

Publishing Databases with Web Services

ORION SOA





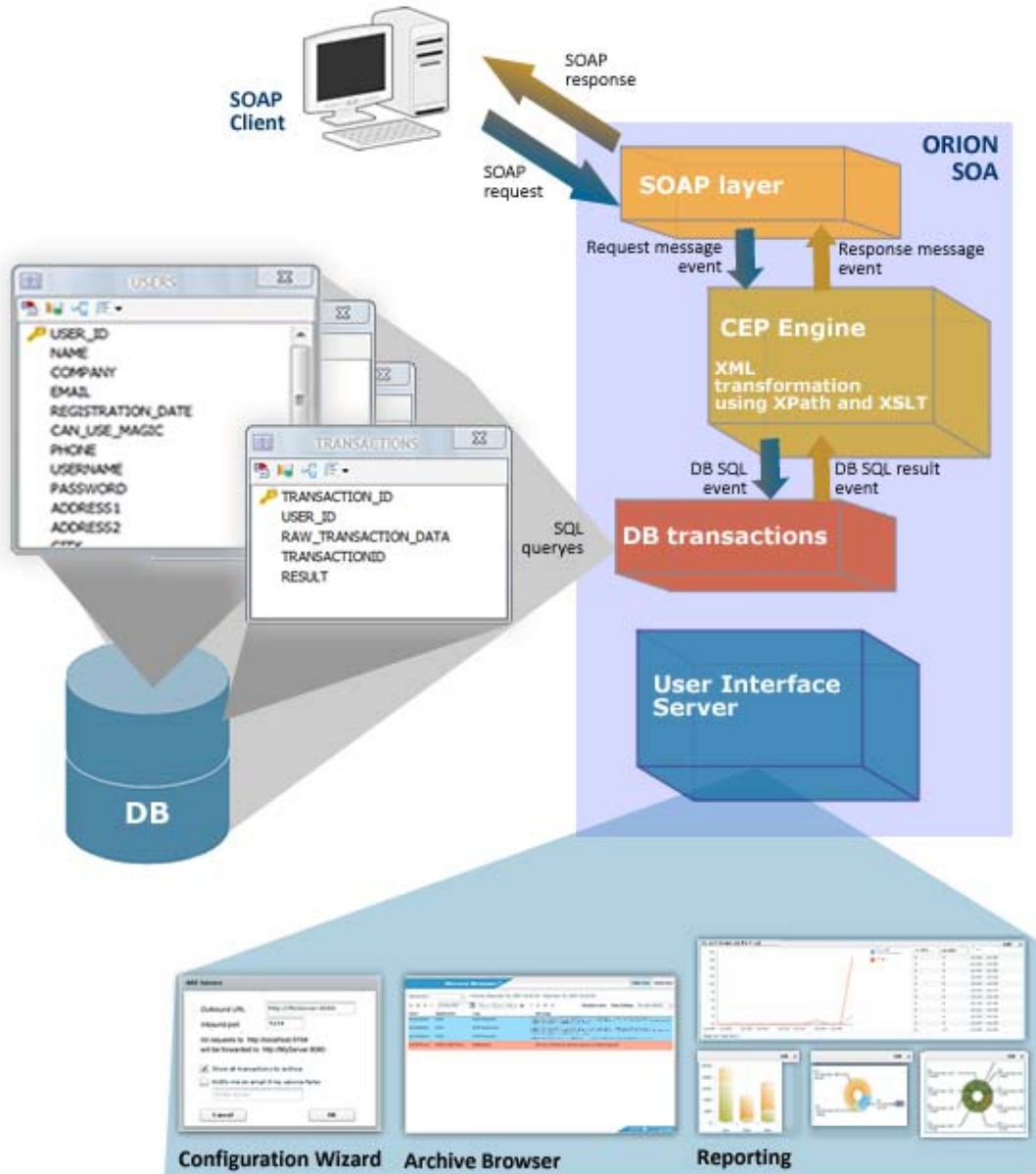
Abstract

Web services technology enables application-to-application interaction over the web - regardless of the platform, language, or data formats. Web services technology usually refers to services implemented and deployed in middle-tier application servers. However, in heterogeneous and disconnected environments, there is an increasing need to access stored procedures as well as data and metadata, through web services interfaces, as well.

