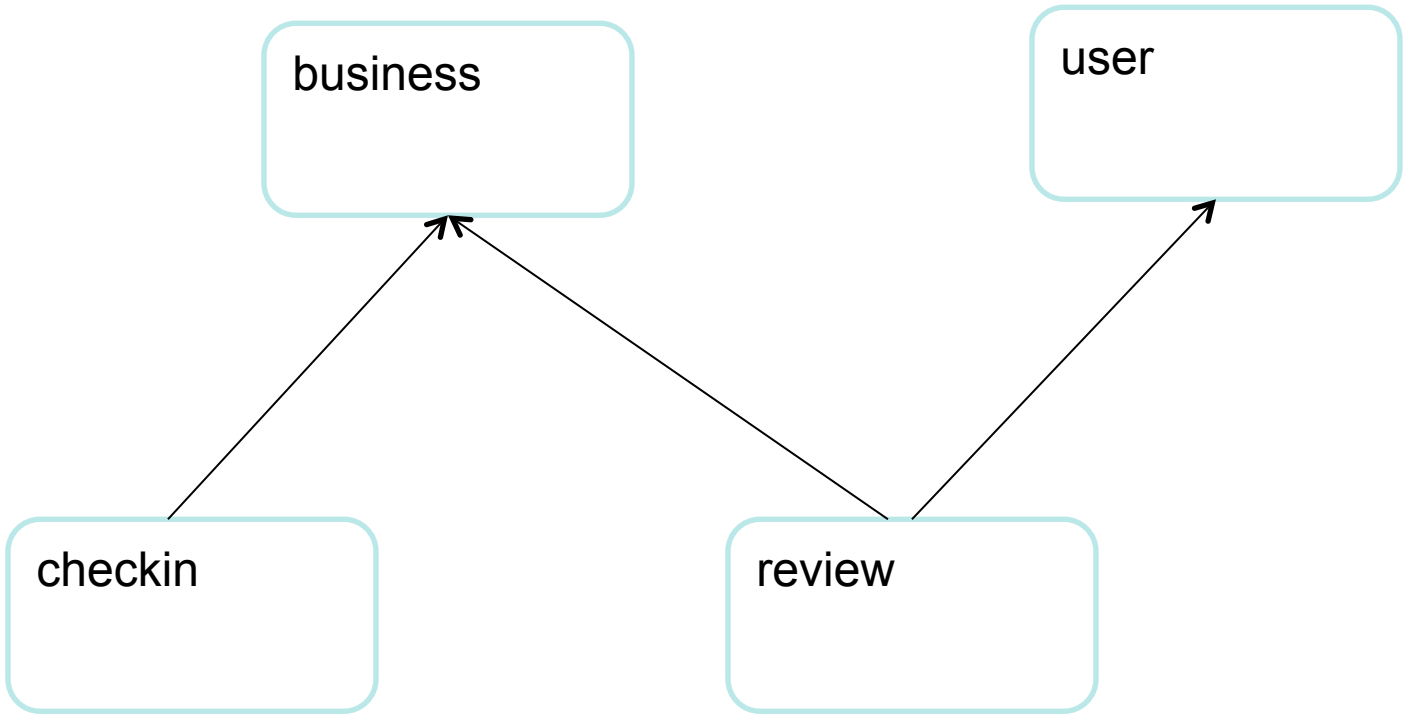
schan@apptsunami.com

Data Source

- http://www.yelp.com/dataset_challenge
- Data set consists of 4 json files
  - Business
  - User
  - Checkin
  - Review
- Each line is a json string with unique static id


```
{"business_id": "rncjoVoEFUJGCUoC1JgnUA", "full_address": "8466 W Peoria Ave\nSte
6\nPeoria, AZ 85345", "open": true, "categories": ["Accountants", "Professional
Services", "Tax Services", "Financial Services"], "city": "Peoria", "review_count": 3,
"name": "Peoria Income Tax Service", "neighborhoods": [], "longitude": -112.241596,
"state": "AZ", "stars": 5.0, "latitude": 33.581867000000003, "type": "business"}
```

