Jersey Test Framework: TDD for Web Services

Brian Westrich
McWest Corp.
bw@mcwest.com

JavaOne 10/4/2011

Slides:
http://code.mcwest.com/jersey-example/downloads/jtf.pdf

Code:
http://code.mcwest.com/jersey-example

Outline

- Tool survey
- Jersey Test Framework (JTF) primer
- Test Driven Design/Development (TDD) with JTF
- Intermediate Jersey via JTF
- Q&A / Discussion

McWest CORPORATION

Web services: What do you use?

Remote calls: RMI, EJB 2, EJB 3, Web Services, other

Web Services type: SOAP, REST

REST implementation: Restlet, Jersey, Other

Web Services:
What tools do you use?

Transfer format:
XML, JSON

Serialization:
JAXB, XStream, JibX, Burlap, text, other

McWest CORPORATION

What is Jersey?

- Reference implementation for JAX-RS (RESTful Web Services for Java)
- Straightforward API
- Production ready
- Major version numbers mirror JAX-RS versions
  - Version 1.0 released October 13, 2008
  - Current version: 1.9.1 (Sept 16, 2011)
  - 2.0 will be released with JEE7

McWest CORPORATION

What is Jersey Test Framework (JTF)?

Framework for testing Jersey Web Services

Runs as “JUnit” test

Automatically starts/stops “test” app server, deploys your services

Available since Jersey version 1.4

User guide:
http://jersey.java.net/nonav/documentation/latest/test-framework.html

McWest CORPORATION

Types of Web Service testing

Type	Description	Benefits	Drawbacks
Mocked	Call service class methods directly from tests	Fastest	Does not test Jersey service annotations (@Path, @GET, @Accept, etc)
JTF	Use Jersey Test Framework	Faster, Tests Jersey service annotations	Does not fully simulate production service use (e.g. character encoding)
Functional	Call live services using tools such as soapUI	Represen-tative	Slowest, Requires test environment, Tests are laborious to write, Tests are brittle

McWest CORPORATION

Supported “test” app servers

- JTF in-memory: FAST, no full web services stack (e.g. character encoding)
- Standalone app servers: slower, more representative
  - Grizzly
  - JTF HTTP
  - Glassfish
  - External (app server of your choice)

Getting started with JTF

Maven:

```
<dependency>
      <groupId>com.sun.jersey.jersey-test-framework</groupId>
      <artifactId>jersey-test-framework-core</artifactId>
      <version>${jersey.version}</version>
      <type>jar</type>
      <scope>test</scope>
</dependency>
<dependency>
      <groupId>com.sun.jersey.jersey-test-framework</groupId>
      <artifactId>jersey-test-framework-inmemory</artifactId>
      <version>${jersey.version}</version>
      <type>jar</type>
      <scope>test</scope>
</dependency>
```

Core

"Test"
app server

Non-Maven: See Jersey user guide section 7

Sample web service

Time Service

Purpose: return current time

Data format: milliseconds since 1/1/1970

McWest CORPORATION

TDD mantra

Red: write a failing test

Green: make it pass

Clean: refactor

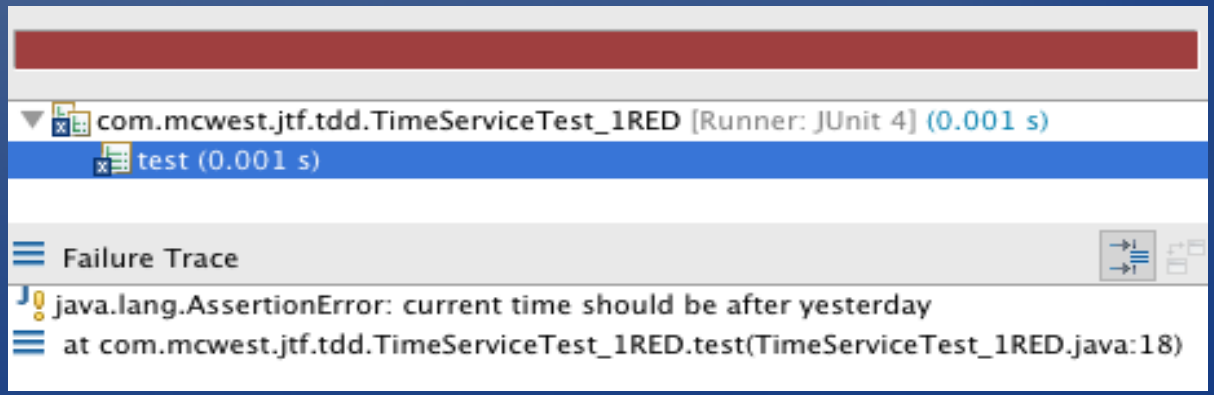
For more on TDD, including cheatsheet and dojo, see
http://code.mcwest.com/java-tdd/wiki/Home

