

EML
Research

The SABIO-RK Web Services

Saqib Mir
SABIO-RK User Meeting

- Introduction

- The What, the How the Why

- Brief Overview of the SABIO-RK Web service

- Use Cases

- Some examples of collaborations

- Demo and Hands-On

- Writing a simple client in Java

- Final Comments

- What?

- Convert your application to a Web application
- Reusable components
- Application-application (platform independent) communication

- How?

- XML
 - SOAP
 - WSDL

- Simple Object Access Protocol*
- XML-based protocol to exchange structured information between applications
- Uses HTTP for message transfer

A SOAP Message



POST /InStock HTTP/1.1 Host: www.example.org Content-Type: application/soap+xml; charset=utf-8 Content-Length: nnn

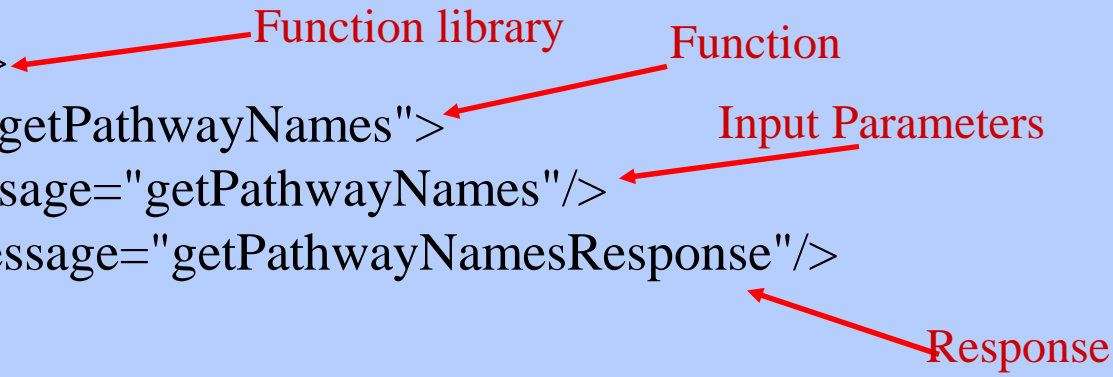
```
<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
<soap:Header> ... .. </soap:Header>
<soap:Body> ... ..
    <soap:Fault> ... .. </soap:Fault>
</soap:Body>
</soap:Envelope>
```

All major programming languages provide APIs!

```

<message name="getPathwayNames">
  <part name="parameters" type="tns:getPathwayNames"/>
</message>
<message name="getPathwayNamesResponse">
  <part name="parameters" type="tns:getPathwayNamesResponse"/>
</message>
<portType name="sabiork">
  <operation name="getPathwayNames">
    <input message="getPathwayNames"/>
    <output message="getPathwayNamesResponse"/>
  </operation>
</portType>
...
<xs:complexType name="getPathwayNames"><xs:sequence>
<xs:element name="arg0" type="xs:string" minOccurs="0"/>
</xs:sequence></xs:complexType>
<xs:complexType name="getPathwayNamesResponse"><xs:sequence>
<xs:element name="return" type="xs:string" nillable="true" minOccurs="0"
maxOccurs="unbounded"/>
</xs:sequence></xs:complexType>

```



WSDL
generation
tools!

- Enables Application-Application communication
 - Language, OS independent
- Bypasses firewalls
- Reusability and Customization
- Easy support for Legacy Code

- 54 Functions

- Fine granulation
- Filter, select and export data in SBML
- Almost complete coverage of SABIO-RK
- Methods that provide entry points with:
 - Pathways, enzymes (name and EC), compounds, ChEBI ID, KEGG ID (compound and reaction), PubMed ID
- Methods that export controlled vocabularies
 - getAllUniProtIDs
 - getAllCompoundIDs
 - getAllReactionIDs
 - getAllEnzymes
 - getAllPathways
 - getAllUnits

- JAX-WS

- Document/Literal Wrapped Style

- **Modellers**

- SysMO (EU)
- HepatoSys (Germany)
- Kinetic Modelling of Hepatocyte Metabolism

- **Modelling and Simulation Platforms**

- CellDesigner (Japan)
- SBMM-Assistant (Spain)
 - Kinetic modeling of metabolic pathways
- BioUML (Russia)
 - Modelling, simulation and Visualization tool for bio-pathways
- VirtualCell (US)
 - Model biological processes in the cell

- **Database**

- ChEBI (UK)
 - Mapping Reactions and Kinetic Data

- You don't need to be a programmer to access a Web service

1. Download the AXIS library

<http://ws.apache.org/axis/>

Axis 1.4

2. Unpack to a folder **MyClient**

3. Open command

3. cd into

MyClient/axis-1_4/lib

4. Go to SABIO-RK WSDL in your browser

5. WSDL-2-Java tool in AXIS

```
>java -cp axis.jar;.....;saaj.jar;  
org.apache.axis.wsdl.WSDL2Java  
http://sabio.bioquant.uni-  
heidelberg.de/sabiork?wsdl  
-Nhttp://localhost:8084/SpringingSabioRk/  
sabiork=org.eml.sdbv.sabioclient -W
```

6. Create JAR for generated code

```
>jar -cvf myJar.jar ./org/*
```

7. Create new java project

8. Add axis and myJar

9. Code!

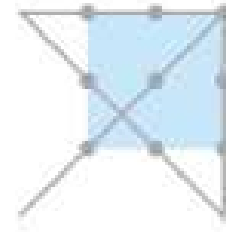
- The API is user-driven

- Don't hesitate

- Help us

- Debug

- Stress-test



EML
Research

Thank you!