Data Model

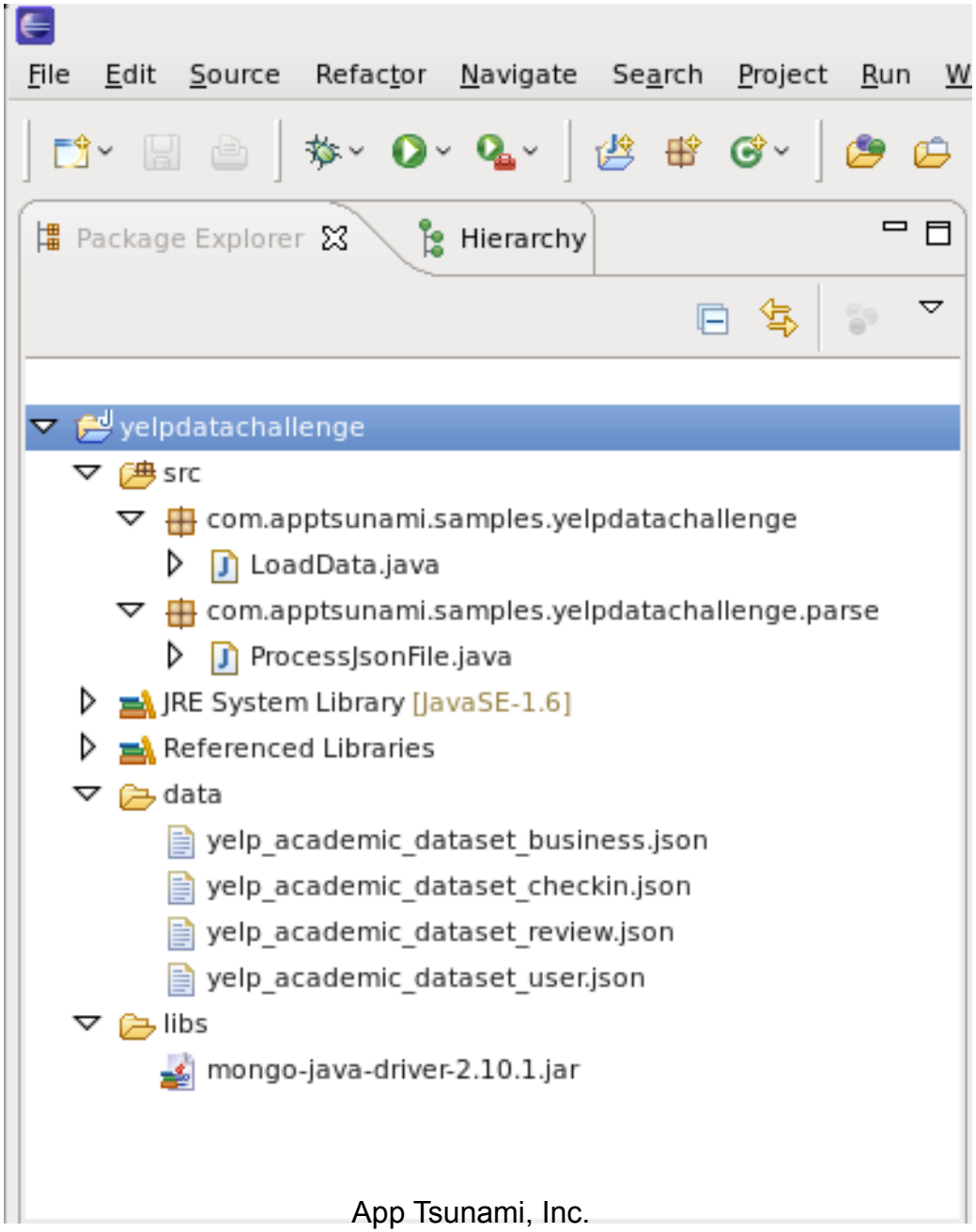


Development Phases

- Data loading
  - Read each file into a mongo collection
  - Each record is one line in a data file with a Yelp unique id
  - There are cross-references between collections by unique ids
- Analysis
  - TBD

IDE

- Eclipse 3.5.2 (Galileo) on Red Hat 5 and Java 6
- Create a Java project *yelpdatachallenge*
- Download Java source files from github into the folder *src*
  - https://github.com/apptsunami/yelpdatachallenge
  - There are 2 files: ProcessJsonFile.java, LoadData.java
- Put the 4 data files in a folder *data*


File Edit Source Refactor Navigate Search Project Run W
Package Explorer
Hierarchy
yelpdatachallenge
src
com.apptsunami.samples.yelpdatachallenge
LoadData.java
com.apptsunami.samples.yelpdatachallenge.parse
ProcessJsonFile.java
JRE System Library [JavaSE-1.6]
Referenced Libraries
data
yelp_academic_dataset_business.json
yelp_academic_dataset_checkin.json
yelp_academic_dataset_review.json
yelp_academic_dataset_user.json
libs
mongo-java-driver-2.10.1.jar

