ChangeLog

This is the list of changes for maintenance of JSR-243.

IssueLog has the list of changes requested by the community.

PROPOSED

1. Run the TCK using maven 2 instead of maven 1. Add a special maven project exectck that executes the tck tests in various configurations.
2. Use Java 6 `ServiceLoader` to find a JDO `PersistenceManagerFactory` implementation.
3. Support `List.get(int)` in JDOQL query.
4. Change the return type for avg in JDOQL query.
5. Add methods to JDOQL query:
    a. `charAt(int)` applies to `String` type
    b. `startsWith(String, int)` applies to `String` type
    c. `length()` applies to `String` type
    d. `trim()` applies to `String` type
    e. `ordinal()` applies to `Enum` types
    f. `toString()` applies to `Enum` types
    g. `getHour()` applies to `java.util.Date` type
    h. `getMinutes()` applies `to java.util.Date` type
    i. `getSeconds()` applies `to java.util.Date` type
    j. `getDay()` applies to `java.util.Date` type
    k. `getMonth()` applies to `java.util.Date` type
    l. `getYear()` applies to `java.util.Date` type
    m. `Math.cos(number)` static method in java.lang.Math, applies to `double` type
    n. `Math.sin(number)` static method in java.lang.Math, applies to `double` type
    o. `Math.tan(number)` static method in java.lang.Math, applies to `double` type
    p. `JDOHelper.getVersion(Object)` static method in `JDOHelper`, allows using the version of an instance directly in a query
6. Allow specifying position of fields in generated schema using the "position" attribute or annotation.
7. Support mapping option "complete-table" in which class is represented by a single table containing all fields from all superclasses.
8. Provide a way to put properties to individual `PersistenceManager` instances.
9. Provide proper OSGi metadata in manifest to enable JDO to be included as OSGi bundles.
10. Add `initialValue` and `allocationSize` to `Sequence` metadata (annotations and xml).
11. Provide a boolean property to serialize access to objects. Add methods to set and clear the property. Add xml metadata to specify the property per class or interface.

ACCEPTED

1. Allow multiple fields (properties) to be used for ordering persistent Lists or Maps.
2. Specify the behavior at flush time if the user has inserted elements or entries out of order in ordered fields (properties).
3. Allow the user to create fetch groups dynamically.
4. Require `SingleFieldIdentity` classes to implement `Comparable` so they can be used in non-persistent ordered Set or Map instances.
5. Add `ReadOnly` to `PersistenceManagerFactory` to allow fail-fast on applications that use a read-only data store.
6. Provide a way to specify the isolation level of connections used for a `PersistenceManagerFactory`.
7. Add an invocation API for a run-time enhancer. This allows for a standard command-line tool and standard IDE plugins for enhancement.
8. Require a no-args constructor for implementations of `PersistenceManagerFactory` and add the `getPersistenceManagerFactory`... methods to the `PersistenceManagerFactory` interface. This change allows `JDOHelper` to use Java 6 `ServiceLoader` in future.
9. Add methods to `PersistenceManager` and `PersistenceManagerFactory` to allow a user to get the current time and date from the server. Allow the `PersistenceManagerFactory` to be configured statically with `TimeZone` and Locale in case the information is not available from the server itself.
10. Clarify 18.15.8 that the ordering field should be managed by the implementation, not by the user.
11. Remove the "serialized" attribute from 18.15.8 since the ordering column is not serialized.
12. Add property `CopyOnAttach` to `PersistenceManager` and `PersistenceManagerFactory`. With the property set to true, makes a copy of the detached instance on `makePersistent`. If the property is set to false, it attaches the detached instance itself.
13. Add `ObjectState` enum and convenience methods in `JDOHelper` to return the `ObjectState` of an instance.
14. Add method `void evictAll (boolean subclasses, Class pcClass)` to `PersistenceManager`. This allows to evict instances for a particular class. Change signatures of `DataStoreCache` methods pinAll, unpinAll, and evictAll from `(Class pcClass, boolean subclasses)` to `(boolean subclasses, Class pcClass)` to accommodate adding methods that change the pcClass parameter from Class to Class....
15. Clarify the behavior of method `evictAll` if the parameter is a persistent interface. This comment also applies to the `DataStoreCache` interface method `void evictAll (Class pcClass, boolean subclasses)`.
16. Require that the implementation not hold a strong reference to flushed dirty instances, allowing these instances to be garbage collected.
17. Change signatures of `PersistenceManager` methods that take `Object[]` as an argument to take `Object...` as an argument. These signatures are source and binary compatible with existing programs. Where the `Object...` parameter would not be the last parameter, deprecate the method and reorder the parameters so `Object...` is last.
18. In 5.4.1, Compound Identity should be updated to reflect that for key fields of reference types, the type of the key field is the reference type in the class but the oid of the reference type in the oid class.
19. In Table 2: State Transitions, the transition for a transient-dirty instance during commit with `DetachAllOnCommit = true` should be to transient-clean. Also, state changes need to be added for detach methods and serialization.

