JSR 236 Status – Jan 28, 2013

Anthony Lai

Agenda

- Introduction
- Overview
- Current Status

Introduction

- Provides asynchronous capabilities to Java EE application components, using extension of Java SE Concurrency Utilities APIs from JSR 166.

Introduction
Provides mechanism for

- Definition and usage of centralized, manageable `java.util.concurrent.ExecutorService` objects in a Java EE application server.
- Usage of Java SE Concurrency Utilities in a Java EE application.
- Propagation of the Java EE container's runtime contextual information to other threads.
- Managing and monitoring the lifecycle of asynchronous operations in a Java EE Application Component.

Overview
Main Interfaces

- Extends existing Java SE concurrency utilities:
  - ManagedExecutorService
  - ManagedScheduledExecutorService
  - ManagedThreadFactory
- New interface
  - ContextService
- Provided to applications by Java EE Container
- Lookup via JNDI or @Resource
- Default objects at

  `java:comp/DefaultManagedExecutorService` etc...

ManagedExecutorService
Overview

- For running tasks asynchronously on threads provided by Java EE container
- Container context captured from submitting thread to be applied on execution thread
  - Configurable – typical: Classloading, JNDI namespace, security identity
  - Not transaction – tasks should use UserTransaction instead

ManagedExecutorService

APIs

- Extends from java.util.concurrent.ExecutorService
- No additional APIs
- Lifecycle APIs disabled – throws IllegalStateException
  - awaitTermination, isTerminated, isShutdown, shutdown, shutdownNow

ManagedExecutorService
APIs

```
void execute(Runnable command)

<T> Future<T> submit(Callable<T> task)
Future<?> submit(Runnable task)
<T> Future<T> submit(Runnable task, T result)

<T> List<Future<T>> invokeAll(Collection<?
extends Callable<T>> tasks)
<T> List<Future<T>> invokeAll(Collection<?
extends Callable<T>> tasks, long timeout,
TimeUnit unit)

<T> T invokeAny(Collection<? extends Callable<T>>
tasks)
<T> T invokeAny(Collection<? extends Callable<T>>
tasks, long timeout, TimeUnit unit)
```

ManagedExecutorService
Example

```
// Within your servlet or EJB method
@Resource(name="concurrent/myExecutor")
ManagedExecutorService mes;
void businessMethod() {
  Callable<Integer> c = new Callable<>() {
    Integer call() {
    // Interact with a database... Return answer.
    // The namespace is available here!
  }

  // Submit the task and do something else. The task
  // will run asynchronously on another thread.
  Future result = mes.submit(c);
  ...
  // Get the result when ready...
  int theValue = result.get();
  ...
```

ManagedScheduledExecutorService
Overview

- For scheduling tasks to run after a given delay, periodically, or at some custom schedule
- Extends from `ManagedExecutorService` and `java.util.concurrent.ScheduledExecutorService`

ManagedScheduledExecutorService
APIs from ScheduledExecutorService

```
<V> ScheduledFuture<V> schedule(Callable<V>
callable, long delay, TimeUnit unit)

ScheduledFuture<V> schedule(Runnable command,
long delay, TimeUnit unit)

ScheduledFuture<?> scheduleAtFixedFate(Runnable
command, long initialDelay, long period, TimeUnit
unit)

ScheduledFuture<?>
scheduledWithFixedDelay(Runnable command, long
initialDelay, long delay, TimeUnit unit)
```

ManagedScheduledExecutorService
Extension APIs

- Extension APIs for custom schedule support
  - `ScheduledFuture<?>` **`schedule`**`(Runnable command,` **`Trigger trigger`**`)`
  - `<V> ScheduledFuture<V>` **`schedule`**`(Callable<V> callable,` **`Trigger trigger`**`)`

ManagedScheduledExecutorService
Trigger

```
interface Trigger {
  // Return true if you want to skip the
  // currently-scheduled execution.
  boolean skipRun(LastExecution
lastExecutionInfo, Date scheduledRunTime);

  // Retrieves the time in which to run the task
  // next. Invoked during submit time and after
  // each task has completed.
  Date getNextRunTime(LastExecution
lastExecutionInfo, Date taskScheduledTime);
}
```

Managed[Scheduled]ExecutorService
ManagedTaskListener

- Listeners can be registered with the task when submitted to Managed[Scheduled]ExecutorService.
- APIs
  - taskSubmitted – The task was submitted to the executor
  - taskAborted – The task was unable to start or was cancelled.
  - taskStarting – The task is about to start
  - taskDone – The task has completed (successfully, exception, cancelled, aborted, or rejected)
- The listener method runs in unspecified context, but can be configured to run in the same container context as the task

ORACLE

Managed[Scheduled]ExecutorService

ManagedTask

- Any task submitted to an ManagedExecutorService or ManagedScheduledExecutorService can optionally implement `ManagedTask`
- APIs
  - `Map<String, String` **`getExecutionProperties`**`()`
  - `ManagedTaskListener` **`getManagedTaskListener`**`()`
- Execution properties
  - CONTEXTUAL_CALLBACK_HINT
  - IDENTITY_NAME
  - LONGRUNNING_HINT