Database Driver

- Mongo provides mongo-java-driver
  - Use v2.10+ (new MongoClient class)
- In the Eclipse project create a folder *libs*
- Download *mongo-java-driver-2.10.1.jar* into *libs*
  - https://github.com/mongodb/mongo-java-driver/downloads
  - http://central.maven.org/maven2/org/mongodb/mongo-java-driver/
- In the Eclipse project *properties* command add *mongo-java-driver-2.10.1.jar* as a library

ProcessJsonFile.java

- Create a class *ProcessJsonFile.java*
- Create a function *processFile*

```
public void processFile(File inputFile, String collectionName)
            throws Exception {
    /*
            Open database
            Read & insert data
            Read & print collection
            Verify data count
            Close connection
     */
}
```

Open Database

- Create a MongoClient object
- Open database
- Login with user name and password (optional)
- Set MongoClient *WriteConcern*
  - **Journaled**: Exceptions are raised for network issues, and server errors; the write operation waits for the server to group commit to the journal file on disk.

Open Database

```
MongoClient mongoClient = new MongoClient(DB_HOST,
       DB_PORT);
DB db = mongoClient.getDB(DB_NAME);
if ((DB_USER != null) && (DB_PASSWORD != null)) {
       boolean auth = db.authenticate(DB_USER,
              DB_PASSWORD.toCharArray());
       if (!auth) {
              throw new Exception(
                     "Incorrect user name or password");
       }
} // if
mongoClient.setWriteConcern(WriteConcern.JOURNALED);
```

Read & Insert Data

- Open collection by name
- Delete all existing records in collection
- Open input json file
- While input file has next line
  - Read next line from input file
  - Convert line into DBObject
  - Insert DBObject into collection
- Close input file

Read & Insert Data

```
DBCollection coll = db.getCollection(collectionName);
coll.drop();
BufferedReader br = new BufferedReader(
       new FileReader(inputFile));
String line = null;
int lineCount = 0;
while ((line = br.readLine()) != null) {
       DBObject dbObject = (DBObject) JSON.parse(line);
       coll.insert(dbObject);
       lineCount++;
} // while
br.close();
```

Read & Print Collection

- Open collection by name
- Create DBCursor by finding all records
- While DBCursor.hasNext
  - Print DBCursor.next
- Close DBCursor
- Return number of records found

Read & Print Collection

```
private int dumpCollection(DB db, String collectionName) {
        DBCollection coll = db.getCollection(collectionName);
        if (coll == null) {
                System.out.println("Cannot get collection " + collectionName);
                return 0;
        }
        System.out.println("Dump collection " + collectionName);
        System.out.println("====");
        DBCursor cursorDoc = coll.find();
        int count = 0;
        while (cursorDoc.hasNext()) {
                System.out.println(cursorDoc.next());
                count++;
        } // while
        cursorDoc.close();
        System.out.println("==== " + count + " ====");
        return count;
} // dumpCollection
```

Verify Data Count

- Retrieve the number of records in collection
- If it does not equal to the number of lines read from input file
  - Report error

Verify Data Count

```
int dumpCount = dumpCollection(db, collectionName);
if (dumpCount != lineCount) {
       System.out.println("Read " + lineCount
              + " from data file but retrieved ”
              + dumpCount
              + " objects in collection "
              + collectionName);
}
```

Close Connection

- Call close() on the MongoClient

Close Connection

```
mongoClient.close();
```

Main Program: LoadData.java

- Create a *ProcessJsonFile* object
- For each argument
  - Compute collection name from file name
    - Call *ProcessJsonFile.processFile* to read the input file and insert all records into collection

