WSDL Fault and Java Exception mapping

Supporting Business Exceptions in Tuscany

Business exception related concerns

1. How to declare business exceptions?

What defines business exception on each interface type. (ie Java wsdl)
In Java interface, business exceptions are declared checked exceptions specified on the operations of the services interface.
Note declared unchecked (runtime) exceptions are not considered business exceptions.
WSDL need to determine if all WSDL defined faults are business exceptions. Need to in investigate JAX-WS mapping of exceptions JAX-B databinding..
SDO has no similar specification. Need to determine a similar means for SDO to provide reliable transforms between JAX-B and SDO.

2. How to represent business exceptions in Java?

In Java business exception are represented as non runtime exceptions. But not all runtime exceptions delivered in a message maybe a Business exception.
If a non runtime exceptions is delivered to a component that is does not declare the exception the exception will be wrappered in a specified runtime exception.
Jax-b need to further investigate the JAX-B specification and follow how it models business exceptions as objects.
SDO there are no specific mappings provided for. May need to wrapper complex parts of an exception as SDO objects in a specific Java exception.

JAX-WS WSDL1.1 to Java mapping for faults

JAX-WS 2.0 spec defines the mapping rule for web service faults in section 2.5.

Using java exception to represent the Web Service fault:

```
/**
 * The java exception to represent the web service fault
 */
public class WebServiceFault extends Exception {
    private static final long serialVersionUID = -3134446726685504039L;
    private Object fault;

    public WebServiceFault(String message, Object fault) {
        super(message);
        this.fault = fault;
    }

    public WebServiceFault(String message, Object fault, Throwable cause) {
        super(message, cause);
        this.fault = fault;
    }

    public Object getFault() {
        return fault;
    }
}
```

3. How to transform business exceptions across databindings?

Provide in the data binding transformations between Axiom OMElement, SDO represented faults and JAX-B exceptions.
How to identify as business exceptions during the transforms.

4. How to propagate business exceptions?

Determine how to propagate exception in the Tuscany runtime message in local interactions. Make sure TargetInvocation exceptions become unwrappered
How in the case of webservice's binding propagate the message through web service binding.

A scenario to test the business exception handling

Integration Tests (iTests)

Several integration tests have been created beginning with prefix exception.

Work items actions

- Either investigate using existing complex type conversions or add additional interfaces to databinding framework for transformation of Exceptions.
- Make SDO generated exceptions match as close as possible to JAX-B. what annotations to SDOs could be added
- Augment Java introspector to capture JAX-B annotations in logical types to help assist in providing hints to determine matching exceptions.
- Use when provided WSDL qname of Exception's message name to match exceptions. If not provided fall back to name matching and possible use of packagename annotations to resolve matching type.
- Attempt to do actual conversion through Axiom transform. If conversion fails try using simple conversion through copying of respective field members.
- Check that only declared checked exceptions are thrown. Wrapper all other checked exceptions.

Questions:

- If the originating exception is a Business exception and conversions fails should we have a **Tuscany standard** runtime exception that will have basic message from the originating exception set ? Should we just pick one of the business exceptions on receiving operation ? This might be more robust than throwing a runtime exception.
- What runtime exception should undeclared, checked exceptions be wrappered in? Tuscany defined ? just RuntimeException ? java.lang.reflect. UndeclaredThrowableException? I can see an SCA client still wanting to be "robust" capturing this and acting on it.
- **Will not directly validate webservices exceptions until axis binding is at incubator-snapshot (kernel trunk) level**
- How are we currently mapping operations during wiring in Tuscany with respect to exceptions ? Need to see if Exceptions are part of operations signature.

Implementation Decisions

- Currently we are unwrappering at the TargetInvokerExtension **all** exceptions and passing them on the message path.
- System exceptions happening during the processing of a message are thrown up the stack.
- We have made some decision on the support of SDO exception wrappers and example of can be seen in the exceptionXbindingTest iTest
  This closely resembles the JAX-B pattern in dealing with faults. One exception is we currently have a FAULT_ELEMENT field type QName on the exception to help tie back to original wsdl element it is associate to.

Appendix

Axis2 WSDL2Java

Exception generated by Axis2 1.1

```
package stockexceptiontestservice.scatesttool;

public class InvalidSymbolFaultException extends java.lang.Exception{

    private stockexceptiontestservice.scatesttool.InvalidSymbolFault faultMessage;

    public InvalidSymbolFaultException() {
        super("InvalidSymbolFaultException");
    }

    public InvalidSymbolFaultException(java.lang.String s) {
       super(s);
    }

    public InvalidSymbolFaultException(java.lang.String s, java.lang.Throwable ex) {
      super(s, ex);
    }

    public void setFaultMessage(stockexceptiontestservice.scatesttool.InvalidSymbolFault msg){
       faultMessage = msg;
    }

    public stockexceptiontestservice.scatesttool.InvalidSymbolFault getFaultMessage(){
       return faultMessage;
    }
}
```

The generated exception has the getter and setter for the fault: setFaultMessage(...) and getFaultMessage() and that's the pattern Axis2 adopts for web service fault to java exception mapping.

The InvalidSymbolFault is a generated class to represent the fault. It can be generated using different databindings such as ADB, JAXB (and hopefully SDO in the future).

JAX-WS RI 2.1 WSDL2Java

Rick provided the WSDL and generated code from JAX-WS RI 2.1.

StockExceptionTest.wsdl

jaxws.wsimport.zip