In this paper, we explore the architecture and implementation of providing web services access to databases through the use of ORION SOA as an event driven rule-based application server. This approach is able to transform, analyze, and report on web services requests and responses and database queries and results in real-time. It is not only a very fast and non-programmatic way to offer web services but it also provides operational performance management and a detailed logging of individual transactions for diagnosis and to achieve transaction non-repudiation.



Publishing Databases using ORION SOA





SOAP Transaction Layer

The SOAP Transaction Layer of ORION SOA receives web services transaction requests over HTTP and transforms them into SOAP request event messages. The HTTP connections are internally held in suspense until a response message is available to be delivered as a SOAP response back to the requester or until a user-defined time-out is reached. This allows for processing of a high volume of simultaneous asynchronous SOAP transactions through the system. Multiple ORION SOA servers can be clustered to increase the overall transaction throughput.

Complex Event Processing (CEP) Engine

At the heart of ORION SOA is a complex event processing and event correlation engine. The CEP engine is able to process a wide range of events in real-time arriving from numerous sources, including Enterprise Services Buses (ESBs) via Java Message Service (JMS), log files, and TCP/UDP connections. The CEP engine is not only able to analyze and correlate data across these different event streams, but is also able to orchestrate complex requests across multiple application and database servers. In addition, the CEP engine can initiate real-time actions across the network, such as restarting unresponsive database servers or adding additional application servers to a computing grid. Any processing problems can cause real-time notifications to be sent via email, Syslog, or SNMP. Transactions can be logged at any level of detail.

Database Transactions

In this example, the CEP engine initiates a database SQL query based on data contained in the original SOAP request, such as the name of a user for whom specific information needs to be retrieved. The database result is appended to the original event and can therefore be processed or transformed in other processing steps (called “filters”) in the CEP engine. In our example, we transform the SQL query result set into an HTML table using an XSLT conversion filter. Once all processing on the SOAP request event is completed it is returned to the SOAP Transaction Layer where the event is matched with its original suspended SOAP request and is returned as a SOAP response to complete the transaction. This process is usually very fast and completes in milliseconds where most of the response delay (latency) is dependent on the speed of the underlying database. ORION SOA can interface with any standard JDBC capable database, such as Oracle, SQL Server, DB2, MySQL, etc.

User Interface Server

The user interface server provides configuration, monitoring, and real-time dashboard and reporting interfaces to the administrator, business end-users, and the operations staff. All user interfaces are web-based. Most configurations takes place in rule based application wizards. This allows for very rapid development of web services applications without low-level programming, even though full Java API integration is available for advanced developers.



ORION SOA Platform

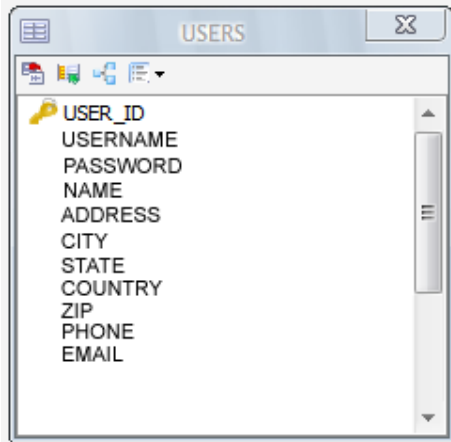
Some of the benefits of using the ORION SOA Platform are:

- Database to Web Service wizard for quick database publishing
- In-line or out-of-band processing modes
- Highly scalable with clustering support
- Very quick deployment time
- Alerting with email, SMS, Syslog, or SNMP
- Multiple transaction audit archives in the Archive Browser
- Reporting module for real-time dashboard or printed reports
- Based on the ORION SOA 7.0 rule-based application server platform
- Rapid development without coding using a wizard driven rich web interface
- Professional services deployment support is available by our team of experts



Application Example – “UsersWebService”

The example shown here is for a simple database of users. We show how this USERS database table can be exposed as a web service.