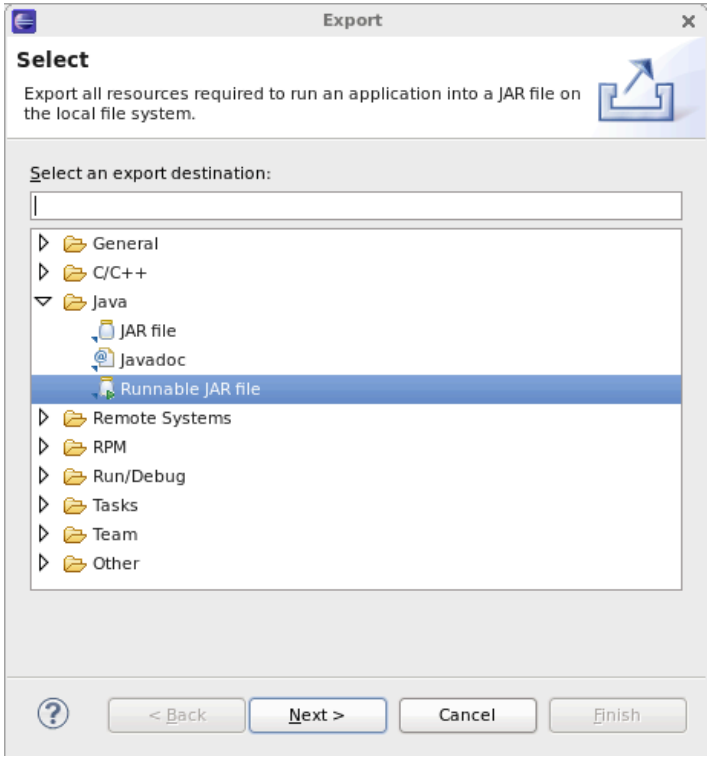
Main Program

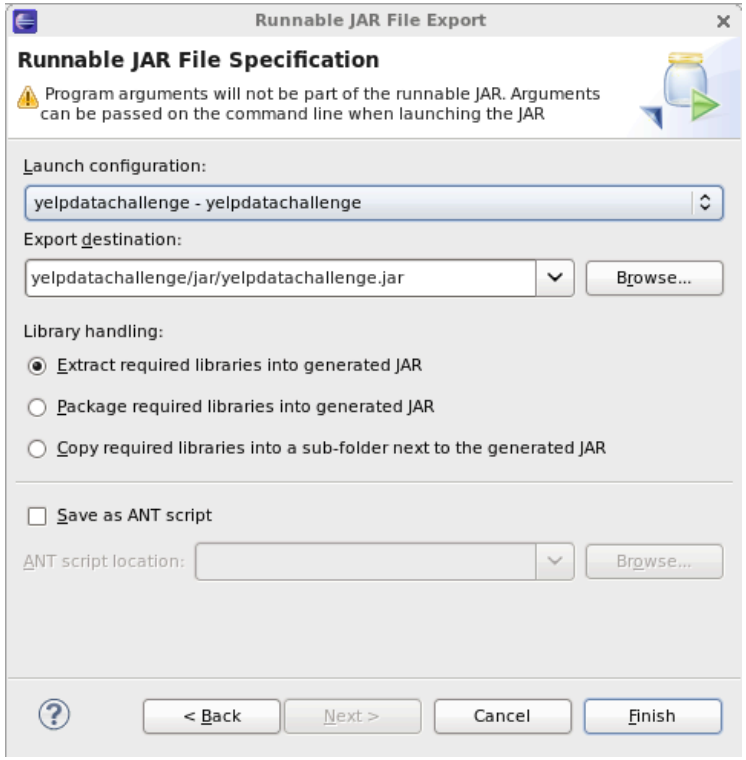
```
public static void main(String[] args) {
	ProcessJsonFile parser = new ProcessJsonFile();
	for (int i = 0; i < args.length; i++) {
		String fileName = args[i];
		File inputFile = new File(fileName);
		try {
			parser.processFile(inputFile,
			getCollectionNameFromFileName(fileName));
		} catch (Exception e) {
			e.printStackTrace();
		}
	} // for
} // main
```

Export Executable Jar

- In Eclipse:
  - Create a folder *jar*
  - *Export* command (from the *File* pulldown)
    - Select *Runnable JAR File*
    - Select *Extract required libraries into generated JAR file*
  - A file *yelpdatachallenge.jar* will be generated in the *jar* folder

Export Executable Jar





Execution

- Launch bash
- Run the following:

```
cd workspace/yelpdatachallenge/jar
java –jar yelpdatachallenge.jar \
../data/yelp_academic_dataset_business.json \
../data/yelp_academic_dataset_checkin.json \
../data/yelp_academic_dataset_review.json \
../data/yelp_academic_dataset_user.json
```

Output (without data dump)

```
start
loading file 0: ../data/yelp_academic_dataset_business.json
Dump collection business
==== 11537 ====
loading file 1: ../data/yelp_academic_dataset_checkin.json
Dump collection checkin
==== 8282 ====
loading file 2: ../data/yelp_academic_dataset_review.json
Dump collection review
==== 229907 ====
loading file 3: ../data/yelp_academic_dataset_user.json
Dump collection user
==== 43873 ====
finish
```

Questions?

- schan@apptsunami.com