Mobile Data Management in Location Based Services

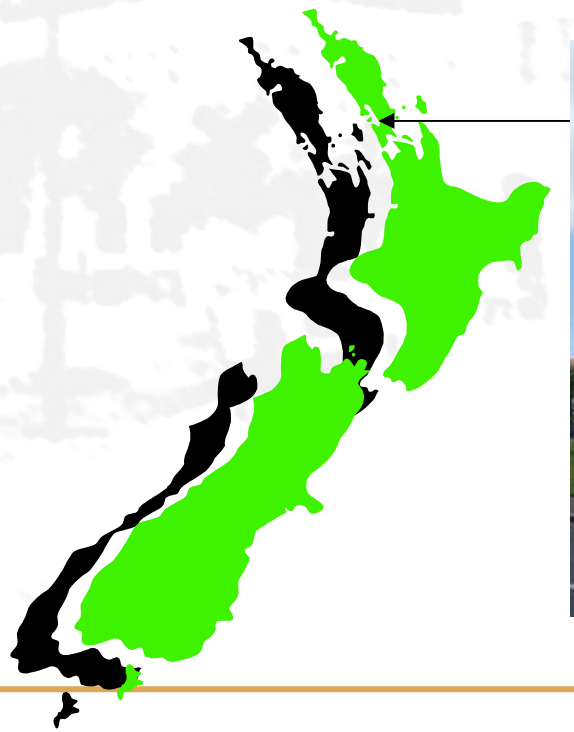
David Parsons
Massey University
Auckland, New Zealand