ORACLE

ManagedTask
Registering ManagedTaskListener

```
// Runnable implements ManagedTask
public class TaskWithListener implements Runnable,
ManagedTask {
  ...
  public ManagedTaskListener getManagedTaskListener {
    return aManagedTaskListener;
  }
}
// Or use ManagedExecutors utility method to associate
// a ManagedTaskListener to a task
Runnable aTask;
ManagedTaskListener myTaskListner;
Runnable taskWithListener =
  ManagedExecutors.managedTask(aTask, myTaskListener);
// submit taskWithListener to a ManagedExecutorService
```

ManagedThreadFactory
Overview

- Method for applications to ask for threads from Java EE Product Provider
- API – same as `java.util.concurrent.ThreadFactory`
  - `Thread newThread(Runnable r)`
  - Threads also implement `ManagableThread` interface
- Container context captured at newThread call to be applied to thread that invokes r.run()
- Can be used with Java SE concurrency utilities APIs where ThreadFactory is needed. E.g. in `java.util.concurrent.Executors`

ManagedThreadFactory
Shutdown

- Thread interrupted when ManagedThreadFactory shuts down.
- Runnable should check `ManagableThread.isShutdown()` when interrupted, and clean up if it is true.

ORACLE

ManagedThreadFactory
Example

```
// Within your servlet or EJB method...
// Lookup the ManagedThreadFactory
InitialContext ctx = new InitialContext();
ManagedThreadFactory tf = (ManagedThreadFactory)
    ctx.lookup("java:comp/env/concurrent/myTF");

// Request a thread and start it.
Thread myThread = tf.newThread(myDaemonRunnable);
myThread.start();
// Container context such as JNDI namespace will
// be available on myThread where
// myDaemonRunnable.run() is invoked.
```

ORACLE

ManagedThreadFactory

Example – Use with ThreadPoolExecutor

```
// Within your servlet or EJB method...
// Lookup the ManagedThreadFactory
@Resource
ManagedThreadFactory tf;

void businessMethod() {
// Use a custom Java SE ThreadPoolExecutor
CustomThreadPoolExecutor pool =
new CustomThreadPoolExecutor(coreSize, maxSize, tf);

// When the executor allocates a new thread, the
// thread will use the current container context.
```

ContextService
Overview

- For applications to create contextual proxy objects to capture container context and run within that context later
- Uses dynamic proxy in java.lang.reflect package
- Cannot create proxy for objects managed by container, such as EJB.
- Customizable through executionProperties
  - USE_PARENT_TRANSACTION
  - Vendor specific properties, e.g. vendorA.security_token_expiration

ContextService
API

- For creating new contextual object proxy for the input object instance
  - `Object` **`createContextualProxy`**`(Object instance, Class<?>... Interfaces)`
  - `Object` **`createContextualProxy`**`(Object instance, Map<String,String> executionProperties, Class<?>... Interfaces)`
  - `<T> T` **`createContextualProxy`**`(T instance, Class<T> intf)`
  - `<T> T` **`createContextualProxy`**`(T instance, Map<String,String> executionProperties, Class<T> intf)`
- For returning the execution properties on the given contextual object proxy instance
  - `Map<String,String>` **`getExecutionProperties`**`(Object contextualProxy)`

ContextService
Example

```
// In application
public interface MessageProcessor {
  public void processMessage(Message msg)
  ...
}
// Within servlet or EJB method...
@Resource
ContextService ctxSvc;
void businessMethod() {
  MessageProcessor msgProcessor = ...
  // Wrap with the current context
  MessageProcessor proxy =
    ctxSvc.createContextualProxy (msgProcessor,
MessageProcessor.class}
  // Store the contextual proxy object somewhere for
  // running later..
  store.putIt(proxy);
  ...
```

ORACLE

ContextService
Example (contd.)

```
// Elsewhere, in a different thread, retrieve the
// MessageProcessor contextual proxy object from the
// store
MessageProcessor proxy = store.getIt();

// The proxy method processMessage() is invoked on
// this thread, but with the context of the servlet or
// EJB that created it.
proxy.processMessage(msg);
```

Current Status
Open Issues

- Remove Optional sections
  - Proposed removal of Distributed ManagedExecutorService an JSR 77-based Managed objects to EG
- Alignment with other Java EE specs
  - With asynchronous support in servlets?
  - With asynchronous methods in EJB?
  - CDI
    - @Inject ManagedExecutorService?
    - Tasks run within scopes?
  - Others specs?
  - Most likely for next release…

ORACLE

Current Status
Open Issues

- Fork/Join support
  - Deferred
- New API in ContextService to pass application objects to execution thread for JSR 359 – SIP Servlets 2.0
  - Ongoing EG discussion

Current Status
Implementation

- Ongoing
  - Maven repository deployment/Build integration
    - Both API and RI artifacts
  - RI implementation (some loose ends)
  - SPI implementation for GlassFish
    - SPI in RI for integrating with Java EE containers
  - Configuration/CLI
  - Console support
    - Working with console team
- Not Started
  - Deployer

ORACLE

Resources

- JSR 236 page
  - http://jcp.org/en/jsr/detail?id=236
- JSR 236 javadoc
  - http://concurrency-ee-spec.java.net/javadoc/
- java.net projects
  - http://concurrency-ee-spec.java.net
  - http://java.net/projects/cu-javaee
- One Pager containing GlassFish configuration and CLI commands
  - http://aseng-wiki.us.oracle.com/asengwiki/x/84B6IP----8