Simple Singleton

Example simple-singleton can be browsed at https://github.com/apache/tomee/tree/master/examples/simple-singleton

As the name implies a `javax.ejb.Singleton` is a session bean with a guarantee that there is at most one instance in the application.

What it gives that is completely missing in EJB 3.0 and prior versions is the ability to have an EJB that is notified when the application starts and notified when the application stops. So you can do all sorts of things that you previously could only do with a load-on-startup servlet. It also gives you a place to hold data that pertains to the entire application and all users using it, without the need for a static. Additionally, Singleton beans can be invoked by several threads at one time similar to a Servlet.

See the Singleton Beans page for a full description of the javax.ejb.Singleton api.

The Code

PropertyRegistry <small>Bean-Managed Concurrency</small>

Here we see a bean that uses the Bean-Managed Concurrency option as well as the @Startup annotation which causes the bean to be instantiated by the container when the application starts. Singleton beans with @ConcurrencyManagement(BEAN) are responsible for their own thread-safety. The bean shown is a simple properties "registry" and provides a place where options could be set and retrieved by all beans in the application.

```
package org.superbiz.registry;

import javax.annotation.PostConstruct;
import javax.annotation.PreDestroy;
import javax.ejb.ConcurrencyManagement;
import javax.ejb.Singleton;
import javax.ejb.Startup;
import java.util.Properties;

import static javax.ejb.ConcurrencyManagementType.BEAN;

@Singleton
@Startup
@ConcurrencyManagement(BEAN)
public class PropertyRegistry {

    // Note the java.util.Properties object is a thread-safe
    // collections that uses synchronization.  If it didn't
    // you would have to use some form of synchronization
    // to ensure the PropertyRegistryBean is thread-safe.
    private final Properties properties = new Properties();

    // The @Startup annotation ensures that this method is
    // called when the application starts up.
    @PostConstruct
    public void applicationStartup() {
        properties.putAll(System.getProperties());
    }

    @PreDestroy
    public void applicationShutdown() {
        properties.clear();
    }

    public String getProperty(final String key) {
        return properties.getProperty(key);
    }

    public String setProperty(final String key, final String value) {
        return (String) properties.setProperty(key, value);
    }

    public String removeProperty(final String key) {
        return (String) properties.remove(key);
    }
}
```

ComponentRegistry <small>Container-Managed Concurrency</small>

Here we see a bean that uses the Container-Managed Concurrency option, the default. With @ConcurrencyManagement(CONTAINER) the container controls whether multi-threaded access should be allowed to the bean (`@Lock(READ)`) or if single-threaded access should be enforced (`@Lock(WRITE)`).

```
package org.superbiz.registry;

import javax.ejb.Lock;
import javax.ejb.Singleton;
import java.util.ArrayList;
import java.util.Collection;
import java.util.HashMap;
import java.util.Map;

import static javax.ejb.LockType.READ;
import static javax.ejb.LockType.WRITE;

@Singleton
@Lock(READ)
public class ComponentRegistry {

    private final Map<Class, Object> components = new HashMap<Class, Object>();

    public <T> T getComponent(final Class<T> type) {
        return (T) components.get(type);
    }

    public Collection<?> getComponents() {
        return new ArrayList(components.values());
    }

    @Lock(WRITE)
    public <T> T setComponent(final Class<T> type, final T value) {
        return (T) components.put(type, value);
    }

    @Lock(WRITE)
    public <T> T removeComponent(final Class<T> type) {
        return (T) components.remove(type);
    }
}
```

Unless specified explicitly on the bean class or a method, the default `@Lock` value is `@Lock(WRITE)`. The code above uses the `@Lock(READ)` annotation on bean class to change the default so that multi-threaded access is granted by default. We then only need to apply the `@Lock(WRITE)` annotation to

the methods that modify the state of the bean.

Essentially `@Lock(READ)` allows multithreaded access to the Singleton bean instance unless someone is invoking an `@Lock(WRITE)` method. With `@Lock(WRITE)`, the thread invoking the bean will be guaranteed to have exclusive access to the Singleton bean instance for the duration of its invocation. This combination allows the bean instance to use data types that are not normally thread safe. Great care must still be used, though.

