Agenda

• Apache CXF’s WebSocket Support
• What is SwaggerSocket
• Using with Apache CXF and Apache Olingo
• Demos
Apache CXF’s WebSocket Support
Advantages of WebSocket (1)

**non-duplex channel**
(normal HTTP)

Data is transferred over multiple socket connections or a single connection (keep-alive) in either direction at time

**duplex channel**
(websocket)

Data can be transferred over a single socket connection in both directions
A WebSocket connection can be established across the network boundaries as long as an HTTP request can be sent in one direction*.

* Unless explicitly blocked or not supported
Apache CXF is an open source services framework supporting jaxws and jaxrs frontends, various WS-* standards and security features, and transports.

http://cxf.apache.org
**WebSocket transport added in CXF 3.0.0**

- enabled JAXWS and JAXRS services to be invoked over a WebSocket
- used Jetty or Atmosphere (with a supported Servlet container) and runs in the standalone or the servlet-container mode
- used one specific protocol binding (CXF’s default WebSocket protocol)
In Jetty-Embedded Mode, use the ws[s] instead of http[s] in the address attribute

```xml
<jaxrs:server id="bookservice" address="ws://localhost:8080/services/rest">
    <jaxrs:serviceBeans>
        <ref bean="bookServiceBean" />
    </jaxrs:serviceBeans>
...
</jaxrs:server>
```

In Servlet-Container-Mode, set the transportId

```xml
<jaxrs:server id="bookservice" address="/services/rest" transportId="http://cxf.apache.org/transports/websocket">
    <jaxrs:serviceBeans>
        <ref bean="bookServiceBean" />
    </jaxrs:serviceBeans>
...
</jaxrs:server>
```
Once a socket is opened, a request message can be sent over the socket. Each request and response message looks like an HTTP message.

Request = Method SP Request-URI CRLF *(( header ) CRLF) CRLF [ body ]

Response = [Status-Code CRLF ] *(( header ) CRLF) CRLF [ body ]

A GET request with optional requestId for message correlation

GET /services/rest/getBook/184... requestId: 77a5114a-3b78-4581...

A POST request

POST /services/rest/addBook
Content-Type: text/xml

A successful response

200
Content-Type: text/json
...

An error response

405
responseId: 77a5114a-3b78-3b78...
New WebSockets Features in CXF 3.0.5

- WebSocket protocol binding is implemented as an Atmosphere interceptor

- Allowing the protocol binding to be easily switched or extended by replacing or adding the Atmosphere interceptors

- Jetty Embedded mode can now use Atmosphere to take advantage of Atmosphere’s features and this protocol binding support

- Jetty-Only mode is supported with no new features
What is Apache Olingo
Apache Olingo Overview

- Apache Olingo is an open source project implementing the OData (Open Data Protocol) standard, a data query and manipulation protocol based on REST principle

  - [http://olingo.apache.org](http://olingo.apache.org)
GET /OData.svc/Category(1)/Products?$top=2&orderby=name

GET /OData.svc/Category(1)/Products(2)

GET /OData.svc/Category(1)/Products(2)/Price

POST /OData.svc/Category(2)/Products(3) HTTP/1.1
Content-Type: application/json
...

PUT /OData.svc/Category(2)/Products(3)/Accessary
Content-Type: application/json
...

OData REST messages
What is SwaggerSocket
What is SwaggerSocket

- REST service calls over WebSocket
  - A series of service invocations can be performed on a single bidirectional, duplex channel.
  - Asynchronous handling directly supported
  - Similar to CXF’s WebSocket protocol binding but uses JSON to package the messages

- Originally introduced by Wordnik in 2012 using Atmosphere Framework
  - Apache Licensed Open Source Project hosted at https://github.com/swagger-api/swagger-socket
  - Atmosphere Framework is a scalable asynchronous application framework supporting various transports such as WebSocket, Server-Side-Events, …
• SwaggerSocket 2.0.1

• Server
  – Atmosphere based, implemented as an Atmosphere protocol interceptor
  – OSGi enabled (available as Karaf-Feature)

• Client
  – Javascript: works in Browsers or Node.js (installable from npm)
  – Scala
// using jQuery variant
var ss = new jQuery.swaggersocket.SwaggerSocketListener();
var swaggerSocket = new jQuery.swaggersocket.SwaggerSocket();