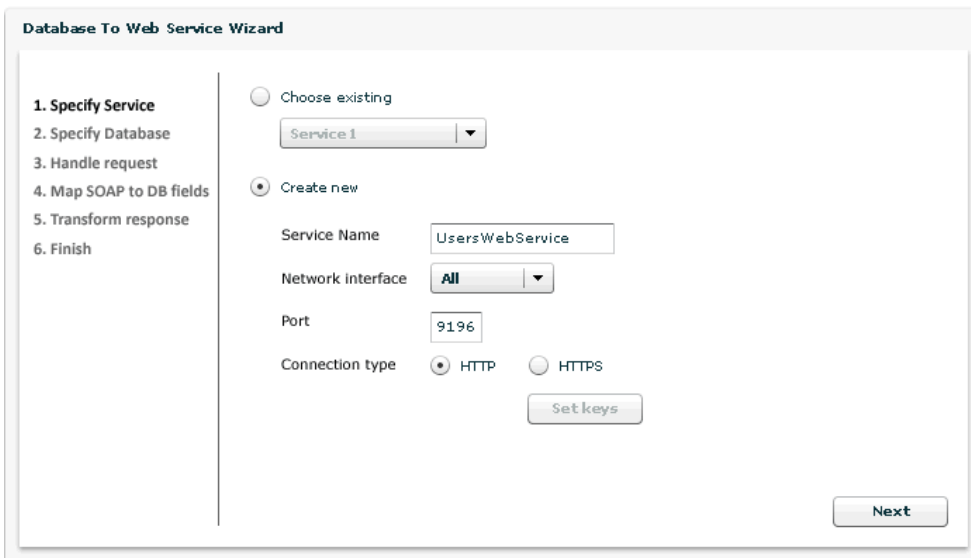
The web service is called “UsersWebService” and can be accessed in this application through “http://127.0.0.1/UsersWebService:9196”. The example shows a SOAP request called “getUserInfo”, which returns well formatted user information for the provided username:

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
  <SOAP-ENV:Body>
    <getUserInfo>
      <username>janedoe</username>
    </ getUserInfo >
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```



The ORION SOA platform with its “Data Base to Web Services Wizard” enables you to quickly publish any database with Java Database Connectivity (JDBC) support (most major databases provide JDBC support). It enables you to quickly map database fields to appropriate fields in the SOAP request/response using predefined templates. No low level coding is required.

When we start the wizard, we first need to specify the service. We can choose from a list of existing ones or create new one. In this example, we create a new service by the name of “UsersWebService” which uses HTTP on port 9196:



Database To Web Service Wizard

1. Specify Service
2. Specify Database
3. Handle request
4. Map SOAP to DB fields
5. Transform response
6. Finish

Choose existing
Service 1

Create new

Service Name: UsersWebService

Network interface: All

Port: 9196

Connection type: HTTP HTTPS

Set keys

Next



Next, we also need to specify the database connection to be used. We give it a name, specify the JDBC URL, locate the JDBC driver, and provide the database login information.

Database To Web Service Wizard

- Specify Service
- Specify Database**
- Handle request
- Map SOAP to DB fields
- Transform response
- Finish

Choose existing database connection
 Create new database connection

My Con

Connection Name: MyConnection2
 URL: jdbc:mysql://localhost/USERS
 Driver: org.gjt.mm.mysql.Driver
 Username: root
 Password: *****

Previous Next

The next step is to specify a configuration for our SOAP request “getUserInfo”. That can be done either by manually adding fields for the request, or by pasting an example request and extracting the list of fields automatically.

Database To Web Service Wizard

- Specify Service
- Specify Database
- Handle request**
- Map SOAP to DB fields
- Transform response
- Finish

Requests List

Add New

Add New Using Example

Enter SOAP request example

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
  <SOAP-ENV:Body>
    <getUserInfo>
      <username>janedoe</username>
    </getUserInfo>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