In the example we see `ComponentRegistryBean` using a `java.util.HashMap` which is not synchronized. To make this ok we do three things:

1. Encapsulation. We don't expose the HashMap instance directly; including its iterators, key set, value set or entry set.
2. We use `@Lock(WRITE)` on the methods that mutate the map such as the `put()` and `remove()` methods.
3. We use `@Lock(READ)` on the `get()` and `values()` methods as they do not change the map state and are guaranteed not to be called at the same as any of the `@Lock(WRITE)` methods, so we know the state of the HashMap is no being mutated and therefore safe for reading.

The end result is that the threading model for this bean will switch from multi-threaded access to single-threaded access dynamically as needed, depending on the method being invoked. This gives Singletons a bit of an advantage over Servlets for processing multi-threaded requests.

See the Singleton Beans page for more advanced details on Container-Managed Concurrency.

Testing

ComponentRegistryTest

```
package org.superbiz.registry;

import org.junit.AfterClass;
import org.junit.Assert;
import org.junit.Test;

import javax.ejb.embeddable.EJBContainer;
import javax.naming.Context;
import java.net.URI;
import java.util.Collection;
import java.util.Date;

public class ComponentRegistryTest {

    private final static EJBContainer ejbContainer = EJBContainer.createEJBContainer(
);

    @Test
```

```
    public void oneInstancePerMultipleReferences() throws Exception {

        final Context context = ejbContainer.getContext();

        // Both references below will point to the exact same instance
        final ComponentRegistry one = (ComponentRegistry) context.lookup(
"java:global/simple-singleton/ComponentRegistry");
        final ComponentRegistry two = (ComponentRegistry) context.lookup(
"java:global/simple-singleton/ComponentRegistry");

        final URI expectedUri = new URI("foo://bar/baz");
        one.setComponent(URI.class, expectedUri);
        final URI actualUri = two.getComponent(URI.class);
        Assert.assertSame(expectedUri, actualUri);

        two.removeComponent(URI.class);
        URI uri = one.getComponent(URI.class);
        Assert.assertNull(uri);

        one.removeComponent(URI.class);
        uri = two.getComponent(URI.class);
        Assert.assertNull(uri);

        final Date expectedDate = new Date();
        two.setComponent(Date.class, expectedDate);
        final Date actualDate = one.getComponent(Date.class);
        Assert.assertSame(expectedDate, actualDate);

        Collection<?> collection = one.getComponents();
        System.out.println(collection);
        Assert.assertEquals("Reference 'one' - ComponentRegistry contains one record",
collection.size(), 1);

        collection = two.getComponents();
        Assert.assertEquals("Reference 'two' - ComponentRegistry contains one record",
collection.size(), 1);
    }

    @AfterClass
    public static void closeEjbContainer() {
        ejbContainer.close();
    }
}
```

PropertiesRegistryTest

```
package org.superbiz.registry;

import org.junit.AfterClass;
import org.junit.Assert;
import org.junit.Test;

import javax.ejb.embeddable.EJBContainer;
import javax.naming.Context;

public class PropertiesRegistryTest {

    private final static EJBContainer ejbContainer = EJBContainer.createEJBContainer(
);

    @Test
    public void oneInstancePerMultipleReferences() throws Exception {

        final Context context = ejbContainer.getContext();

        final PropertyRegistry one = (PropertyRegistry) context.lookup(
"java:global/simple-singleton/PropertyRegistry");
        final PropertyRegistry two = (PropertyRegistry) context.lookup(
"java:global/simple-singleton/PropertyRegistry");

        one.setProperty("url", "http://superbiz.org");
        String url = two.getProperty("url");
        Assert.assertSame("http://superbiz.org", url);

        two.removeProperty("url");
        url = one.getProperty("url");
        Assert.assertNull(url);

        two.setProperty("version", "1.0.5");
        String version = one.getProperty("version");
        Assert.assertSame("1.0.5", version);

        one.removeProperty("version");
        version = two.getProperty("version");
        Assert.assertNull(version);
    }

    @AfterClass
    public static void closeEjbContainer() {
        ejbContainer.close();
    }
}
```

Running

Running the example is fairly simple. In the "simple-singleton" directory run:

```
$ mvn clean install
```

