The back end used was MS SQL Server 2014. Here is the schema for the database

Table Name – Claim

Columns – ClaimNumber(nvarchar), ClaimantFirstName(nvarchar), ClaimantLastName(nvarchar), Status(nvarchar), LossDate(Date), AssignedAdjusterID(Int)

Table Name – Cause of Loss

Columns – CauseOfLoss(nvarchar), ReportedDate(Date), LossDescription(nvarchar), ClaimID(nvarchar)

Table Name – VehicleInfo

Columns – ModelYear(int), MakeDescription(nvarchar), ModelDescription(nvarchar), EngineDescription(nvarchar), ExteriorColor(nvarchar), Vin(nvarchar), LicPlate(nvarchar), LicPlateState(nvarchar), LicPlateExpDate(Date), DamageDescription(nvarchar), Mileage(Int), ClaimId(nvarchar)

Claim.ClaimNumber , CauseOfLoss.ClaimID and VehicleInfo.ClaimId are the primary keys for the corresponding tables. Claim.ClaimNumber acts as a foreign key for the other two tables. Assumption – the vehicles list contains only one instance per claim as opposed to a list of vehicles.

Please let me know should you require more information on the data modle

Currently using JDBC for making the connection, retrieving and storing the values to/from database to java objects. Although this process can be greatly simplified by using ORM mapping framework.

-----------------------------------------------------------------------------------------------------------------------------------------

The webservice is basically comprised of a Java Class(hiThere.java) and its methods :

-helloName : for inserting a claim into the backend

-readClaim : for getting a claim based on a claim id

-getListOfClaims : for getting claimIds for claims within a given date range

The webservice was setup in Eclipse using the Axis2 plug-in (Service Archive Generator Wizard). Based on the methods of the core webservice class, the plug in sets up the webservice in Tomcat and generates the other required files(Web.xml, WSDL etc)

**Client**

The webservice client is primarily implemented by a servlet called Start.java. It is placed inside the ‘Mitchell’ project. The index.html is the starting point where the user is displayed the options – Insert a claim, search claim OR find claims within a date range. The submit action on index.html is handled by Start.java. ‘Start’ then constructs soap messages and makes the calls to corresponding methods of hiThere.java(Webservice). Currently, I am not unmarshalling the xml data received from the webservice into java objects. Instead, directly dumping the xml output onto the screen for the end user to view. Again using the JAXB API for binding data to/from xml would prove beneficial.

*Insert a claim* – The user has an option of inserting a claim into database by uploading a valid claims file. The uploaded claims file is validated against the mitchellClaim.xsd schema. The xsd file needs to be placed inside the Mitchell Webcontent folder. NOTE – *the claims file that has been provided, fails to pass the test as the VIN number is not placed in the order defined by the xsd. Hence either the position of the VIN field needs to be changed OR the xsd needs to be modified (can be rectified by changing <xs:sequence> to <xs:all>. Also the LicPlateExpDate has a date field which is not conforming to a date pattern. Hence for the sake of making the application work, the date field in the xml was modified to fit the ‘yyyy-mm-dd’ format)*

The uploaded claims file is then appended with a helloName xml tag and the whole document is sent off to the webservice as a soap message.

The projects from Eclipse(version - eclipse-jee-indigo-SR2-win32-x86\_64) : ‘hiThere’ and Mitchell’, have been exported into JARS. Please set up eclipse accordingly to run the client. Once set up, the index page can be accessed using the following link -http://localhost:8080/Mitchell/

Here are a couple of test cases that can be used to test the application(Please note that some of these test cases may not pass when validated against the actual project)

TestCases –

* User upload a file an xml file with wrong schema. Error message to be displayed. Please enter a valid xml
* User enters wrong date formats. Error message can be displayed or instead, datepickers can be used.
* User enters end date that occurs before the start date. Prompt user to enter the right dates.
* User enters the wrong claim number- display claim not found