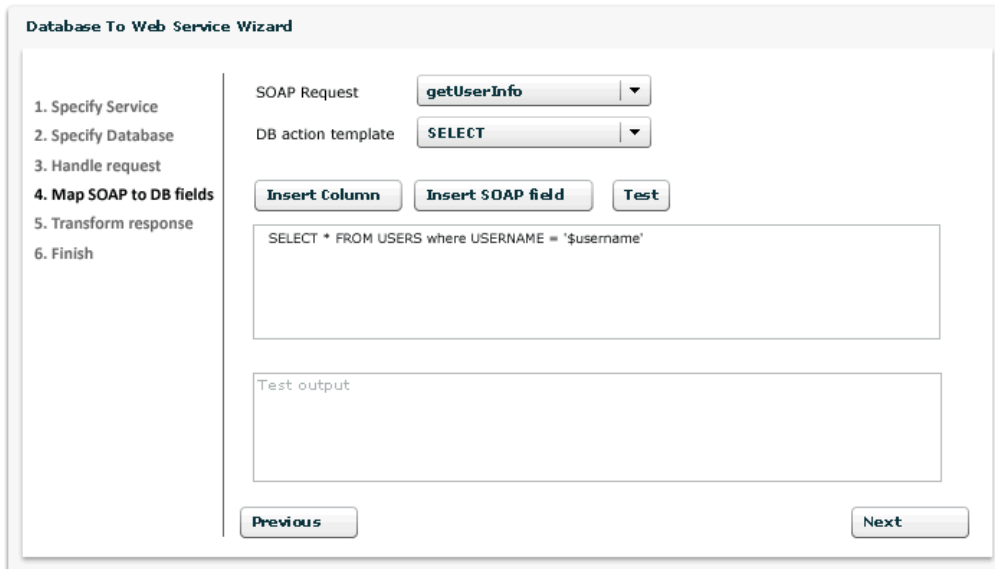
Extract Fields

Field Name	XPath Command	Example Value
username	//getUserInfo/username	janedoe

Previous Next



Next, we will specify the database SQL query that needs to be executed on each SOAP request. The value of the “\$username” field that we previously extracted from the SOAP request is now be used as part of the SQL query.



Database To Web Service Wizard

1. Specify Service
2. Specify Database
3. Handle request
4. Map SOAP to DB fields
5. Transform response
6. Finish

SOAP Request: **getUserInfo**

DB action template: **SELECT**

Insert Column **Insert SOAP field** **Test**

SELECT * FROM USERS where USERNAME = '\$username'

Test output

Previous **Next**

Templates are available for most of the common actions: select, select with foreign key, insert, insert with foreign key, update and delete.

The record set created from the SQL query above is automatically provided in a common XML format:

```
<Recordset requestID="0234823479913323" time="01022008000122.342" >
  <Row ID="77">
    <USERNAME>janedoe</USERNAME>
    <NAME>Jane Doe</NAME>
    <ADDRESS>Address 1</ ADDRESS >
    <CITY>Some City</ CITY >
    <STATE>CA</ STATE >
    <COUNTRY>USA</ COUNTRY >
    <ZIP>231231</ ZIP >
    <PHONE>+1(707)111-111-209</ PHONE >
    <EMAIL>janedoe@yahoo.com</ EMAIL >
  </Row>
</Recordset>
```



The XML result can be sent back directly in the SOAP response as it is, or it can be transformed using XSLT. In this example we will transform the output to an HTML page with a table. XSLT is a standard XML language which allows you to transform one type of XML into another XML format.

Database To Web Service Wizard

1. Specify Service
2. Specify Database
3. Handle request
4. Map SOAP to DB fields
- 5. Transform response**
6. Finish

SOAP Response getUserInfo ▾

Enter example DB resultset xml Get response for test request

```

<ResultSet reqID="0234823479913323" time="01022008000122.342">
  <Row ID="77">
    <USERNAME>janedoe</USERNAME>
    <NAME>Jane Doe</NAME>
    <ADDRESS>Address 1</ADDRESS>
          
```

Enter XSLT Get from XSLT file

```

<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="html" encoding="UTF-8" indent="yes"/>
<xsl:template match="//Row">
<html>
  <body>
    <table border="1">
      <thead>
        <tr>
          <th>USERNAME</th>
          
```

```

<html>
  <body>
    <table border="1">
      <thead>
        <tr>
          <th>USERNAME</th>
          
```