Massey University

Massey University

My Department

- Information Systems, Massey University, Auckland, New Zealand

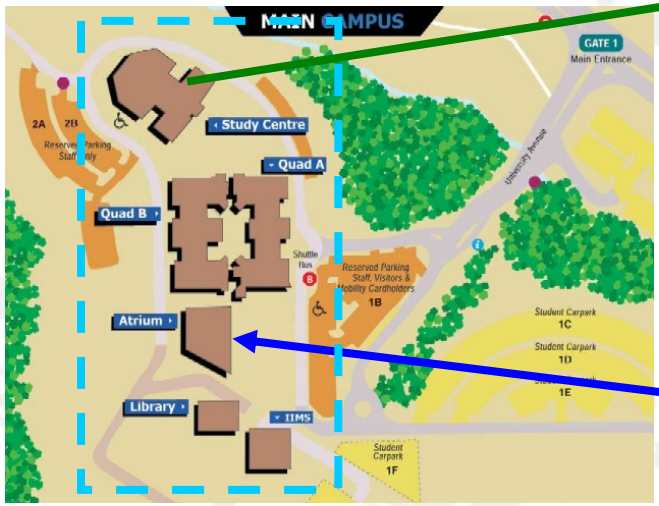


Massey University

Agenda

- Location API for J2ME
- Landmark Store Mobile Database
- JDO
- Aspects

Location Based Service components

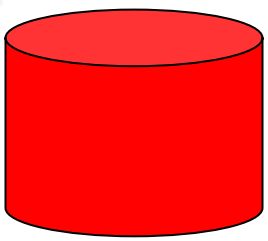


Geocoding

Azimuth / distance

Reverse geocoding

Point Of Interest data



Maps, yellow pages, user profiles, dynamic data…

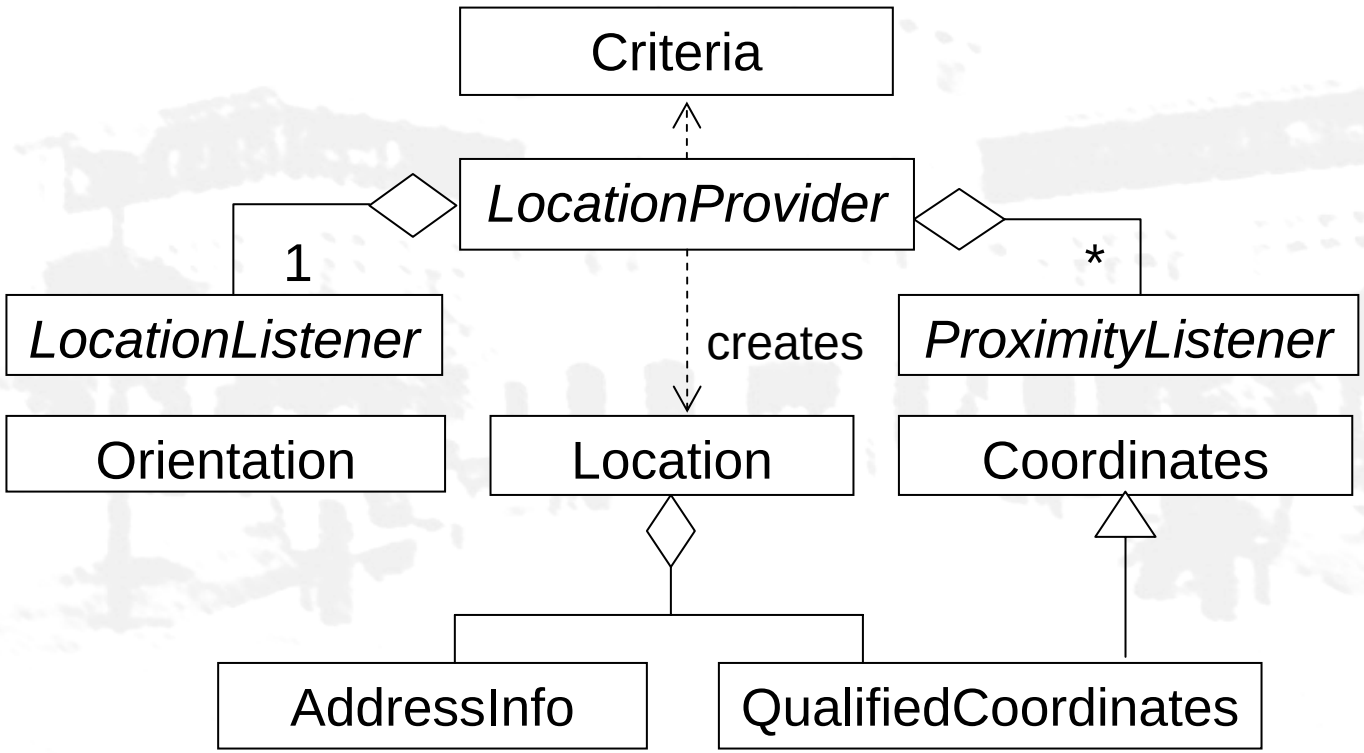
Massey University

Java Location API for J2ME

- Compatible with multiple positioning technologies
  - Network / GPS / Bluetooth…
- Locations
  - Latitude, longitude, timestamp , etc.
- Addresses
  - Including building floor, room etc.
- Landmarks in categories

