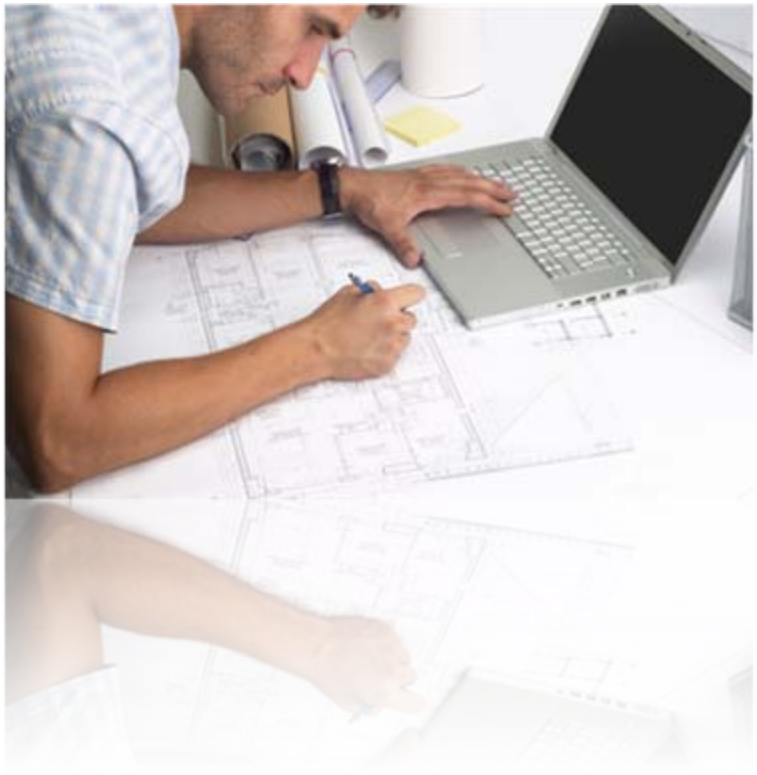

Coherence Code Examples

(c) Copyright 2007. Oracle Corporation

Clustering Java Processes

- Joins an existing cluster or forms a new cluster
  - Time “to join” configurable

```
Cluster cluster = CacheFactory.ensureCluster();
```

- `cluster` contains information about the Cluster
  - Cluster Name
  - Members
  - Locations
  - Processes
- No “master” servers
- No “server registries”

(c) Copyright 2007. Oracle Corporation

Leaving a Cluster

- Leaves the current cluster
- `shutdown` blocks until "data" is safe

- Failing to call `shutdown` results in Coherence having to detect process death/exit and recover information from another process.

- Death detection and recovery is automatic

```
CacheFactory.shutdown();
```

(c) Copyright 2007. Oracle Corporation

Using a Cache
get, put, size & remove

- `CacheFactory` resolves cache names (ie: `"mine"`) to configured `NamedCache`s
- `NamedCache` provides data topology agnostic access to information
- `NamedCache` interfaces implement several interfaces;
  - java.util.Map, Jcache, ObservableMap*, ConcurrentMap*, QueryMap*, InvocableMap*

```
NamedCache nc = CacheFactory.getCache("mine");

Object previous = nc.put("key", "hello world");

Object current = nc.get("key");

int size = nc.size();

Object value = nc.remove("key");
```

Coherence* Extensions

(c) Copyright 2007. Oracle Corporation

Using a Cache
keySet, entrySet, containsKey

- Using a `NamedCache` is like using a java.util.Map
- What is the difference between a Map and a Cache data-structure?
  - Both use (key,value) pairs for entries
  - Map entries don't expire
  - Cache entries may expire
  - Maps are typically limited by heap space
  - Caches are typically size limited (by number of entries or memory)
  - Map content is typically in-process (on heap)

```
NamedCache nc = CacheFactory.getCache("mine");

Set keys = nc.keySet();

Set entries = nc.entrySet();

boolean exists = nc.containsKey("key");
```

(c) Copyright 2007. Oracle Corporation

Observing Cache Changes
ObservableMap

- Observe changes in real-time as they occur in a `NamedCache`
- Options exist to optimize events by using Filters, (including pre and post condition checking) and reducing on-the-wire payload (Lite Events)
- Several `MapListeners` are provided out-of-the-box.
  - Abstract, Multiplexing...

```
NamedCache nc = CacheFactory.getCache("stocks");

nc.addMapListener(new MapListener() {
    public void onInsert(MapEvent mapEvent) {
    }

    public void onUpdate(MapEvent mapEvent) {
    }

    public void onDelete(MapEvent mapEvent) {
    }
 });
```

(c) Copyright 2007. Oracle Corporation

Querying Caches
QueryMap

- Query `NamedCache` keys and entries across a cluster (Data Grid) in parallel* using Filters
- Results may be ordered using natural ordering or custom comparators
- Filters provide support almost all SQL constructs
- Query using non-relational data representations and models
- Create your own Filters

```
NamedCache nc = CacheFactory.getCache("people");

Set keys = nc.keySet(
                  new LikeFilter("getLastName",
                                 "%Stone%"));
Set entries = nc.entrySet(
                  new EqualsFilter("getAge",
                                   35));
```

* Requires Enterprise Edition or above

ORACLE

(c) Copyright 2007. Oracle Corporation

Continuous Observation
Continuous Query Caches

- `ContinuousQueryCache` provides real-time and in-process copy of filtered cached data
- Use standard or your own custom Filters to limit view
- Access to “view”of cached information is instant
- May use with MapListeners to support rendering real-time local views (aka: Think Client) of Data Grid information.

```
NamedCache nc = CacheFactory.getCache(“stocks”);


NamedCache expensiveItems =
    new ContinuousQueryCache(nc,
        new GreaterThan(“getPrice”, 1000));
```

(c) Copyright 2007. Oracle Corporation

Aggregating Information
InvocableMap

- Aggregate values in a `NamedCache` across a cluster (Data Grid) in parallel* using Filters
- Aggregation constructs include; Distinct, Sum, Min, Max, Average, Having, Group By
- Aggregate using non-relational data models
- Create your own aggregators

```
NamedCache nc = CacheFactory.getCache("stocks");


Double total = (Double)nc.aggregate(
    AlwaysFilter.INSTANCE,
    new DoubleSum("getQuantity"));


Set symbols = (Set)nc.aggregate(
    new EqualsFilter("getOwner", "Larry"),
    new DistinctValue("getSymbol"));
```

* Requires Enterprise Edition or above

(c) Copyright 2007. Oracle Corporation

Mutating Information
InvocableMap

- Invoke `EntryProcessors` on zero or more entries in a `NamedCache` across a cluster (Data Grid) in parallel* (using Filters) to perform operations
- Execution occurs where the entries are managed in the cluster, not in the thread calling `invoke`
- This permits **Data + Processing Affinity**

* Requires Enterprise Edition or above

```
NamedCache nc = CacheFactory.getCache("stocks");


nc.invokeAll(
    new EqualsFilter("getSymbol", "ORCL"),
    new StockSplitProcessor());


 ...


class StockSplitProcessor extends
      AbstractProcessor {

    Object process(Entry entry) {
        Stock stock = (Stock)entry.getValue();
        stock.quantity *= 2;
        entry.setValue(stock);
        return null;
    }
}
```

(c) Copyright 2007. Oracle Corporation