Author : Monica Devino

Date: 05/29/2018

Version: 1.8

Assignment” Please Diagram a system, application or project that you worked on. Describe what the end-product did and provide technical details on how it was built, such as frameworks, references, components and assemblies, programmatic languages, design patterns, server technology,. Please provide details at any level or scope , from off the shelf solutions, to method signatures.”

================================================================================================

Goal: Business purchased a 3rd party software system called RegOnline (website: [www.regonline.com](http://www.regonline.com))

The purpose of the product was to allow professors to setup Online classes, using a point and click interface. The product allows a non-technical end user to build, promote, host and analyze your events and classes. Allows Online class registration, track agendas, sessions, class content, and student attendance. In addition the software allows Payment collection for class and credit card fees.

Problem: 3rd Party software was brought online with no delivered method to get financial information into our ERP financial System “PeopleSoft”. The software vendor provided an API to access its financial information. The goal was to write an interface from this system to our Peoplesoft Financial System and also create a program that would summarize this data and create Journal entries monthly

High Level Overview - The data inputs from vendor API – and the big moving processes into Peoplesoft.

The numbers in this diagram are noted in the document throughout with details about each object.

Blue objects are external (Regonline) Orange Objects are internal (Peoplesoft ERP System)

**TECHNICAL DETAILS**

**Data Input :** ><https://www.regonline.com/api/>

The Regonline software vendor exposes the following operations detailed on it’s website here

By clicking on an operation you can see the SOAP contents behind the operation.

We utilized three operations to form our journal entry data.

1. REGONLINE API - GetEvent
2. REGONLINE API - GetRegistration
3. REGONLINE API - GetRegistrationsByEventID

Details of these Operations -

1. GETEVENT The following is a sample SOAP 1.1 request and response. The **placeholders** shown needed to be replaced with actual values. This is the REQUEST - I had to pass an encrypted APIToken (we had an api token per college, and I had a page that stored all the colleges and their Associated API token

POST /api/default.asmx HTTP/1.1

Host: www.regonline.com

Content-Type: text/xml; charset=utf-8

Content-Length: **length**

SOAPAction: "http://www.regonline.com/api/GetEvent"

<?xml version="1.0" encoding="utf-8"?>

<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">

<soap:Header>

<TokenHeader xmlns="http://www.regonline.com/api">

<APIToken>**string**</APIToken>

</TokenHeader>

</soap:Header>

<soap:Body>

<GetEvent xmlns="http://www.regonline.com/api">

<eventID>**int**</eventID>

</GetEvent>

</soap:Body>

</soap:Envelope>

**GETEVENT - THIS IS THE RESPONSE FROM REGONLINE**

It would return a list of all the events that were going on for a given college.