McWest CORPORATION

RED

```
@Test
public void test() {
    Calendar yesterday = GregorianCalendar.getInstance();
    yesterday.add(Calendar.DATE, -1);
    long currentTime = callTimeService();
    assertTrue("current time should be after yesterday",
            new Date(currentTime).after(yesterday.getTime()));
}

private long callTimeService() {
    return 0;
}
```



GREEN

1) Descend from JerseyTest.

```
public class TimeServiceTest_2GREEN extends JerseyTest {
```

2) Write test code that calls the Jersey service.

```
import com.sun.jersey.api.client.Client;
```

```
import com.mcwest.jtf.tdd.Time;
```

```
private long callTimeService() {
    Client client = client();
    Time time = client.resource("/currentTime_2GREEN").get(Time.class);
    return time.getTime();
}
```

Notes:

- client() returns an automatically configured client.
- Test does not compile since “Time” DTO does not exist.
- Jersey will automatically call JAXB to unmarshal XML into object.

GREEN (cont.)

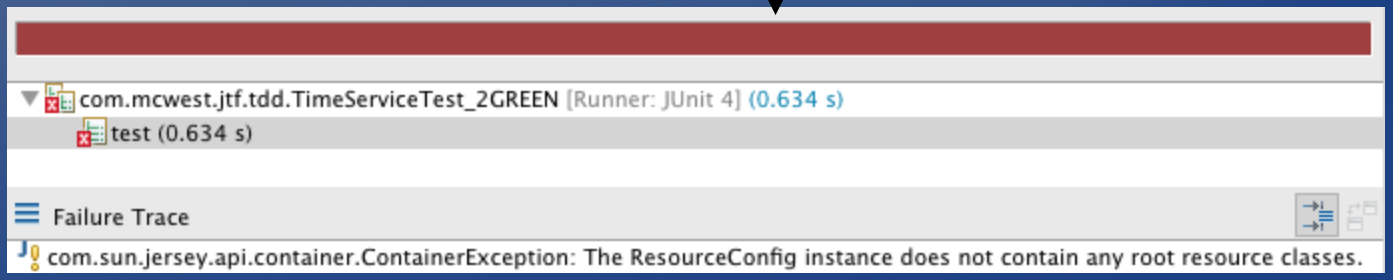
3) Write XSD that generates dto.

