ORACLE®

Oracle Coherence 3.7.1 POF Annotation Support

Harvey Raja | Software Engineer | Oracle Coherence

Recap

- Prior to 3.7.1, one of the following was required to make classes conform to the Portable Object Format:
  - Implement PortableObject
  - Implement PofSerializer
- Verbose / explicit forms of serialization

PortableObject

```
public class Person implements PortableObject
    {
    public void readExternal(PofReader in) throws IOException
        {
        m_sFirstName = in.readString(FIRST_NAME);
        m_sLastName  = in.readString(LAST_NAME);
        m_nAge       = in.readInt(AGE);
        }

    public void writeExternal(PofWriter out) throws IOException
        {
        out.writeString(FIRST_NAME, m_sFirstName);
        out.writeString(LAST_NAME, m_sLastName);
        out.writeInt(AGE, m_nAge);
        }

    private String m_sFirstName;
    private String m_sLastName;
    private int    m_nAge;

    public static final int FIRST_NAME = 0;
    public static final int LAST_NAME  = 1;
    public static final int AGE        = 2;
    }
```

PofSerializer

```
public class PersonSerializer implements PofSerializer
    {
    public void serialize(PofWriter out, Object o) throws IOException
        {
        Person p = (Person) o;
        out.setVersionId(max(p.getDataVersion(), p.getImplVersion()));

        out.writeString(FIRST_NAME, p.getFirstName());
        out.writeString(LAST_NAME,  p.getLastName());
        out.writeInt(AGE, p.getAge());

        out.writeRemainder(p.getFutureData());
        }

    public Object deserialize(PofReader in) throws IOException
        {
        Person p = new Person();
        p.setDataVersion(in.getVersionId());

        p.setFirstName(in.readString(FIRST_NAME));
        p.setLastName(in.readString(LAST_NAME));
        p.setAge(in.readInt(AGE));

        p.setFutureData(in.readRemainder());
        return p;
        }

    public static final int FIRST_NAME = 0;
    public static final int LAST_NAME  = 1;
    public static final int AGE        = 2;
    }
```

Annotations

- Decorate classes with annotations that drive POF serialization
  - Portable
  - PortableProperty
- Portable is a marker annotation
- PortableProperty allows index and codec to be specified
- A Codec provides the ability to customize the behavior when (de)serilaizing a property

Annotated Person Object - Java

- Annotated Class
- Annotated method
  - Must conform to accessor notation (get/set/is)
- Annotated field

```
@Portable
public class Person
    {
    @PortableProperty(FIRST_NAME)
    public String getFirstName()
        {
        return m_sFirstName;
        }

    private String m_sFirstName;

    @PortableProperty(LAST_NAME)
    private String m_sLastName;

    @PortableProperty(AGE)
    private int    m_nAge;

    public static final int FIRST_NAME = 0;
    public static final int LAST_NAME  = 1;
    public static final int AGE        = 2;
    }
```

Annotated Person Object - .NET

- Annotated Class
- Annotated property
- Annotated method
  - Must conform to accessor notation (get/set/is)
- Annotated field

```
[Portable]
public class Person
{
    [PortableProperty(FIRST_NAME)]
    public string GetFirstName()
    {
        return m_firstName;
    }

    [PortableProperty(LAST_NAME)]
    public string LastName
    {
        get; set;
    }

    private String m_firstName;

    [PortableProperty(AGE)]
    private int m_age;

    public const int FIRST_NAME = 0;
    public const int LAST_NAME  = 1;
    public const int AGE        = 2;
}
```

Annotated Person Object - C++

- Class must be registered with SystemClassLoader COH_REGISTER_CLASS
- Annotated Class
- Annotated method
  - Must conform to accessor notation (get/set/is)

```
class Person
    : public class_spec<Person>
    {
    friend class factory<Person>;

    public:
        String::View getFirstName() const
            {
            return m_vsFirstName;
            }

        void setFirstName(String::View vsFirstName)
            {
            m_vsFirstName = vsFirstName;
            }

    private:
        MemberView<String> m_vsFirstName;
        MemberView<String> m_vsLastName;
        int32_t            m_nAge;

    public:
        static const int32_t FIRST_NAME = 0;
        static const int32_t LAST_NAME  = 1;
        static const int32_t AGE        = 2;
    };
COH_REGISTER_CLASS(TypedClass<Person>::create()
    ->annotate(Portable::create())
    ->declare(COH_PROPERTY(Person, FirstName, String::View)
        ->annotate(PortableProperty::create(Person::FIRST_NAME)))
    ->declare(COH_PROPERTY(Person, LastName, String::View)
        ->annotate(PortableProperty::create(Person::LAST_NAME)))
    ->declare(COH_PROPERTY(Person, Age,  BoxHandle<const Integer32>)
        ->annotate(PortableProperty::create(Person::AGE)))
        );
```