Which should create output like the following.

```
-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running org.superbiz.registry.ComponentRegistryTest
INFO -
********************************************************************************
INFO - OpenEJB http://tomee.apache.org/
INFO - Startup: Sun Jun 09 03:46:51 IDT 2013
INFO - Copyright 1999-2013 (C) Apache OpenEJB Project, All Rights Reserved.
INFO - Version: 7.0.0-SNAPSHOT
INFO - Build date: 20130608
INFO - Build time: 04:07
INFO -
********************************************************************************
INFO - openejb.home = C:\Users\Oz\Desktop\ee-examples\simple-singleton
INFO - openejb.base = C:\Users\Oz\Desktop\ee-examples\simple-singleton
INFO - Created new singletonService
org.apache.openejb.cdi.ThreadSingletonServiceImpl@448ad367
INFO - Succeeded in installing singleton service
INFO - Using 'javax.ejb.embeddable.EJBContainer=true'
INFO - Cannot find the configuration file [conf/openejb.xml].  Will attempt to create
one for the beans deployed.
INFO - Configuring Service(id=Default Security Service, type=SecurityService,
provider-id=Default Security Service)
INFO - Configuring Service(id=Default Transaction Manager, type=TransactionManager,
provider-id=Default Transaction Manager)
INFO - Creating TransactionManager(id=Default Transaction Manager)
INFO - Creating SecurityService(id=Default Security Service)
INFO - Found EjbModule in classpath: c:\users\oz\desktop\ee-examples\simple-
singleton\target\classes
INFO - Beginning load: c:\users\oz\desktop\ee-examples\simple-singleton\target\classes
INFO - Configuring enterprise application: C:\Users\Oz\Desktop\ee-examples\simple-
singleton
INFO - Auto-deploying ejb PropertyRegistry: EjbDeployment(deployment-
id=PropertyRegistry)
INFO - Auto-deploying ejb ComponentRegistry: EjbDeployment(deployment-
id=ComponentRegistry)
INFO - Configuring Service(id=Default Singleton Container, type=Container, provider-
id=Default Singleton Container)
INFO - Auto-creating a container for bean PropertyRegistry: Container(type=SINGLETON,
```

```
id=Default Singleton Container)
INFO - Creating Container(id=Default Singleton Container)
INFO - Configuring Service(id=Default Managed Container, type=Container, provider-
id=Default Managed Container)
INFO - Auto-creating a container for bean org.superbiz.registry.ComponentRegistryTest:
Container(type=MANAGED, id=Default Managed Container)
INFO - Creating Container(id=Default Managed Container)
INFO - Using directory C:\Users\Oz\AppData\Local\Temp for stateful session passivation
INFO - Enterprise application "C:\Users\Oz\Desktop\ee-examples\simple-singleton"
loaded.
INFO - Assembling app: C:\Users\Oz\Desktop\ee-examples\simple-singleton
INFO - Jndi(name="java:global/simple-
singleton/PropertyRegistry!org.superbiz.registry.PropertyRegistry")
INFO - Jndi(name="java:global/simple-singleton/PropertyRegistry")
INFO - Jndi(name="java:global/simple-
singleton/ComponentRegistry!org.superbiz.registry.ComponentRegistry")
INFO - Jndi(name="java:global/simple-singleton/ComponentRegistry")
INFO - Existing thread singleton service in SystemInstance():
org.apache.openejb.cdi.ThreadSingletonServiceImpl@448ad367
INFO - OpenWebBeans Container is starting...
INFO - Adding OpenWebBeansPlugin : [CdiPlugin]
INFO - All injection points were validated successfully.
INFO - OpenWebBeans Container has started, it took 68 ms.
INFO - Created Ejb(deployment-id=PropertyRegistry, ejb-name=PropertyRegistry,
container=Default Singleton Container)
INFO - Created Ejb(deployment-id=ComponentRegistry, ejb-name=ComponentRegistry,
container=Default Singleton Container)
INFO - Started Ejb(deployment-id=PropertyRegistry, ejb-name=PropertyRegistry,
container=Default Singleton Container)
INFO - Started Ejb(deployment-id=ComponentRegistry, ejb-name=ComponentRegistry,
container=Default Singleton Container)
INFO - Deployed Application(path=C:\Users\Oz\Desktop\ee-examples\simple-singleton)
[Sun Jun 09 03:46:52 IDT 2013]
INFO - Undeploying app: C:\Users\Oz\Desktop\ee-examples\simple-singleton
INFO - Destroying OpenEJB container
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 1.431 sec
Running org.superbiz.registry.PropertiesRegistryTest
INFO -