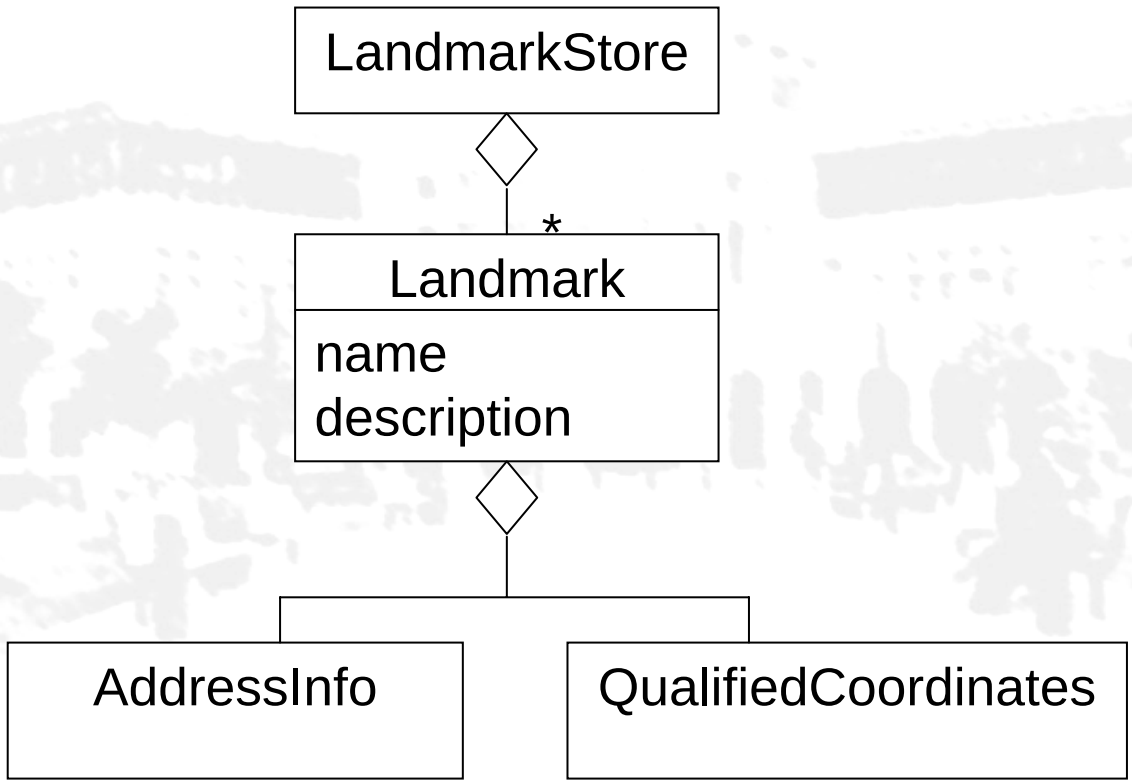
Massey University

Location-related classes



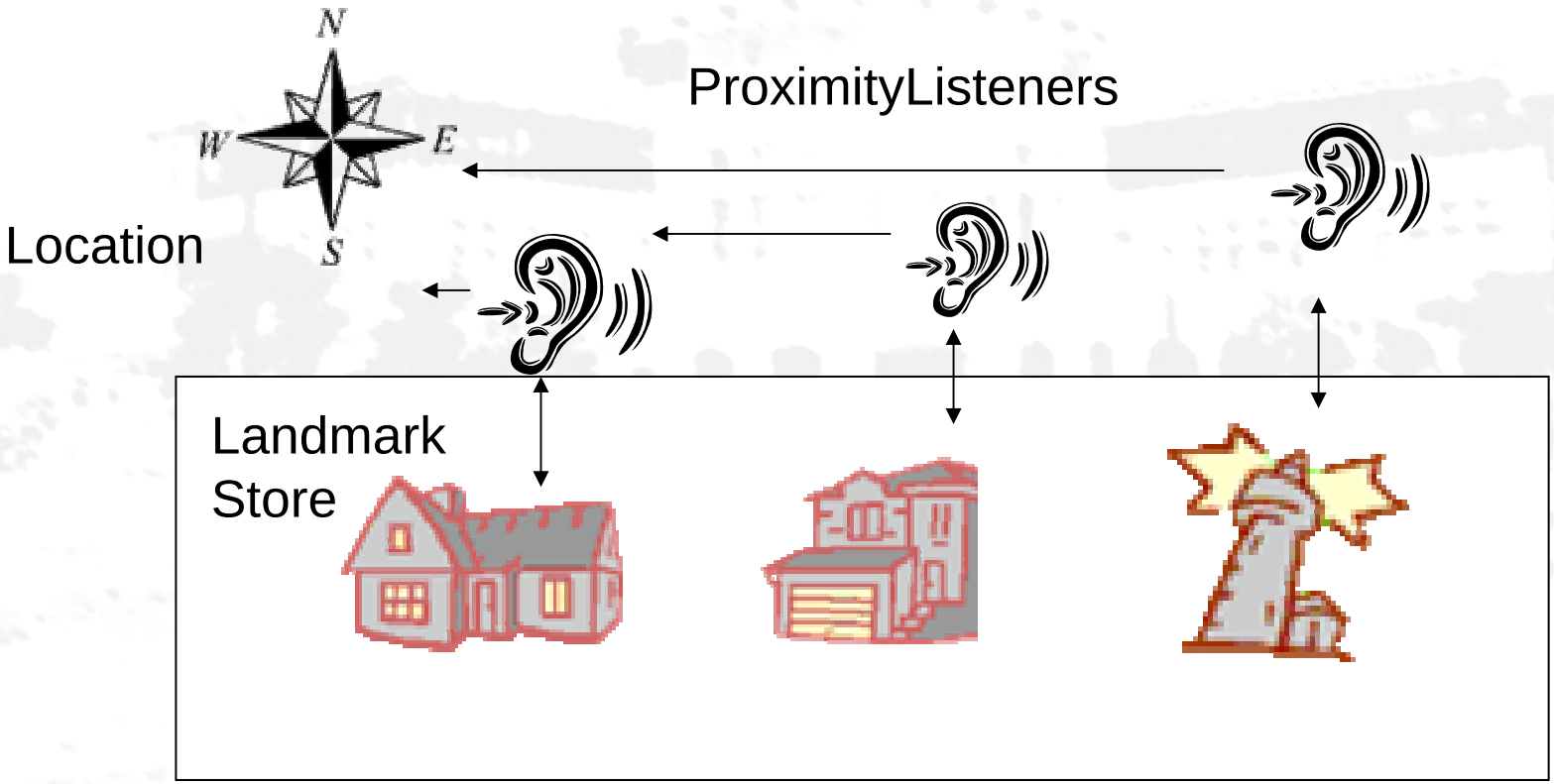
Massey University

(Persistent) POI classes



Massey University

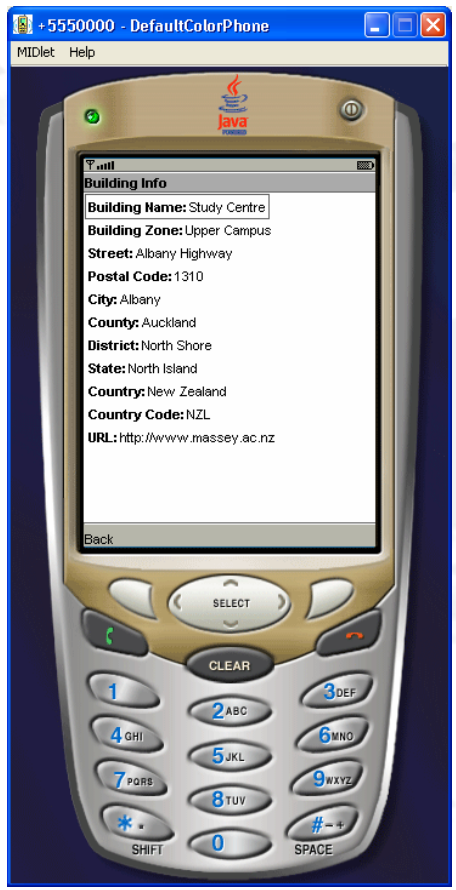
Proximity Listeners



Massey University

Location API Demo

- ProximityListeners attached to preloaded database of landmarks



Massey University

MIDP RMS

- Current implementation uses MIDP Record Management System
  - Relatively low level data management

```
db = RecordStore.openRecordStore(storeName, true);
ByteArrayOutputStream byteStream = new ByteArrayOutputStream();
DataOutputStream dataStream = new DataOutputStream(byteStream);
dataStream.writeUTF(recordData.toString());
byte[] landmarkArray = byteStream.toByteArray();
recordId = db.addRecord(landmarkArray, 0, landmarkArray.length);
recordIds.addElement(new Integer(recordId));
db.closeRecordStore();
```