dto package

dto name

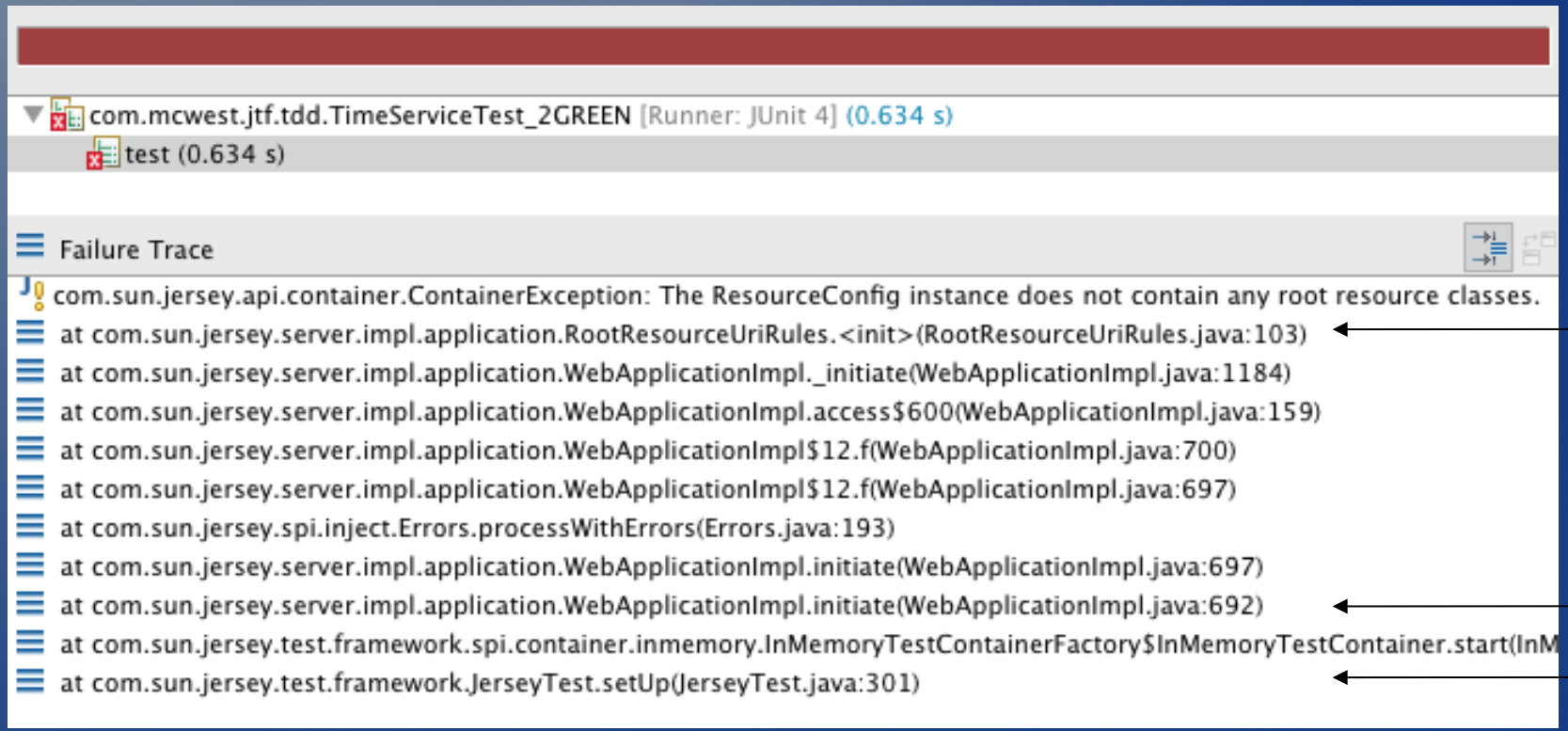
```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
    targetNamespace="urn:www.mcwest.com/jtf/tdd"
    elementFormDefault="qualified" >
    <element name="Time">
        <complexType>
            <attribute name="time" use="required" type="long" />
        </complexType>
    </element>
</schema>
```

Code now compiles, but test fails (fast!) as service does not exist.



GREEN (cont.)

Stack trace shows how JTF works...



GREEN (cont.)

4) Write service class

Jersey automatically calls JAXB to marshal object into XML.

```
package com.mcwest.jtf.tdd;

import javax.ws.rs.GET;
import javax.ws.rs.Path;

@Path("/currentTime_2GREEN")
public class TimeService_2GREEN {

    @GET
    public Time get() {
        Time time = new Time();
        time.setTime(System.currentTimeMillis());
        return time;
    }
}
```

GREEN (cont.)

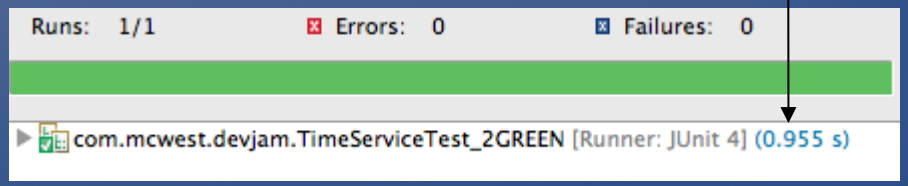
5) Declare location of service class.

```
public TimeServiceTest_2GREEN() throws Exception {
    super(TimeService_2GREEN.class.getPackage().getName());
}
```