********************************************************************************
INFO - OpenEJB http://tomee.apache.org/
INFO - Startup: Sun Jun 09 03:46:52 IDT 2013
INFO - Copyright 1999-2013 (C) Apache OpenEJB Project, All Rights Reserved.
INFO - Version: 7.0.0-SNAPSHOT
INFO - Build date: 20130608
INFO - Build time: 04:07
INFO -
********************************************************************************
INFO - openejb.home = C:\Users\Oz\Desktop\ee-examples\simple-singleton
INFO - openejb.base = C:\Users\Oz\Desktop\ee-examples\simple-singleton
INFO - Created new singletonService
org.apache.openejb.cdi.ThreadSingletonServiceImpl@448ad367
```

```
INFO - Succeeded in installing singleton service
INFO - Using 'javax.ejb.embeddable.EJBContainer=true'
INFO - Cannot find the configuration file [conf/openejb.xml].  Will attempt to create
one for the beans deployed.
INFO - Configuring Service(id=Default Security Service, type=SecurityService,
provider-id=Default Security Service)
INFO - Configuring Service(id=Default Transaction Manager, type=TransactionManager,
provider-id=Default Transaction Manager)
INFO - Creating TransactionManager(id=Default Transaction Manager)
INFO - Creating SecurityService(id=Default Security Service)
INFO - Using
'java.security.auth.login.config=jar:file:/C:/Users/Oz/.m2/repository/org/apache/opene
jb/openejb-core/7.0.0-SNAPSHOT/openejb-core-7.0.0-SNAPSHOT.jar!/login.config'
INFO - Found EjbModule in classpath: c:\users\oz\desktop\ee-examples\simple-
singleton\target\classes
INFO - Beginning load: c:\users\oz\desktop\ee-examples\simple-singleton\target\classes
INFO - Configuring enterprise application: C:\Users\Oz\Desktop\ee-examples\simple-
singleton
INFO - Auto-deploying ejb ComponentRegistry: EjbDeployment(deployment-
id=ComponentRegistry)
INFO - Auto-deploying ejb PropertyRegistry: EjbDeployment(deployment-
id=PropertyRegistry)
INFO - Configuring Service(id=Default Singleton Container, type=Container, provider-
id=Default Singleton Container)
INFO - Auto-creating a container for bean ComponentRegistry: Container(type=SINGLETON,
id=Default Singleton Container)
INFO - Creating Container(id=Default Singleton Container)
INFO - Configuring Service(id=Default Managed Container, type=Container, provider-
id=Default Managed Container)
INFO - Auto-creating a container for bean
org.superbiz.registry.PropertiesRegistryTest: Container(type=MANAGED, id=Default
Managed Container)
INFO - Creating Container(id=Default Managed Container)
INFO - Using directory C:\Users\Oz\AppData\Local\Temp for stateful session passivation
INFO - Enterprise application "C:\Users\Oz\Desktop\ee-examples\simple-singleton"
loaded.
INFO - Assembling app: C:\Users\Oz\Desktop\ee-examples\simple-singleton
INFO - Jndi(name="java:global/simple-
singleton/ComponentRegistry!org.superbiz.registry.ComponentRegistry")
INFO - Jndi(name="java:global/simple-singleton/ComponentRegistry")
INFO - Jndi(name="java:global/simple-
singleton/PropertyRegistry!org.superbiz.registry.PropertyRegistry")
INFO - Jndi(name="java:global/simple-singleton/PropertyRegistry")
INFO - Existing thread singleton service in SystemInstance():
org.apache.openejb.cdi.ThreadSingletonServiceImpl@448ad367
INFO - OpenWebBeans Container is starting...
INFO - Adding OpenWebBeansPlugin : [CdiPlugin]
INFO - All injection points were validated successfully.
INFO - OpenWebBeans Container has started, it took 4 ms.
INFO - Created Ejb(deployment-id=PropertyRegistry, ejb-name=PropertyRegistry,
container=Default Singleton Container)
```

```
INFO - Created Ejb(deployment-id=ComponentRegistry, ejb-name=ComponentRegistry,
container=Default Singleton Container)
INFO - Started Ejb(deployment-id=PropertyRegistry, ejb-name=PropertyRegistry,
container=Default Singleton Container)
INFO - Started Ejb(deployment-id=ComponentRegistry, ejb-name=ComponentRegistry,
container=Default Singleton Container)
INFO - Deployed Application(path=C:\Users\Oz\Desktop\ee-examples\simple-singleton)
INFO - Undeploying app: C:\Users\Oz\Desktop\ee-examples\simple-singleton
INFO - Destroying OpenEJB container
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.171 sec

Results :

Tests run: 2, Failures: 0, Errors: 0, Skipped: 0
```