20. In 5.5.8 and 5.5.9, `detachCopy` should be removed from the list of methods that throw exceptions if applied to detached-clean or detached-dirty instances.
21. In 6.4.3, change "Portable JDO applications must not depend on whether instances of these classes are treated as SCOs or FCOs. " to "Portable JDO applications must not depend on SCO or FCO uniquing behavior, nor on the storage mechanism in the datastore. Portable applications may use the same instance of these classes as field values in any persistence-capable class instance."
22. In section 7.5, change "If the class is abstract, null is returned." to "If the class is abstract, `JDOFatalInternalException` is thrown".
23. In section 7.5, add `public byte fetchByteField(int fieldNumber);` to `ObjectIdFieldSupplier`.
24. In section 7.5, add `public byte storeByteField(int fieldNumber, byte value);` to `ObjectIdFieldConsumer`.
25. In Chapter 8, add after `class JDOHelper {`
    - `public JDOHelper();`
      For some usage patterns, an instance of `JDOHelper` on which to invoke methods is preferable to the use of static methods. For this purpose, a public constructor is provided.
26. In Chapter 8, add a convenience method that returns a `PersistenceManager` proxy that can be used in web and ejb containers to dynamically bind to the transaction-associated `PersistenceManager`.
27. In Chapter 9, add section on managing date formatting for `ObjectIdentity` constructors.
    - `public synchronized void registerDateFormat(java.text.DateFormat df);`
28. In 9.4, add method to retrieve persistence-capable classes that have been registered.
    - `public java.util.Collection getRegisteredClasses();`
29. In 9.5, add method to verify that the class is authorized to be a state manager.
    - `public static void checkAuthorizedStateManagerClass(Class smClass);`
30. In 9.5, add method to register multiple state manager classes.
    - `public static void registerAuthorizedStateManagerClasses(java.util.Collection smClasses) throws SecurityException;`
31. In Chapter 11, add properties for configuring `PersistenceManagerFactory` that are consistent with JPA specification of `TransactionType` and Persistence Unit Name.
32. In Chapter 12, specify the behavior of `PersistenceManager` if it extends `Serializable` and `writeObject` is called.
33. In 12.6.6, clarify that a `JDOUserException` will be thrown when invoking `newInstance`: if a class is not persistence-capable, or does not declare a public no-args constructor; if an interface is not persistence-capable or declares methods that are not defined as persistent properties; if an abstract class is not persistence-capable or declares abstract methods that are not defined as persistent properties.
34. In 12.6.8, section heading Explicit Detach, the sentence "If the parameter instance is detached, then `JDOUserException` is thrown. " should be removed.
35. In 12.6.8, add a note that serialization for storage using the serialized, serialized-element, serialized-key, or serialized-value metadata attributes does not create a detached instance.
36. In 12.6.8, clarify the behavior of instances during serialization both with and without an active transaction.
37. In 12.7.5, change `public int setMaxFetchDepth();` to `public int getMaxFetchDepth();`
38. In 12.7.5, specify that getFetchGroups returns a read-only copy of the active Fetch Groups.
39. In 12.7.6, p. 127, change "A recursion-depth of 0 will fetch the whole graph of instances reachable from this field." to "A recursion-depth of -1 will fetch the whole graph of instances reachable from this field."
40. In 12.7.6 p. 129, change `fetch-depth` to `recursion-depth` in the example.
41. In 14, add subqueries to permit e.g. select from Employee where this.salary > (select avg(salary) from Employee)
42. In 14.6.2 p. 159, the sectionheading "Methods" is not marked as a sectionheading.
43. Add to 14.6.9: Projected SCOs are never owned, projected FCOs are always managed. Modifying unowned SCOs never has an effect on the database. Modifying FCOs no matter how you get them always has an effect if the tx commits.
44. In 15.3, p. 187, the third paragraph, beginning "The field on the other side" and ending "in the next transaction", is duplicated and will be removed.
45. In 15.3, add text to describe updating the other side of relationships where this side is deleted. This maintains referential integrity for delete as well as update.
46. Add to 17.1.11
    - `JDOUserCallbackException extends javax.jdo.JDOUserException`
47. In Chapter 18, add an xml element to specify the fetch plan to use for a query.
48. In 18.15.1, change "(e.g. a field of type Object can specify field-type="Integer")." to "(e.g. an element of type Object can specify element-type="Integer").
49. In Chapter 18, add to .jdo metadata:
    - `<!ATTLIST property field-type CDATA #IMPLIED>`
50. Add to 18.15.1 "The default for dependent-element is false."
51. Add to 18.15.2 "The default for dependent-key is false."
52. Add to 18.15.2 "The default for dependent-value is false."
53. Change the last paragraph of 21.6 from "For Detachable classes, the results of restoring a serialized persistent instance graph is a graph of interconnected detached instances that might be attached via the attachCopy methods." to "For Detachable classes, the results of restoring a serialized persistent instance graph is a graph of interconnected detached instances that might be attached via the makePersistent methods."
54. Change 21.13 from "Some methods require a non-null state manager. In these cases, if the `jdoStateManager` is null, then `IllegalStateException` is thrown." to "Some methods require a non-null state manager. In these cases, if the `jdoStateManager` is null, then `JDOFatalInternalException` is thrown."
55. Change 21.21.7 Generated jdoGetManagedFieldCount sample implementation to avoid using the jdoFieldNames field that might be initialized after it is used during initialization of a subclass.

```
protected static int jdoGetManagedFieldCount () {
   return jdoFieldNames.length;
}
```