Massey University

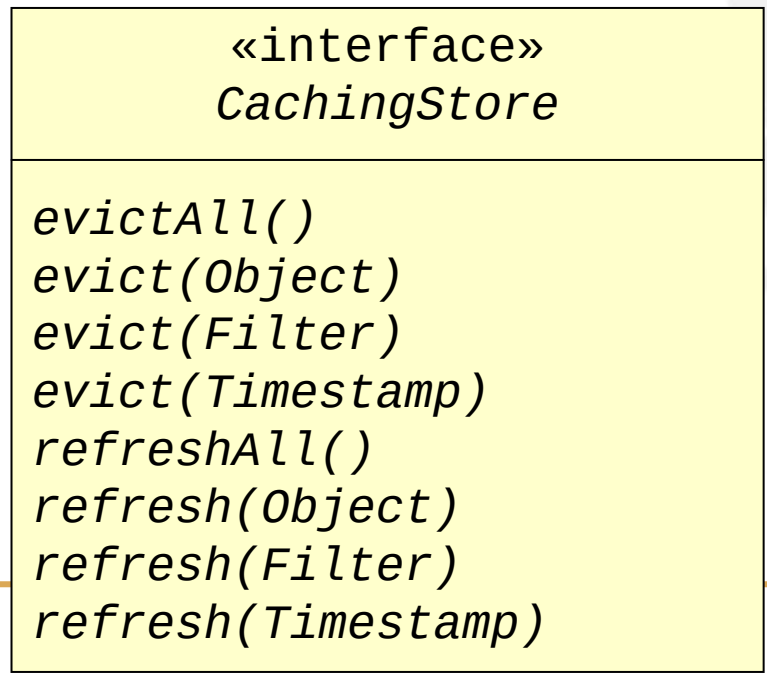
LandmarkStore API

LandmarkStore
createLandmarkStore(storeName : String) deleteLandmarkStore(storeName : String) getInstance(storeName : String) : LandmarkStore ListLandmarkStores() : String[] addCategory(categoryName : String) addLandmark(landmark : Landmark, categoryName : String) deleteCategory(categoryName : String) deleteLandmark(landmark : Landmark) getCategories() : Enumeration getLandmarks() : Enumeration getLandmarks(categoryName : String, minLatitude : double, maxLatitude : double, minLongitude : double, maxLongitude : double) : Enumeration getLandmarks(categoryName : String, name : String); removeLandmarkFromCategory(landmark : Landmark, categoryName : String) updateLandmark(landmark : Landmark)

Massey University

Possible role of JDO

- Simplify persistence related coding
- Support more useful access and queries
- Encapsulate facilities such as caching, hoarding etc.



Massey University

What about aspects?

- JDO is not a transparent persistence mechanism
  - There are still a number of explicit calls to the framework, even after byte code enhancement

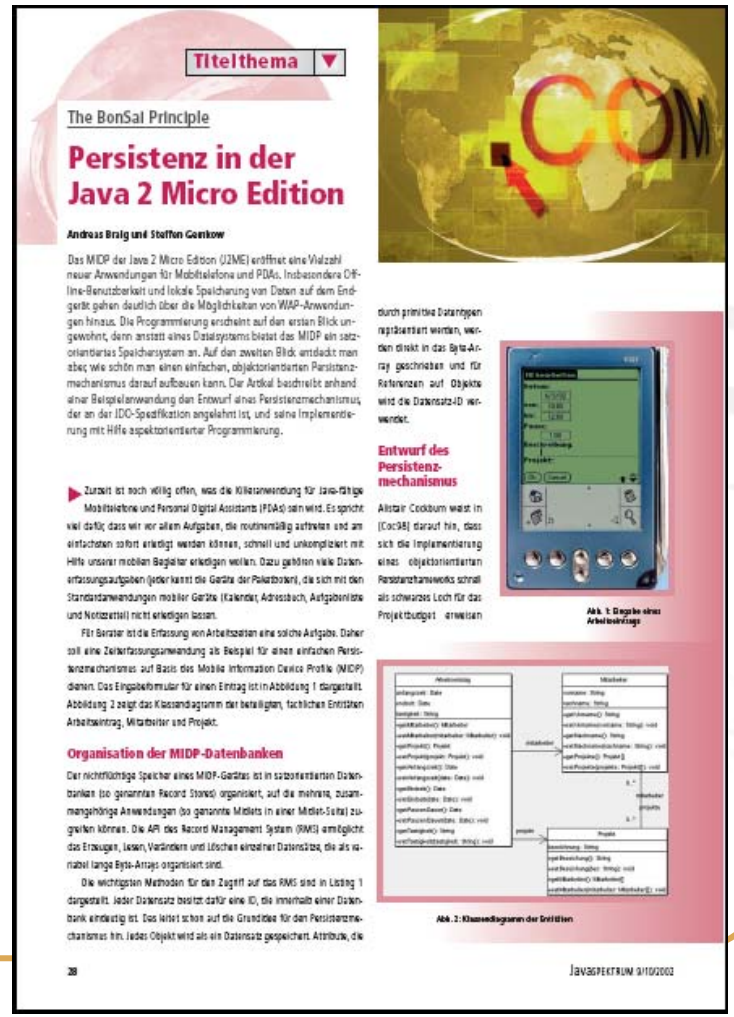
```
transaction = manager.currentTransaction();
if(!transaction.isActive()) {
        transaction.begin();
}
Object id = manager.getTransactionalObjectId(xxx);
MyObj transactionalObj = (MyObj)manager.getObjectById(id, false);
transactionalObj.addObj(yyy);
transaction.commit();
```

Application related code!

