EclipseLink: Bean Validation in JAXB
MOXy grows stronger (again)!

Marcel Valovy
TopLink Software Engineer
April 14, 2015
Program Agenda

- EclipseLink
- History
- MOXy
- Bean Validation in JAXB
- Conclusion
- References
EclipseLink Overview
EclipseLink

Advanced object-persistence and object-transformation project

<table>
<thead>
<tr>
<th>Object-Relational Mapping</th>
<th>Object-XML Binding</th>
<th>Database Web Services</th>
</tr>
</thead>
<tbody>
<tr>
<td>JPA</td>
<td>JAXB, SDO</td>
<td>JAX-WS, JAX-RS</td>
</tr>
</tbody>
</table>

[Diagram showing JPA, MOXy (JAXB), and DBWS]
History of EclipseLink
History ‘90s

1990s The Object People
Consulting company
Research in ORM during 1990s

1994 - Toplink for SmallTalk
History ‘96

1996 TOP team started developments for Java™

1997 TopLink for Java 1.0, written from scratch
History ‘99

1999 “era of J2EE application servers”

Mainstream Adoption:

“TopLink for WebLogic”

“TopLink for WebSphere”
History ‘99

1999 Hot & Award winning product
“*The Hottest and Coolest Product of ‘99*”

Java Pro, Java World and JDJ “Readers’ Choice Award”
History ‘Millenium

April 2000 - TopLink acquired by startup WebGain

TopLink crew grew from 30 to 90 people

One “TopLink” product (merge of “TopLink for: WebLogic, WebSphere, BlueStone/HP-AS”)

History ‘02

**June 2002** - Oracle acquired TopLink with all its 90 people

“Industry leading persistence layer”

O-R mapping extended with O-X mapping
History ‘07

2007 - Oracle contributed TopLink 11g source code to Eclipse Foundation

EclipseLink 1.0 was born!
MOXy - Overview
MOXy Process Flow
MOXy Process Flow - Goal

XML schema -> Binding compiler -> Generate Constraints Metadata!

XML document -> Marshal -> MOXy (JAXB) -> Unmarshal

Validate!

Content objects

Schema-derived classes & interfaces
Bean Validation in JAXB
BV in JAXB - Goals

Automatic **runtime validation** for:

- Objects constructed from unmarshalled documents
- Objects entering marshalling process

**Tooltime metadata generation:**

- Generate BV annotations in XJC compiled Java classes
- Generate XML Facets in SchemaGen generated Schemas
Usage 1 - Unmarshaller

JAXBContext context = JAXBContextFactory.createContext ( … )
Unmarshaller unmarshaller = context.createUnmarshaller ( )

Employee employee = ( Employee ) unmarshaller.unmarshal ( file )
assert unmarshaller.getConstraintViolations ( ).isEmpty ( )

No configuration :-(
Usage 2 - Marshaller, handling constraint violations

JAXBContext context = JAXBContextFactory.createContext ( … )
Marshaller marshaller = context.createMarshaller ( )

try {
    marshaller.marshal ( invalidEmployee , file )
} catch ( BeanValidationException bve ) {
    ... business logic handling invalid objects
}
Usage 3 - XJC: Bean Validation Plugin

Command Line

xjc file.xsd -XBeanVal

Example usage with mods:

xjc file.xsd -XBeanVal jsr303 simpleRegex

Programmatically

Driver.run ( new String [ ] { schemaPath, "-extension", "-XBeanVal" }, System.out, System.out )
Usage 4 - SchemaGen

Map props = new HashMap()
props.put("eclipselink.beanvalidation.facets", true)

JAXBContext jc = JAXBContext.newInstance(classes, properties)

SchemaOutputResolver sor = new MSOR()
jc.generateSchema(sor)
MOXy Process Flow - Enhanced

XML schema ➔ Binding compiler ➔ MOXy (JAXB) ➔ JAXBBeanValidator ➔ Unmarshal

XML document ➔ Marshal ➔ Content objects

XJC BeanValidationPlugin

Schema-derived classes & interfaces
Excellent Performance

**Extensively fine-tuned**

Green = percentage boost achieved

<table>
<thead>
<tr>
<th>Method</th>
<th>thrpt</th>
<th>Rate</th>
<th>Throughput</th>
<th>Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>.testWorkOrderMarshal</td>
<td>1</td>
<td>50</td>
<td>22452.601</td>
<td>76.6456 ops/s</td>
</tr>
<tr>
<td>.testOutputStream</td>
<td>1</td>
<td>50</td>
<td>34695.3601</td>
<td>102.8361 ops/s</td>
</tr>
<tr>
<td>.testOrderResponse</td>
<td>1</td>
<td>50</td>
<td>23666.7424</td>
<td>72.323 ops/s</td>
</tr>
<tr>
<td>.testOutputStream</td>
<td>1</td>
<td>50</td>
<td>39340.0181</td>
<td>128.3776 ops/s</td>
</tr>
</tbody>
</table>
MOXy Runtime Enhancements - Class Diagram

Simple API
Modular Code
Concurrency Support
High Performance
Customizations
Customization

Easy to use, pass **key:value** to method 

`.setProperty()` on either **JAXBContext**, **Marshaller** or **Unmarshaller**

<table>
<thead>
<tr>
<th>BEAN_VALIDATION_MODE</th>
<th>ON, AUTO, OFF</th>
</tr>
</thead>
<tbody>
<tr>
<td>BEAN_VALIDATION_FACTORY</td>
<td>custom</td>
</tr>
<tr>
<td>BEAN_VALIDATION_GROUPS</td>
<td>custom</td>
</tr>
<tr>
<td>GENERATE_FACETS</td>
<td>boolean</td>
</tr>
<tr>
<td>BEAN_VALIDATION_NO_OPTIMISATION</td>
<td>boolean</td>
</tr>
</tbody>
</table>
Conclusion
Conclusion

Bean Validation in JAXB enables developing safe applications with:

- automatic runtime verification of constraints inside the marshalling framework
- automatic generation of Bean Validation metadata
- automatic generation of XML Facets
- translating between XML Facets and Bean Validation
Thanks for Your attention!

Marcel Valovy
TopLink Software Engineer
April 14, 2015
References