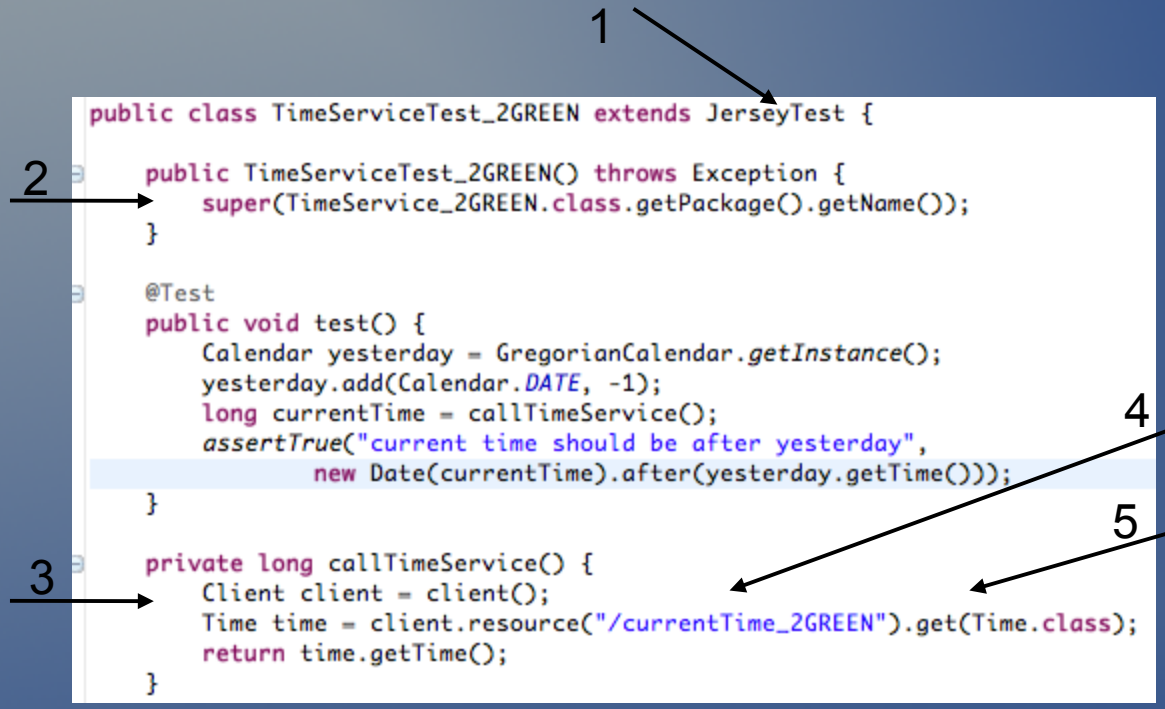
Registers all classes in package that have @Path annotation

GREEN (cont.)

6) Test passes (in less than a second!).



GREEN (review)



1. inherit from JerseyTest
2. register server resources
3. get client
4. specify path
5. select verb

```
@Path("/currentTime_2GREEN")
public class TimeService_2GREEN {

    @GET
    public Time get() {
        Time time = new Time();
        time.setTime(System.currentTimeMillis());
        return time;
    }
}
```

- Note: Automatic JAXB marshalling/unmarshalling

CLEAN

Example refactor: Extract path to a constant

```
@Path(TimeService_3CLEAN.PATH)
public class TimeService_3CLEAN {

    public static final String PATH = "/currentTime_3CLEAN";
```

server

```
    Time time = client.resource(TimeService_3CLEAN.PATH).get(Time.class);
```

