BeanHelloWorld2

No recommendations | Updated today at 5:05 PM by David A. Wood | Tags: *None*

This tutorial shows how to use the ISharedBeanPlatform to share data between other ISharedBeanPlatform instances running in the same application name space. In this case, the data being shared is a simple string (i.e. hello world), however, any serializable Java objects may be exchanged in this way. We then start multiple instances of the application and they will begin exchanging messages. The steps we will take to build and run this are as follows:

1. Define a class to hold our main() Java method.
2. Create the shared data platform and start it.
3. Add an action method that will listen for changes to remote data in our peer applications.
4. Periodically reshare our hello world message into our local platform.

First we define the basic class and its fields as follows:

```
public class BeanHelloWorld2 implements IBeanAction {

    /** Provides the data sharing and messaging services */
    ISharedBeanPlatform platform = null;

...
```

First, we see that our class implements the IBeanAction interface. We'll come back to this later, but it serves as the mechanism by which we are notified about others' shared data changes. The ISharedBeanPlatform is the object that provides all the messaging and data sharing services we need. Next we define the constructor to simply create the shared bean platform based on a common application name space and instance id for this instantiation of main().

```
    public BeanHelloWorld2(String instanceid) {
        // Create a descriptor defining a common name space, BeanHelloWorld2, and the optional id for this instance.
        IApplicationDescriptor appDesc = new ApplicationDescriptor("BeanHelloWorld2", instanceid);

        // Create the shared bean platform and assign it the given application namespace and id.
        platform = new SharedBeanPlatform(appDesc);
    }
```

There are other constructors available for SharedBeanPlatform that allow finer grain control over the types of services (e.g., IPhysicalTopologyMonitor, IMessageReceiver, etc) used by the instance. Next,w e need to define our main() method.

```
        public static void main(String[] args) throws DSMException {
```

We want allow the instance ID to be defined on the command line if desired so we'll parse a -instance option

```
       CommandArgs cmdargs = new CommandArgs(args);
       String instanceid = cmdargs.getOption("instance");
```

Then we create an instance of our class so we can get access to the shared bean platform contained there.

```
        BeanHelloWorld2 helloWorld= new BeanHelloWorld2(instanceid);
```

Each shared data platform much be started before it is ready to perform its function.

```
        helloWorld.platform.start();
```

Then we set up the platform to notify us when new data is published/shared by any of our peer applications that are also network neigbhors (i.e. within 1 network link of this instance). We can now see why our class had to implement the IBeanAction interface (i.e. so that it defines the action() method called when data is received).

```
        helloWorld.platform.addRemoteDataListener(TopologyRelationship.Neighbor, helloWorld);
```

There are other methods to add such listeners. These allow finer-grain control over the specific data being requested and conditions on those data that must be met before notification is given. Also, this restricts notifications to that from immediate network (0 hop) neighbors. You can use other topological relationshipos here include Reachable and if the IApplicationTopology used by the shared platform supports it, Parent, Child, etc.

Now we go into a loop in which we periodically send data (i.e. Hello World message) to any of our peers that are listening (i.e. all of our neighbors).

```
        for (int i=0 ; i<10 ; i++) {
            // Create the message we want to send.
            String msg = "hello world #" + i + " from " + helloWorld.platform.getApplicationDescriptor().getInstanceID();

            // Share the msg into the shared bean platform, which will cause it to be sent to those listening for it.
            helloWorld.platform.setBeanData("hello", msg);
            try {
                Thread.sleep(5000);
            } catch (InterruptedException e) {
                ;
            }
        }
```

Finally, before we exit, we should stop the platform and any of its services.

```
        helloWorld.platform.stop();
    }
```

Finally, we need to define the IBeanAction.action() method to receive notifications of changes in others' data. We'll define a very simple implementation that shows both the data we've shared and the data we receivved.

```
    public void action(List<ISharedBean> localData, List<IBeanQueryResult> remoteData, Object listenerKey) {

        System.out.println("locaData=" + localData);
        System.out.println("remoteData" + remoteData);

    }
```

Now, we would like to run at least 2 instances of this class. They can run within Eclipse as separate applications. Since this sample is part of the release sample code, you can run these from the command line using the **dsmrun** utility (assuming you have installed the release zip according to the instructions). We'll use the command line option. Open two separate windows and issue the commands shows in the first line below. You shold then see the output shown.

One	Two
C:\dev\dsm>**dsmrun com.ibm.watson.dsm.samples.BeanHelloWorld2 -instance One** Loading dsm properties from file:/c:/dev/dsm/lib/dsm.properties Watson Policy Management Library - Copyright IBM Corporation 2008,2011. Copyrights also by Apache Software Foundation (http://apache.org) and Antlr (http://antlr2.org) Appending properties file to logging configuration: file:/c:/dev/dsm/lib/dsm.properties Jun 18, 2012 12:54:13 PM com.ibm.watson.dsm.platform.messaging.network.NetworkMessageReceiver run INFO: Listening for serialized data on port 3022 **locaData=[hello world #1 from One]** **remoteData[Source: BeanHelloWorld2/Two, results=[hello world #1 from Two]]** **locaData=[hello world #2 from One]** **remoteData[Source: BeanHelloWorld2/Two, results=[hello world #2 from Two]]**	C:\Users\IBM_ADMIN>**dsmrun com.ibm.watson.dsm.samples.BeanHelloWorld2 -instance Two** Loading dsm properties from file:/c:/dev/dsm/lib/dsm.properties Watson Policy Management Library - Copyright IBM Corporation 2008,2011. Copyrights also by Apache Software Foundation (http://apache.org) and Antlr (http://antlr2.org) Appending properties file to logging configuration: file:/c:/dev/dsm/lib/dsm.properties Jun 18, 2012 12:54:15 PM com.ibm.watson.dsm.platform.messaging.network.NetworkMessageReceiver run INFO: Listening for serialized data on port 3023 **locaData=[hello world #0 from Two]** **remoteData[Source: BeanHelloWorld2/One, results=[hello world #0 from One]]** **locaData=[hello world #0 from Two]** **remoteData[Source: BeanHelloWorld2/One, results=[hello world #1 from One]]** **locaData=[hello world #1 from Two]** **remoteData[Source: BeanHelloWorld2/One, results=[hello world #2 from One]]**

Comments

There are no comments.