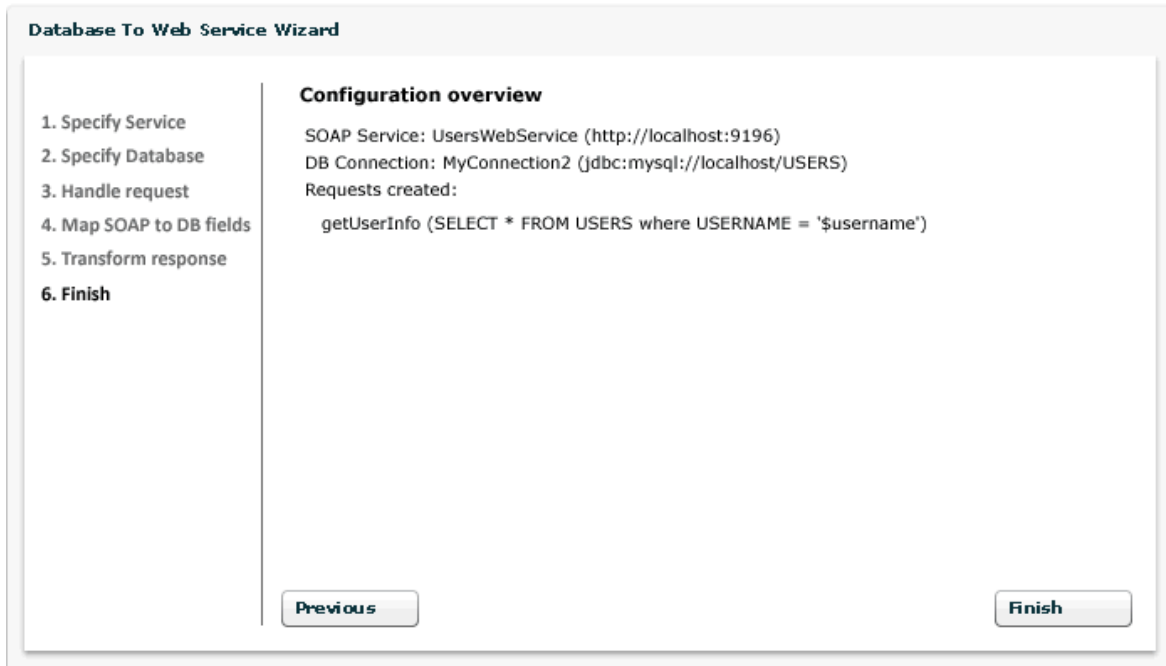
Previous
Next

The full XSLT and the complete SOAP response containing the HTML for the formatted table are shown in the Appendix. However, below you can see the resulting HTML page/table that shows the database data requested in the original SOAP request:

USERNAME	NAME	ADDRESS	CITY	STATE	COUNTRY	ZIP	PHONE	EMAIL
janedoe	Jane Doe	Address 1	Some City	CA	USA	231231	+1(707)111-111-209	janedoe@yahoo.com



The final screen in the configuration wizard provides an overview of the services and queries configured.





Application Example – Alerting, Archival, and Reporting

ORION SOA has extensive real-time monitoring, alerting, and dashboard reporting capabilities. The following example shows how to configure a real-time alert when there are more than 3 transaction failures in 5 minutes.

Add Action ✕

Action Type: **Create Sum Alert** ▼

For each unique value of field: **Custom** ▼

sum the value in field: **Custom** ▼

Create new event if counter reaches: within **Min** ▼

New event fields **Add** **Remove**

Message ▼	<input type="text" value="ev:msg"/>	<input type="text" value="Multiple transaction failure"/>	..
Priority ▼	<input type="text" value="ev:priority"/>	<input type="text" value="critical"/>	..