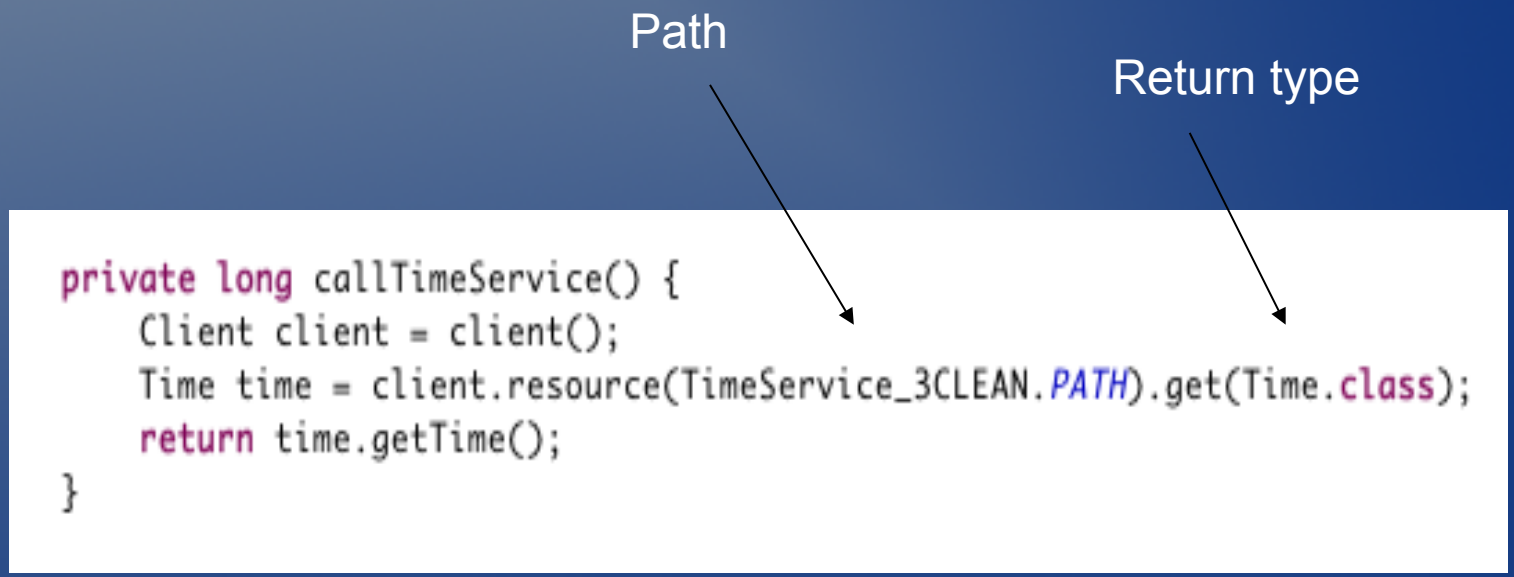
client

This example refactor...

- keeps test in sync with service
- helps users locate service (Eclipse F3)
- assumes caller in same classloader as service (e.g. thick client jar, service test code).

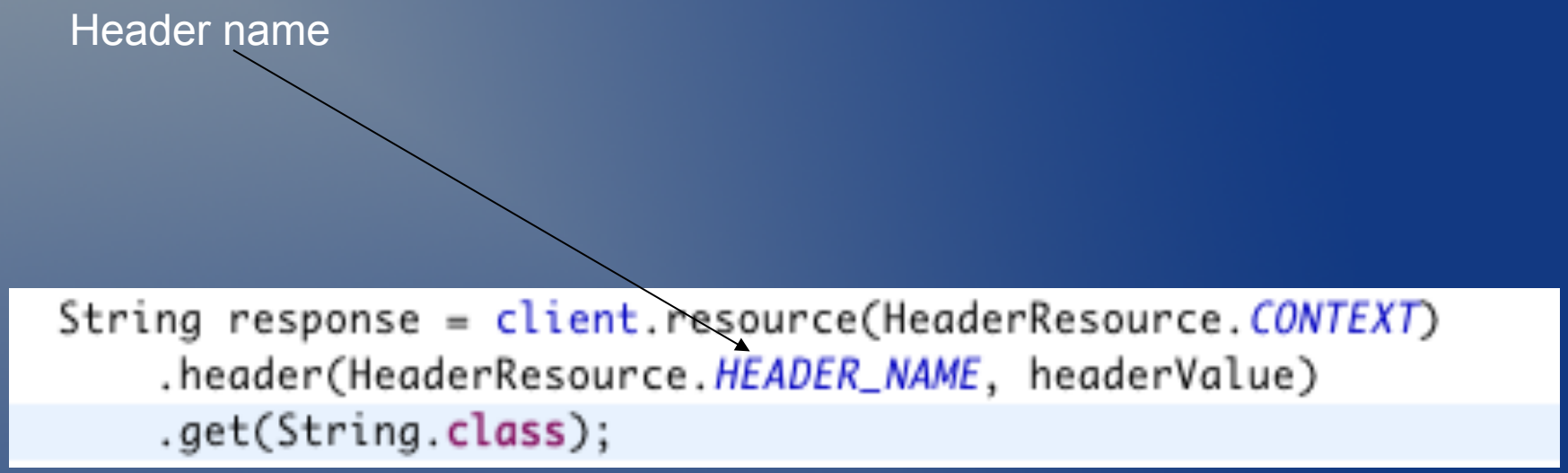
JTF tests as sample clients (executable documentation)