Registration

- Java & .NET
  - Add user-type into pof config

```
<user-type>
    <type-id>1001</type-id>
    <class-name>com.tangosol.io.pof.Person</class-name>
</user-type>
```

- C++
  - Add pre-processor macro

```
COH_REGISTER_POF_ANNOTATED_CLASS(1001,Person);
```

Auto Index

- Automatically generate the index used for each PortableProperty
- Predictable Algorithm
  - Property Name with explicit indices respected
- Currently this feature does not work with Evolvable classes

Auto Index - Registration

- Java & .NET (POF configuration)

```
<user-type>
    <type-id>1001</type-id>
    <class-name>com.tangosol.io.pof.Person</class-name>
    <serializer>
        <class-name>com.tangosol.io.pof.PofAnnotationSerializer</class-name>
        <init-params>
            <init-param>
                <param-type>int</param-type>
                <param-value>{type-id}</param-value>
            </init-param>
            <init-param>
                <param-type>class</param-type>
                <param-value>{class}</param-value>
            </init-param>
            <init-param>
                <param-type>boolean</param-type>
                <param-value>true</param-value>
            </init-param>
        </init-params>
    </serializer>
</user-type>
```

- C++ (pre-processor macro)

```
COH_REGISTER_POF_ANNOTATED_CLASS_AI(1001,Person);
```

Codec

- A Codec defines how to encode and decode a PortableProperty using a PofWriter and PofReader
- The absence of a Codec will instruct the use of DefaultCodec
- DefaultCodec uses:
  - PofWriter.writeObject()
  - PofReader.readObject()

Codec

- Concrete implementation lost in (de)serialization
  - e.g. List implementation could be lost when serialized

- Registration within PortableProperty Annotation:

```
@PortableProperty(codec = LinkedListCodec.class)
private List<String> m_aliases;
```

- Implementation:

```
public static class LinkedListCodec implements Codec
    {
    public Object decode(PofReader in, int index) throws IOException
        {
        return (List<String>) in.readCollection(index, new LinkedList<String>());
        }

    public void encode(PofWriter out, int index, Object value) throws IOException
        {
        out.writeCollection(index, (Collection) value);
        }
    }
```

DEMO

Object Identities and References

- POF Stream now understands an object's identity and references to the same object

- Pros:
  - Reduce the Data Size
  - Support Complex Object Graphs

- Cons:
  - User Defined Objects only

Example

```
public class PortablePerson implements PortableObject
   {
   public PortablePerson()
      {
      }
   public PortablePerson(String sName, Date dtDOB)
      {
      m_sName = sName;
      m_dtDOB = dtDOB;
      }
   ...
   public String          m_sName;
   public Date            m_dtDOB;
   public PortablePerson  m_Spouse;
   public PortablePerson[] m_aChildren;
   }
```

Example (cont.)

Circular References:

PortablePerson Ivan = new PortablePerson("Ivan Smith", new Date(74, 7, 24));

PortablePerson Jane = new PortablePerson("Jane Smith", new Date(78, 3, 12));

Ivan.m_Spouse = Jane;

Jane.m_Spouse = Ivan;

Example (cont.)

Nested Objects:

```
PortablePerson   Joe       = new PortablePerson("Joe Smith",
new Date(2003, 1, 14));

PortablePerson   Ann       = new PortablePerson("Ann Smith",
new Date(2005, 7, 2));

PortablePerson[] children = new PortablePerson[] {Joe, Ann};

Ivan.m_aChildren = children;

Jane.m_aChildren = children;
```

Enable Object References

pof-config.xml, Configuration File

enable-references element:

```
<enable-references>true</enable-references>
```

JAVA and .NET

Enable Object References (cont.)

Programmatically

JAVA:

```
SimplePofContext ctx = new SimplePofContext();
ctx.setReferenceEnabled(true);
```

C++:

```
SystemPofContext::Handle hCtx = SystemPofContext::getInstance();
hCtx->setReferenceEnabled(true);
```

.NET:

```
SimplePofContext ctx = new SimplePofContext();
ctx.IsReferenceEnabled = true;
```

Determine Whether Object References Are Enabled

JAVA:

ConfigurablePofContext.isReferenceEnabled()

SimplePofContext.isReferenceEnabled()

C++:

SimplePofContext ::isReferenceEnabled()

.NET:

ConfigurablePofContext.IsReferenceEnabled

SimplePofContext.IsReferenceEnabled

Register an object with POF

com.tangosol.io.pof.PofReader.registerIdentity(Object o):

```
  public Object deserialize(PofReader pofReader)
      throws IOException
          {
          PortablePerson p = new PortablePerson();
          pofReader.registerIdentity(p);
          p.setName(reader.readString(NAME));
          p.setDate(reader.readDate(DOB));
p.setSpouse((PortablePerson)pofReader.readObject(SPOUSE));
          …
          return p;
          }
```

Limitations and Support

- Object References for Evolvable Object Is Not Supported

- POF Extractors Do Not Support Object References

Questions?