// listeners methods called at open, close, error, response,..
ss.onOpen = function(response) {};
ss.onClose = function(Response) {};
ss.onError = function(Response) {};
ss.onResponse = function(Response) {};
ss.onResponses = function (Response) {};

// opening a connection
var request = new jQuery.swaggersocket.Request()
    .path(path)
    .listener(ss);
swaggerSocket.open(request);
SwaggerSocket Protocol (1)

After WebSocket is open, a handshake request is sent:

```json
{"handshake":{"protocolVersion":"1.0",
   "protocolName":"SwaggerSocket",
   "uuid":"0",
   "path":"ws://localhost:8080/swaggersocket"}
}
```

A handshake response with the identity key:

```json
{"status":{"statusCode":200,"reasonPhrase":"OK"},
   "identity":"a5b9363c-ba21-4916-8ca8-b61e66529cbd"}
```
// sending a request
var request = new jQuery.swaggersocket.Request()
  .path("path")
  .method("GET")
  .listener(ss);
swaggerSocket.send(request);

// sending multiple requests at once
var requests = new Array();
requests[0] = new jQuery.swaggersocket.Request()
  .path("path1")
  .method("POST")
  .data("Hello World")
  .dataFormat("text/plain")
  .listener(ss);
requests[1] = new jQuery.swaggersocket.Request()
  .path("/path2")
  .method("GET")
  .listener(ss);
swaggerSocket.send(requests);
a request message can be sent as

```json
{"identity":"a5b9363c-ba21-4916-8ca8-b61e66529cbd",
 "requests":[{"uuid":"5e4dbf1f-2117-f024-3d59-a1e71060d13e",
 "method":"POST",
 "path":"/echo",
 "dataFormat":"text/plain",
 "messageBody":"Hello World"}]
}
```

a response message can be received as

```json
{"identity":"a5b9363c-ba21-4916-8ca8-b61e66529cbd",
 "responses":[{"headers":[{"name":"Content-Type","..."}],
 "path":"/echo",
 "uuid":"5e4dbf1f-2117-f024-3d59-a1e71060d13e",
 "messageBody":"Hello World",
 "last":true,
 "reasonPhrase":"OK","statusCode":200}]}
```
additional request attributes can be supplied as required by the application

```json
{"identity":"a5b9363c-ba21-4916-8ca8-b61e66529cbd",
"requests":[{"uuid":"5e4dbf1f-2117-f024-3d59-a1e71060d13e",
"method":"POST",
"path": "/echo",
"dataFormat": "application/json",
"headers": [{"name":"name1","value":"value1"},
            {"name":"name2","value":"value2"}, ...],
"queryStrings": [{"name":"name1","value":"value1"},
                {"name":"name2","value":"value2"}, ...],
"messageBody": "..."}
}
```
Enabling SwaggerSocket for REST services

- **Option 1**
  - When publishing JAXRS resources, simply use SwaggerSocketServlet to host the services

- **Option 2**
  - For existing servlet based applications, SwaggerSocketServlet can be added to process the original requests and route internally over the application’s own servlet

- **Option 3**
  - When using CXF, which is Atmosphere-enabled, simply register the SwaggerSocket protocol interceptor in its Atmosphere transport’s interceptors list.
Demos
- CXF with SwaggerSocket on Karaf
- Olingo with SwaggerSocket
- **SwaggerSocket Project**
  - [https://github.com/swagger-api/swagger-socket](https://github.com/swagger-api/swagger-socket)
  - [https://github.com/swagger-api/swagger-socket/tree/master/samples](https://github.com/swagger-api/swagger-socket/tree/master/samples)
  - Googlegroup: swagger-swaggersocket
- **Other SwaggerSocket samples (Demo) at**
  - [https://github.com/elakito/swaggersocket-samples](https://github.com/elakito/swaggersocket-samples)
- **Atmosphere Framework**
  - [https://github.com/Atmosphere/atmosphere](https://github.com/Atmosphere/atmosphere)
- **Apache CXF**
  - [http://cxf.apache.org](http://cxf.apache.org)
- **Apache Olingo**
  - [http://olingo.apache.org](http://olingo.apache.org)

- **Contact**
  Akitoshi Yoshida
  ay@apache.org
  @elakitoyo