Example: get without parameters



JTF tests as sample clients

Example: get with header



Providing a sample client is more important as service API complexity increases

Other JTF testable Web Service features*

- Parameters (path, query, post/put form or dto)
- Verbs (get, post, put, delete)
- Content types (xml, html, pdf, ...)
- Filters (client)
- Exception mapping
- Serialization methods (JAXB, JibX, Castor, ...)
- For sample code, see code.mcwest.com/jersey-example
- For names of classes of sample code, see slide notes

Serialization

- A key element of web services
- Jersey integrated with JAXB serialization, can use other serialization methods

XStream: Simplified object serialization

```java
public class XStreamSimpleExampleTest {

    @Test
    public void simple() {
        Calendar now = GregorianCalendar.getInstance();
        XStream xstream = new XStream();
        String xml = xstream.toXML(now);
        System.out.println(xml);
        Calendar timeFromXml = (Calendar) xstream.fromXML(xml);
        assertThat(timeFromXml, is(now));
    }
```

Convert object to xml

Convert xml back to object

```xml
<gregorian-calendar>
  <time>1313872539855</time>
  <timezone>America/Chicago</timezone>
</gregorian-calendar>
```

XStream specific xml

Serialization research using JTF

- Try to use of XStream instead of JAXB
  - Will XStream simplify Jersey WS serialization?

XStream: client deserialization

```
public class TimeServiceTest_XStream extends JerseyTest {

    public TimeServiceTest_XStream() throws Exception {
        super(TimeService_XStream.class.getPackage().getName());
    }

    @Test
    public void test() {
        Calendar yesterday = GregorianCalendar.getInstance();
        yesterday.add(Calendar.DATE, -1);
        long currentTime = callTimeService();
        assertTrue("current time should be after yesterday",
                new Date(currentTime).after(yesterday.getTime()));
    }

    private long callTimeService() {
        Client client = client();
        String timeAsXml = client.resource(TimeService_XStream.PATH).get(String.class);
        XTime xtime = unmarshallXml(timeAsXml);
        return xtime.getTime();
    }

    private XTime unmarshallXml(String xml) {
        XStream xstream = TimeServiceXStreamUtils.getXStreamForTime();
        XTime xtime = (XTime) xstream.fromXML(xml);
        return xtime;
    }
```

dto

Service returns XML

Must unmarshall XML manually

XStream: dto

- Manually coded XStream data transfer object (dto) (JAXB generates dto from XSD)
  - Xstream leads to added dto maintenance effort

```
public class XTime {

    public XTime(Long time) {
        super();
        this.time = time;
    }

    private long time;

    public long getTime() {
        return time;
    }

    public void setTime(long time) {
        this.time = time;
    }

    @Override
    public int hashCode() {
        final int prime = 31;
        int result = 1;
        result = prime * result + (int) (time ^ (time >>> 32));
        return result;
    }

    @Override
    public boolean equals(Object obj) {
        if (this == obj)
            return true;
        if (obj == null)
            return false;
        if (getClass() != obj.getClass())
            return false;
        XTime other = (XTime) obj;
        if (time != other.time)
            return false;
        return true;
    }

    @Override
    public String toString() {
        return "XTime [time=" + time + "]";
    }
```

Xstream: namespace

- Manual name space configuration
  - Needed for each return type

```
public class TimeServiceXStreamUtils {

    public static XStream getXStreamForTime() {
        QNameMap qnameMap = new QNameMap();
        QName qname = new QName("urn:www.mcwest.com/jtf/tdd", "Time");
        qnameMap.registerMapping(qname, XTime.class);
        XStream xstream = new XStream(new StaxDriver(qnameMap));
        return xstream;
    }
```

  - JAXB uses namespace declared in the XSD
  - XStream leads to duplicate maintenance of namespace info

XStream: server side serialization

```java
@Path(TimeService_XStream.PATH)
public class TimeService_XStream {

    public static final String PATH = "/currentTime_XStream";

    @GET
    public String get() {
        XTime xtime = new XTime(System.currentTimeMillis());
        String xml = marshallObject(xtime);
        return xml;
    }

    private String marshallObject(XTime xtime) {
        XStream xStream = TimeServiceXStreamUtils.getXStreamForTime();
        return xStream.toXML(xtime);
    }
}
```

Must manually marshall object into XML

Manual coding:

