API Design

Stephen Hicks

Interfaces, pointless classes

- Interfaces, not classes
  - Interchangeable implementations
  - Factories to define constructor behaviour
    - Specify factory implementation at run-time
- “Pointless” classes
  - TimeInterval
    - Replace with primitive type
    - Add functionality

Naming, types

- Consistent naming
  - registerXXXQuery / removeConsumer
  - getRelevantConsumers / getProducerConnections
- Use more restrictive types
  - Consumer(…, int queryType) is prone to misuse
  - Consumer(…, QueryType qt)
    - Allows only valid query types
    - Clarifies what is expected in the parameter

QueryType

- public class QueryType {
  - private static final int C = 1;
  - …
  - private static final int H = 8;
  - public static final QueryType CONTINUOUS = new QueryType(C);
  - …
  - public static final QueryType HISTORY = new QueryType(H);
  - private int m_queryType;
  - protected QueryType(int p_queryType) {…}
  - public boolean equals(QueryType p_qt) {…}
- }

Parameters, collection types

- Combine parameters into a single object
  - registerProducerTable
    - 8 parameters become…
    - …(ProducerRegistration pr, EndpointReference producer, TerminationTime tt)
- Avoid generic collection types
  - E.g.
    - Consumer(…, Vector scs, …)
    - Consumer(…, EndpointReferenceList erl, …)
  - Language-neutral
  - Clearer API

General points

- Only use ResultSets where appropriate
  - Just pop(), blockingPop() etc.
- Remove unnecessary abstraction
  - Schema API uses
    - Table ID
    - Column ID
  - translateTableName() could…
    - become validateTableName() or
    - be removed
- Combine methods to minimise remote calls
  - e.g. in Schema, to retrieve column types and names, given a table name:
    - Currently requires 4 remote calls (translateTableName, getTableInfo, translateColumnNames, getColumnTypes)
    - Should only require 1 (getColumns)

SQL

- Abstract away from SQL 

  - E.g.
    - createConsumer(String select, …)
    - createConsumer(Query query, …)
  - Query class contains
    - Column names
    - Table names
    - Details of predicate
  - Subclasses provide SQL parsing etc.
    - SQLSelectQuery extends Query
      - createConsumer(new SQLSelectQuery(“SELECT * FROM GlueSE”), …)
    - XPathQuery…

API Hierarchy

- APIBase (getStatus, …)
  - ?Registry (getRelevantConsumers, …)
  - ?Schema (getSchemaTables, …)
  - Resource (TerminationInterval, close, …)
    - Consumer (pop, start, …)
    - Declarable (declareTable, …)
      - CanonicalProducer
      - Republisher
      - Producer (insert, …)