Creates new alert event

Unique counter and timer instances are generated for each unique value of the first Field Name. The Time Interval starts when the first event arrives. If the Time Interval expires before the Threshold is reached,

Back **Ok**

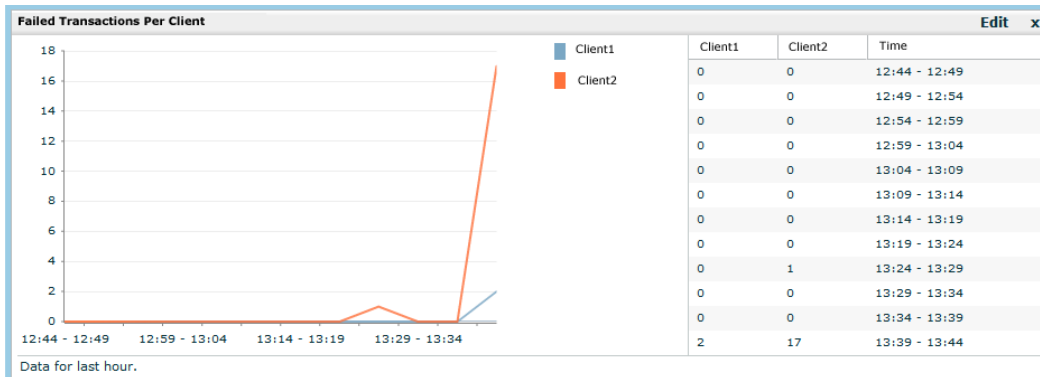
The alert event created has a “critical” priority which means it will be highlighted red in the transactions/event archive. We can also send it out as a notification by email, SMS or Syslog.

Archive Browser™ Table view Detail view			
Host	Application	Log	Message
myClientHost	SOAP	SOAP Requests	<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"><SOAP-ENV:Header/><SOAP-ENV:Body><TransferOutORA count><senderAccount>1</senderAccount></SOAP-ENV:Body></SOAP-ENV:Envelope>
myClientHost	SOAP	SOAP Responses	<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"><SOAP-ENV:Header/><SOAP-ENV:Body><TransferOutORA countResponse>OK</TransferOutORA countResponse></SOAP-ENV:Body></SOAP-ENV:Envelope>
myClientHost	SOAP	SOAP Requests	<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"><SOAP-ENV:Header/><SOAP-ENV:Body><TransferOutORA countResponse>OK</TransferOutORA countResponse></SOAP-ENV:Body></SOAP-ENV:Envelope>
mySOAPproxy	ORION SOAP Proxy	Notifications	Multiple transaction failures for Client2



Using other action types we can create charting data which can be used for displaying application performance metrics in a real-time dashboard or in a printed report.

Here is one example for chart displaying failed transactions per client:





Conclusion

With the high performance ORION SOA platform you are able to bring advanced web services applications to live quickly. It eliminates the need for low-level coding of web services applications, allows for rapid application development, and makes production management and monitoring of web services application easier. Overall ORION SOA reduces the total cost of ownership of your web services efforts.

Please visit <http://www.EventGnosis.com/SOA> if you like to know more, or to request a WebEx product demonstration.



APPENDIX

The following is the complete SOAP response which includes the database result set formatted as a web page with a table as it was created from the final XSLT conversion step.

SOAP Response:

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
  <SOAP-ENV:Body>
    <getUserInfoResponse>
      <html>
        <body>
          <table border="1">
            <thead>
              <tr>
                <th>USERNAME</th>
                <th>NAME</th>
                <th>ADDRESS</th>
                <th>CITY</th>
                <th>STATE</th>
                <th>COUNTRY</th>
                <th>ZIP</th>
                <th>PHONE</th>
                <th>EMAIL</th>
              </tr>
            </thead>
            <tr align="left">
              <td>janedoe</td>
              <td>Jane Doe</td>
              <td>Address 1</td>
              <td>Some City</td>
              <td>CA</td>
              <td>USA</td>
              <td>231231</td>
              <td>+1(707)111-111-209</td>
              <td>janedoe@yahoo.com</td>
            </tr>
          </table>
        </body>
      </html>
    </getUserInfoResponse >
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```



Here is the XSLT used in example.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="html" encoding="UTF-8" indent="yes"/>
<xsl:template match="//Row">
<html>
<body>
<table border="1">
<thead>
<tr>
<xsl:for-each select="*">
<th><xsl:value-of select="name()"/></th>
</xsl:for-each>
</tr>
</thead>
<tr align="left">
<xsl:for-each select="*">
<td>
<xsl:value-of select="./text()"/>
</td>
</xsl:for-each>
</tr>
</table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```