JAXB

```java
public class TimeServiceTest_3CLEAN extends JerseyTest {

    public TimeServiceTest_3CLEAN() throws Exception {
        super(TimeService_3CLEAN.class.getPackage().getName());
    }

    @Test
    public void test() {
        Calendar yesterday = GregorianCalendar.getInstance();
        yesterday.add(Calendar.DATE, -1);
        long currentTime = callTimeService();
        assertTrue("current time should be after yesterday",
                new Date(currentTime).after(yesterday.getTime()));
    }

    private long callTimeService() {
        Client client = client();
        Time time = client.resource(TimeService_3CLEAN.PATH).get(Time.class);
        return time.getTime();
    }

}
```

```java
@Path(TimeService_3CLEAN.PATH)
public class TimeService_3CLEAN {

    public static final String PATH = "/currentTime_3CLEAN";

    @GET
    public Time get() {
        Time time = new Time();
        time.setTime(System.currentTimeMillis());
        return time;
    }
}
```

XStream

```java
public class TimeServiceTest_XStream extends JerseyTest {

    public TimeServiceTest_XStream() throws Exception {
        super(TimeService_XStream.class.getPackage().getName());
    }

    @Test
    public void test() {
        Calendar yesterday = GregorianCalendar.getInstance();
        yesterday.add(Calendar.DATE, -1);
        long currentTime = callTimeService();
        assertTrue("current time should be after yesterday",
                new Date(currentTime).after(yesterday.getTime()));
    }

    private long callTimeService() {
        Client client = client();
        String timeAsXml = client.resource(TimeService_XStream.PATH).get(String.class);
        XTime xtime = unmarshallXml(timeAsXml);
        return xtime.getTime();
    }

    private XTime unmarshallXml(String xml) {
        XStream xstream = TimeServiceXStreamUtils.getXStreamForTime();
        XTime xtime = (XTime) xstream.fromXML(xml);
        return xtime;
    }
}
```

```java
@Path(TimeService_XStream.PATH)
public class TimeService_XStream {

    public static final String PATH = "/currentTime_XStream";

    @GET
    public String get() {
        XTime xtime = new XTime(System.currentTimeMillis());
        String xml = marshallObject(xtime);
        return xml;
    }

    private String marshallObject(XTime xtime) {
        XStream xStream = TimeServiceXStreamUtils.getXStreamForTime();
        return xStream.toXML(xtime);
    }
}
```

```java
public class TimeServiceXStreamUtils {

    public static XStream getXStreamForTime() {
        QNameMap qnameMap = new QNameMap();
        QName qname = new QName("urn:www.mcwest.com/jtf/tdd", "Time");
        qnameMap.registerMapping(qname, XTime.class);
        XStream xstream = new XStream(new StaxDriver(qnameMap));
        return xstream;
    }

}
```

```java
public class XTime {

    public XTime(Long time) {
        super();
        this.time = time;
    }

    private long time;

    public long getTime() {
        return time;
    }

    public void setTime(long time) {
        this.time = time;
    }

    @Override
    public int hashCode() {
        final int prime = 31;
        int result = 1;
        result = prime * result + (int) (time ^ (time >>> 32));
        return result;
    }

    @Override
    public boolean equals(Object obj) {
        if (this == obj)
            return true;
        if (obj == null)
            return false;
        if (getClass() != obj.getClass())
            return false;
        XTime other = (XTime) obj;
        if (time != other.time)
            return false;
        return true;
    }

    @Override
    public String toString() {
        return "XTime [time=" + time + "]";
    }
}
```

For Jersey WS,
XStream requires much more manual coding than JAXB

Serialization research using JTF

- Intended simplification would have added end-to-end complexity
- JTF eases Web Service architectural decisions

Optional Slides: Start

Nested XSDs

- A more serious concern of using Xstream in web services

Nested XSDs

- Reusable XSD (file = jtf-tdd-nested-namespaced-time.xsd)

```
<schema xmlns="http://www.w3.org/2001/XMLSchema"
    targetNamespace="urn:www.mcwest.com/jtf/tdd/time/reusable"
    xmlns:time="urn:www.mcwest.com/jtf/tdd/time/reusable"
    elementFormDefault="qualified">
    <complexType name="ReusableTime">
        <sequence>
            <element name="timeInMillis" type="long" />
            <element name="preferredTimeZone" type="string" />
        </sequence>
    </complexType>
</schema>
```