to

```
protected static int jdoGetManagedFieldCount () {
   return <enhancer-generated constant>;
}
```

56. Add to Chapter 23, constants defined in the `JDOPermission` class:

```
public static final javax.jdo.spi.JDOPermission CLOSE_PERSISTENCE_MANAGER_FACTORY =
"closePersistenceManagerFactory";
public static final javax.jdo.spi.JDOPermission GET_METADATA = "getMetadata";
public static final javax.jdo.spi.JDOPermission MANAGE_METADATA = "manageMetadata";
public static final javax.jdo.spi.JDOPermission SET_STATE_MANAGER = "setStateManager";
```

57. Provide an API to dynamically create and access metadata equivalent to specifying metadata via annotations or xml.
58. Change the meaning of the validate flag in getObjectById to allow users to assert that the class of the oid is the exact class of the datastore instance.
59. Add extensions[] to the @Order annotation to mirror the xml.
60. Deprecate the api2-legacy and tck2-legacy projects as these are for use with JDK 1.4 and no longer need to be maintained.
61. Add an API to cancel a running query.
62. Add an API and PMF properties to get and set datastore read and write timeout values.

JDK 1.5 changes

These are changes proposed for JDO 2 to better support JDK 1.5.

1. Add to 6.3 a section requiring support for enum types, including subclasses of enum types.
2. Change in 12.6 signatures of the following `PersistenceManager` methods to be generic. Note that these changes are source compatible with existing application programs.

JDO 2.0	JDO 2.0 Maintenance Release		
Object getObjectById (Class cls, Object key)	T getObjectById (Class<T> cls, Object key)		
Object newInstance(Class persistenceCapable)	T newInstance(Class<T> persistenceCapable)		
Object makePersistent (Object pc)	T makePersistent (T pc)		
<ac:structured-macro ac:name="unmigrated-wiki-markup" ac:schema-version="1" ac:macro-id="b198fa66-bf18-4b3b-9ddf-034662633d50"><ac:plain-text-body><![CDATA[	Object[] makePersistentAll (Object[] pcs)	T[] makePersistentAll (T[] pcs)	]]></ac:plain-text-body></ac:structured-macro>
Collection makePersistentAll (Collection pcs)	Collection<T> makePersistentAll (Collection<T> pcs)		
Object detachCopy(Object pc)	T detachCopy(T pc)		
Collection detachCopyAll(Collection pcs)	Collection<T> detachCopyAll (Collection<T> pcs)		
<ac:structured-macro ac:name="unmigrated-wiki-markup" ac:schema-version="1" ac:macro-id="4782e666-09c9-4847-8921-3732cf301522"><ac:plain-text-body><![CDATA[	Object[] detachCopyAll(Object [] pcs)	T[] detachCopyAll(T [] pcs)	]]></ac:plain-text-body></ac:structured-macro>

3. Add to 15.1 a paragraph describing that mapping an enum to a fixed-precision numeric type uses the ordinal() value for storage; mapping to a character column type (CHAR, VARCHAR, etc.) uses the name() value for storage; mapping to any other column type is an error.
4. Provide in a new chapter a set of annotations that map directly to xml elements as an alternative to using xml metadata. Describe how jdo implementations can use either JDO annotations or JPA annotations to provide metadata.
5. Provide interfaces that extend both JDO and JPA in order to make it easier to migrate applications from JDO to JPA.

DEFERRED

- none yet -