Massey University

J2ME + JDO + Aspects

- Briag & Gemkow (2002) proposed a J2ME persistence framework based on JDO and Aspects
- Only published in German



Titelthema

The BonSai Principle

Persistenz in der Java 2 Micro Edition

Andreas Briag und Steffen Gemkow

Das MIDP der Java 2 Micro Edition (J2ME) eröffnet eine Vielzahl neuer Anwendungen für Mobiltelefone und PDAs. Insbesondere Offline-Benutzbarkeit und lokale Speicherung von Daten auf dem Endgerät gehen deutlich über die Möglichkeiten von WAP-Anwendungen hinaus. Die Programmierung erscheint auf den ersten Blick ungewohnt, denn anstatt eines Dateisystems bietet das MIDP ein satzorientiertes Speichersystem an. Auf den zweiten Blick entdeckt man aber, wie schön man einen einfachen, objektorientierten Persistenzmechanismus darauf aufbauen kann. Der Artikel beschreibt anhand einer Beispielanwendung den Entwurf eines Persistenzmechanismus, der an der JDO-Spezifikation angelehnt ist, und seine Implementierung mit Hilfe aspektorientierter Programmierung.

Zurzeit ist noch völlig offen, was die Killeranwendung für Java-fähige Mobiltelefone und Personal Digital Assistants (PDAs) sein wird. Es spricht viel dafür, dass wir vor allem Aufgaben, die routinemäßig auftreten und am einfachsten sofort erledigt werden können, schnell und unkompliziert mit Hilfe unserer mobilen Begleiter erledigen wollen. Dazu gehören viele Datenerfassungsaufgaben (jeder kennt die Grüße der Paketboten), die sich mit den Standardanwendungen mobiler Geräte (Kalender, Adressbuch, Aufgabenliste und Notizzettel) nicht erledigen lassen.

Für Berater ist die Erfassung von Arbeitszeiten eine solche Aufgabe. Daher soll eine Zeiterfassungsanwendung als Beispiel für einen einfachen Persistenzmechanismus auf Basis des Mobile Information Device Profile (MIDP) dienen. Das Eingabeformular für einen Eintrag ist in Abbildung 1 dargestellt. Abbildung 2 zeigt das Klassendiagramm der beteiligten, fachlichen Entitäten Arbeitseintrag, Mitarbeiter und Projekt.

Organisation der MIDP-Datenbanken

Der nichtflüchtige Speicher eines MIDP-Gerätes ist in satzorientierten Datenbanken (so genannten Record Stores) organisiert, auf die mehrere, zusammengehörige Anwendungen (so genannte Midlets in einer Midlet-Suite) zugreifen können. Die API des Record Management System (RMS) ermöglicht das Erzeugen, Lesen, Verändern und Löschen einzelner Datensätze, die als variabel lange Byte-Arrays organisiert sind.

Die wichtigsten Methoden für den Zugriff auf das RMS sind in Listing 1 dargestellt. Jeder Datensatz besitzt dafür eine ID, die innerhalb einer Datenbank eindeutig ist. Das leitet schon auf die Grundidee für den Persistenzmechanismus hin. Jedes Objekt wird als ein Datensatz gespeichert. Attribute, die durch primitive Datentypen repräsentiert werden, werden direkt in das Byte-Array geschrieben und für Referenzen auf Objekte wird die Datensatz-ID verwendet.

Entwurf des Persistenzmechanismus

Alistair Cockburn weist in [Coc98] darauf hin, dass sich die Implementierung eines objektorientierten Persistenzframeworks schnell als schwarzes Loch für das Projektbudget erweisen

Abb. 1: Eingabe eines Arbeitseintrags

Abb. 2: Klassendiagramm der Entitäten

Massey University

Useful ideas

- The implementation simplifies the persistence code abstracting the RMS access behind a JDO style interface
- AspectJ removes the explicit persistence calls (in this example the non-JDO load and save methods) from the application code

Massey University

Issues

- Only a partial implementation of JDO APIs
- Does not include some core features
- JUnit tests are limited in scope
- Demo not deployed onto a mobile platform
- Does not address dependency issues of either JDO or AspectJ in a MIDP environment

Massey University

Further work

- A JDO MIDP platform
  - Kodo ME?
- How to integrate Aspects
  - A CLDC compliant runtime?

Massey University

Related Papers

- Braig and Gemkow, The BonSai Principle - Persistenz in der Java 2 Micro Edition, *Java Spektrum*, September 2002
- Parsons et al, 2005, Java Technology for the Wireless Industry – Toys or Tools? *Java Developers Journal*, 10(1)
- Parsons, 2005, Extending the Client Side Object Model in the Location API for J2ME, MOS Workshop, *ECOOP 2005 Workshop Reader*
- Parsons 2006, Exploring the Location API for J2ME, *Dr. Dobbs Journal* January 2006