- Questions?
- Requests?
- Issues?

$$[Ga]' = k_1 - k_2[Ca_{cyt}][Ga] + (k_2[Ca_{cyt}]) - k_3 \frac{[Ga][PLC]}{([Ga] + K_4)} - k_5 \frac{[Ga][Ca_{cyt}]}{([Ga] + K_5)}$$

$$[PLC]' = k_7[Ca_{cyt}] - k_8 \frac{[PLC]}{([PLC] + K_9)}$$

$$[Ca_{cyt}]' = (Ca_{ER} - Ca_{cyt}) * \frac{k_{10} * Ca_{cyt} * PLC^4}{PLC^4 + K_{11}^4} + k_{12} * PLC + k_{13} * [Ga]$$

$$- k_{14} \frac{[Ca_{cyt}]}{([Ca_{cyt}] + K_{15})} - k_{16} \frac{[Ca_{cyt}]}{([Ca_{cyt}] + K_{17})} - k_{18} \frac{[Ca_{cyt}]^n}{([Ca_{cyt}]^n + K_{19}^n)}$$

$$+ (Ca_{mit} - Ca_{cyt}) * k_{20} \frac{[Ca_{cyt}]}{([Ca_{cyt}] + K_{21})}$$

$$[Ca_{ER}]' = - (Ca_{ER} - Ca_{cyt}) * \frac{k_{10} * Ca_{cyt} * PLC^4}{PLC^4 + K_{11}^4} + k_{16} \frac{[Ca_{cyt}]}{([Ca_{cyt}] + K_{17})}$$

$$[Ca_{Mito}]' = k_{18} \frac{[Ca_{cyt}]^n}{([Ca_{cyt}]^n + K_{19}^n)} - (Ca_{mit} - Ca_{cyt}) * k_{20} \frac{[Ca_{cyt}]}{([Ca_{cyt}] + K_{21})}$$

- Reference:

- <http://www.ibm.com/developerworks/webservices/library/ws-whichwsdl/>

- RPC/encoded

- +

- WSDL is simple
 - Easy dispatch because of operation name

- -

- Type encoding is overhead
 - SOAP request message cannot be easily validated
 - Not WS-I compliant

```
<soap:envelope>
```

```
<soap:body>
```

```
<myMethod>
```

```
<x xsi:type="xsd:int">5</x>
```

```
<y xsi:type="xsd:float">5.0</y>
```

```
</myMethod>
```

```
</soap:body>
```

```
</soap:envelope>
```

- RPC/literal

- +

- WSDL is simple

- Easy dispatch because of operation name

- Type encoding is removed

- WS-I compliant

- -

- SOAP request message cannot be easily validated

```
<soap:envelope>
```

```
<soap:body>
```

```
<myMethod> <x>5</x> <y>5.0</y>
```

```
</myMethod> </soap:body> </soap:envelope>
```

- Document/literal

– +

- Type encoding is removed
- SOAP request message can be easily validated

– –

- Not completely WS-I compliant (soap:body can only have one child)
- WSDL slightly more complex
- Dispatch is difficult

<soap:envelope>

<soap:body>

<xElement>5</xElement>

<yElement>5.0</yElement>

</soap:body>

</soap:envelope>

- Document/literal wrapped

– +

- Type encoding is removed
- SOAP request message can be easily validated
- Dispatch is easy, as you have the method name in the message
- Completely WS-I compliant

– -

- WSDL slightly more complex (But it is not supposed to be human-readable anyway!)

<soap:envelope>

<soap:body>

<myMethod>

<x>5</x> <y>5.0</y>

</myMethod>

</soap:body>

</soap:envelope>

- Defines message format & protocol

```
<binding type="sabiork" name="AnyName">
  <soap:binding style="document"
    transport="http://schemas.xmlsoap.org/soap/http" />
  <operation>
    <soap:operation
      soapAction="getPathwayNames"/>
    <input>
      <soap:body use="literal"/>
    </input>
    <output>
      <soap:body use="literal"/>
    </output>
  </operation>
</binding>
```

- Slow processing
- Client-Server architecture

$$\begin{aligned}
 [Ca_{cyt}]' &= (Ca_{ER} - Ca_{cyt}) * \frac{[PLC]^4}{[PLC]^4 + K_{11}^4} + k_{12} * PLC + k_{13} * [Ga] \\
 &\quad - k_{14} \frac{[Ca_{cyt}]}{([Ca_{cyt}] + K_{15})} - k_{16} \frac{[Ca_{cyt}]}{([Ca_{cyt}] + K_{17})} - k_{18} \frac{[Ca_{cyt}]^n}{([Ca_{cyt}]^n + K_{19}^n)} \\
 &\quad + (Ca_{mit} - Ca_{cyt}) * k_{20} \frac{[Ca_{cyt}]}{([Ca_{cyt}] + K_{21})} \\
 [Ca_{ER}]' &= - (Ca_{ER} - Ca_{cyt}) * \frac{k_{10} * Ca_{cyt} * PLC^4}{[PLC]^4 + K_{11}^4} + k_{16} \frac{[Ca_{cyt}]}{([Ca_{cyt}] + K_{17})} \\
 [Ca_{Mito}]' &= k_{18} \frac{[Ca_{cyt}]^n}{([Ca_{cyt}]^n + K_{19}^n)} - (Ca_{mit} - Ca_{cyt}) * k_{20} \frac{[Ca_{cyt}]}{([Ca_{cyt}] + K_{21})}
 \end{aligned}$$