Nested XSDs

- Example reuse in another XSD. Generated Java object will contain nested XSD namespace in package import statement.

```
<schema xmlns="http://www.w3.org/2001/XMLSchema"
    targetNamespace="urn:www.mcwest.com/jtf/tdd"
    elementFormDefault="unqualified"
    xmlns:time="urn:www.mcwest.com/jtf/tdd/time/reusable">
    <import namespace="urn:www.mcwest.com/jtf/tdd/time/reusable"
        schemaLocation="jtf-tdd-nested-namespaced-time.xsd" />
    <element name="CalendarAppointment">
        <complexType>
            <sequence>
                <element name="description" type="string" />
                <element name="startTime" type="time:ReusableTime" />
                <element name="endTime" type="time:ReusableTime" />
            </sequence>
        </complexType>
    </element>
</schema>
```

Xstream limitation re: nested XSDs

- No support for namespaces on nested XSDs
- Must use chameleon XSDs

Chameleon XSD

- Example (file = jtf-tdd-nested-chameleon-time.xsd)
  - Does not declare a namespace

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="qualified">
    <xsd:complexType name="ChameleonTime">
        <xsd:sequence>
            <xsd:element name="timeInMillis" type="xsd:long" />
            <xsd:element name="preferredTimeZone" type="xsd:string" />
        </xsd:sequence>
    </xsd:complexType>
</xsd:schema>
```

Chameleon XSD reuse

- “include” (like a macro or ‘C’ include)
- Chameleon assumes namespace of file it is included into

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    targetNamespace="urn:www.mcwest.com/jtf/tdd"
    xmlns="urn:www.mcwest.com/jtf/tdd"
    elementFormDefault="qualified">
    <xsd:include schemaLocation="jtf-tdd-nested-chameleon-time.xsd" />
    <xsd:element name="CalendarAppointmentUsingChameleons">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="description" type="xsd:string" />
                <xsd:element name="startTime" type="ChameleonTime" />
                <xsd:element name="endTime" type="ChameleonTime" />
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>
```

Chameleon XSDs: Problems

- Type name cannot be qualified by namespace
  - Think “Java class in default package”
  - Leads to unwieldly names
    - Item -> InputBatchItem_v3_2
- Generated Java class has no record of origin (package) of chameleon code
- Name collision potential
  - Two chameleon XSDs with same name can’t be imported into same “parent” XSDs

Chameleon XSDs: Problems

“Ultimately, namespaces, like packages, are there for a reason, which is to locally group a set of types and set them off as being constituent of a given general idea. If you can't put your types in a namespace, you may need to reconsider your design. If you could put your types in a namespace, but choose not to in the hopes of gaining flexibility, expect interoperability and collision problems sooner or later.” Eben Hewitt (2009) Java SOA Cookbook (O'Reilly and Associates)

XStream: no nested namespaces

- SpringSource's position on XStream for WS
  - *Note that XStream is an XML serialization library, not a data binding library. Therefore, it has limited namespace support. As such, it is rather unsuitable for usage within Web services.* – http://http://static.springsource.org/spring-ws/site/reference/html/oxm.html
- Problems easily demonstrated using JTF
- JTF can help demonstrate rationale for architectural decisions

Advanced server resource init

Initialize server resources with **classes**:
when don't want to load all @Resource classes in a package for a specific test
(e.g. avoid declaration of duplicate contexts)

Initialize server resources with **objects**:
create objects and initialize yourself (e.g. inject mocks)
(e.g. use a prepopulated Spring bean as a service resource)

For working code,
see ResourceRegistrar.class in sample code repository
(jersey-example/src/test/java/com/mcwest/jersey/example/resource/ResourceRegistrar.java)

McWest CORPORATION

JTF wish list

- Spring servlet capabilities (e.g. filter registration)
- Easier setup of custom message body readers/ writers (e.g. for specialized content types)

Optional Slides: End

Summary

Jersey Test Framework enables...

- TDD of web services: RED – GREEN – CLEAN
- Executable documentation for Web services
- Quick learning of Jersey features
- Sound architectural choices

McWest CORPORATION

Download Slides and Code

Slides

Link: http://code.mcwest.com/jersey-example/downloads/jtf.pdf

Code (Mercurial)

Host: http://code.mcwest.com

Repository: jersey-example

McWest CORPORATION

Jersey Test Framework: TDD for Web Services

Brian Westrich
McWest Corp.
bw@mcwest.com

JavaOne 2011

Slides:
http://code.mcwest.com/jersey-example/downloads/jtf.pdf

Code:
http://code.mcwest.com/jersey-example