HTTP/1.1 200 OK

Content-Type: text/xml; charset=utf-8

Content-Length: **length**

<?xml version="1.0" encoding="utf-8"?>

<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">

<soap:Body>

<GetEventResponse xmlns="http://www.regonline.com/api">

<GetEventResult>

<Success>**boolean**</Success>

<Message>**string**</Message>

<Data>

<APIEvent>

<ID>**int**</ID>

<CustomerID>**int**</CustomerID>

<ParentID>**int**</ParentID>

<Status>**string**</Status>

<Title>**string**</Title>

<StartDate>**dateTime**</StartDate>

<EndDate>**dateTime**</EndDate>

<ActiveDate>**dateTime**</ActiveDate>

<ClientEventID>**string**</ClientEventID>

<TypeID>**int**</TypeID>

<Type>**string**</Type>

<City>**string**</City>

<State>**string**</State>

<Country>**string**</Country>

<CountryCode>**string**</CountryCode>

<PostalCode>**string**</PostalCode>

<LocationName>**string**</LocationName>

<LocationRoom>**string**</LocationRoom>

<LocationPhone>**string**</LocationPhone>

<LocationBuilding>**string**</LocationBuilding>

<LocationAddress1>**string**</LocationAddress1>

<LocationAddress2>**string**</LocationAddress2>

<TimeZone>**string**</TimeZone>

<Capacity>**int**</Capacity>

<CurrencyCode>**string**</CurrencyCode>

<Keywords>**string**</Keywords>

<AddDate>**dateTime**</AddDate>

<AddBy>**string**</AddBy>

<ModDate>**dateTime**</ModDate>

<ModBy>**string**</ModBy>

<Channel>**string**</Channel>

<IsWaitlisted>**boolean**</IsWaitlisted>

<Culture>**string**</Culture>

<MediaType>**string**</MediaType>

<IsActive>**boolean**</IsActive>

<IsOnSite>**boolean**</IsOnSite>

<Latitude>**decimal**</Latitude>

<Longitude>**decimal**</Longitude>

<FloorMap>**string**</FloorMap>

<InternalNotes>**string**</InternalNotes>

<TotalRevenue>**decimal**</TotalRevenue>

<TotalRegistrations>**int**</TotalRegistrations>

<TotalCancels>**int**</TotalCancels>

<TotalSubstitutions>**int**</TotalSubstitutions>

<TargetAttendance>**int**</TargetAttendance>

<TotalIncompletes>**int**</TotalIncompletes>

</APIEvent>

<APIEvent>

<ID>**int**</ID>

<CustomerID>**int**</CustomerID>

<ParentID>**int**</ParentID>

<Status>**string**</Status>

<Title>**string**</Title>

<StartDate>**dateTime**</StartDate>

<EndDate>**dateTime**</EndDate>

<ActiveDate>**dateTime**</ActiveDate>

<ClientEventID>**string**</ClientEventID>

<TypeID>**int**</TypeID>

<Type>**string**</Type>

<City>**string**</City>

<State>**string**</State>

<Country>**string**</Country>

<CountryCode>**string**</CountryCode>

<PostalCode>**string**</PostalCode>

<LocationName>**string**</LocationName>

<LocationRoom>**string**</LocationRoom>

<LocationPhone>**string**</LocationPhone>

<LocationBuilding>**string**</LocationBuilding>

<LocationAddress1>**string**</LocationAddress1>

<LocationAddress2>**string**</LocationAddress2>

<TimeZone>**string**</TimeZone>

<Capacity>**int**</Capacity>

<CurrencyCode>**string**</CurrencyCode>

<Keywords>**string**</Keywords>

<AddDate>**dateTime**</AddDate>

<AddBy>**string**</AddBy>

<ModDate>**dateTime**</ModDate>

<ModBy>**string**</ModBy>

<Channel>**string**</Channel>

<IsWaitlisted>**boolean**</IsWaitlisted>

<Culture>**string**</Culture>

<MediaType>**string**</MediaType>

<IsActive>**boolean**</IsActive>

<IsOnSite>**boolean**</IsOnSite>

<Latitude>**decimal**</Latitude>

<Longitude>**decimal**</Longitude>

<FloorMap>**string**</FloorMap>

<InternalNotes>**string**</InternalNotes>

<TotalRevenue>**decimal**</TotalRevenue>

<TotalRegistrations>**int**</TotalRegistrations>

<TotalCancels>**int**</TotalCancels>

<TotalSubstitutions>**int**</TotalSubstitutions>

<TargetAttendance>**int**</TargetAttendance>

<TotalIncompletes>**int**</TotalIncompletes>

</APIEvent>

</Data>

<StatusCode>**int**</StatusCode>

<Authority>**int**</Authority>

</GetEventResult>

</GetEventResponse>

</soap:Body>

</soap:Envelope>

\*\*\*\*I only put the first Operation – to see the technical details of the other two operations go to ><https://www.regonline.com/api/> and find the corresponding operation.

1. The next two operations -2) GetRegistration &) - GetRegistrationsByEventID

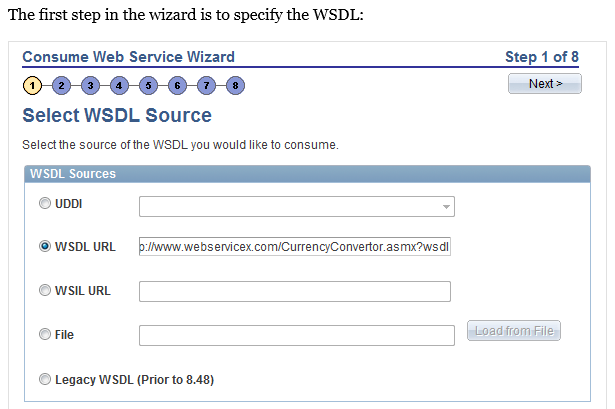
GetRegistration contained the financial dollar amounts, credit card information

GetRegistrationsbyEventID – contained both Events & registration ids, - this was used to JOIN THE DATA between GetEvent & GetRegistrations

1. **Consume WSDL:** In order to create the appropriate receiving structures in the PeopleSoft Financial system, I followed this document > <http://peoplesoft.wikidot.com/consuming-a-web-service>. This website gives a very simplified example of how consuming a WSDL - A webservice is any piece of software that makes itself available over the internet and uses a standardized XML messaging system. For example, a client invokes a web service by sending an XML message, then waits for a corresponding XML response. As all communication is in XML, web services are not tied to any one operating system or programming language--Java can talk with Perl; Windows applications can talk with Unix applications.

Here is the RegOnline WSDL. <https://www.regonline.com/api/default.asmx?WSDL>

We added this URL in the first step…



I used the delivered PeopleSoft WSDL wizard to consume this service, the next step creates some services, service operations, messages, and routings which secure the data in PeopleSoft and move the data to the custom table. I used the pagelet wizard shown to test that my connectivity was working properly.

In order to fully support this application I needed to know C+, javascript, HTML and XML.

**5) Custom Peoplesoft Table –** Our Peoplesoft system resided on an Oracle Database – I created a table using SQL called UV\_JRNL\_REGONLINE. I tied this custom table to a ROUTING in the Service operation created above.

**6) Custom Application Engine program** = Built in C+ & Peoplecode - reads from the CUSTOM table & performs next step.

**7)Insert into Delivered Peoplesoft Tables - SQL** Journal Entry data into JRNL\_HEADER, JRNL\_LINE, JRNL\_DETAIL