Web Services

This assignment will give you practical experience on building a self-hosted web service with modern frameworks, **Nancy** in this case. It also brings together various bits from the previous assignments, A#5, A#6, A#7.

For a general view on **Nancy**, please follow the links given in the Nancy folder and experiment with the given examples (now updated).

The server will be started on port 8081 (which is available in the labs) and will listen at simple **POST** requests of the following multi-line format:

- The first line indicates a **table-name**, one of **Customers** or **Orders**.
- Each one of the next lines indicate a **required transformation** - in a format similar to the one used in A#7.

Our application contains a folder **called** data, where we have two XML serialised versions of the "standard" **Northwind** tables **Customers** and **Orders**, here called **Customers.XML** and **Orders.XML** (respectively).

Essentially, we have to parse the XML "tables" into corresponding collections, apply the required transformations, in the given order, and finally return the result as a JSON formatted response.

For parsing XML, we recommend extension methods from the **System.Xml.Linq** namespace, such as **.Elements** (*name*), **.Element** (*name*).

The XML data must be parsed into objects of this format, exactly (otherwise the request may fail):

```
public class Customer {
    public string CustomerID { get; set; }
    public string CompanyName { get; set; }
    public string ContactName { get; set; }
}
```

```
public class Order {
    public int OrderID { get; set; }
    public string CustomerID { get; set; }
    public DateTime? OrderDate { get; set; }
    public DateTime? ShippedDate { get; set; }
    public decimal? Freight { get; set; }
    public string ShipName { get; set; }
    public string ShipCity { get; set; }
    public string ShipCountry { get; set; }

    // more about this field later:
    public Customer Customer { get; set; }
}
```

Note that Order has three nullable properties (?), which need special care when parsing! For the moment, also ignore the property Order.Customer.

The required transformations use method names as in A#7, i.e., **Where**, **Select**, **OrderBy**, **Take**, **Skip**, which means that you can still use **Dynamic Linq**.

However, the element types of the sequences can be **arbitrary tuples** (not just strings as in A#7).

The assignment has several levels of difficulty, which will be separately assessed.

Level 1 – 4 marks

We should be able to work with transformations starting from the **Customers** table. For example:

POST Request Body:

```
Customers

OrderBy ContactName DESC

Where CustomerID < "D"

Take 3

Select new (CustomerID, ContactName, CompanyName)
```

POST Response (minified):

```
[{"customerID":"CHOPS","contactName":"Yang
Wang","companyName":"Chop-suey
Chinese"},{"customerID":"BSBEV","contactName":"Victoria
Ashworth","companyName":"B's
Beverages"},{"customerID":"AROUT","contactName":"Thomas
Hardy","companyName":"Around the Horn"}]
```

Level 2 – +2 marks (total 6 marks)

We should **also** be able to work with transformations starting from the **Orders** table, which do NOT use the nullable fields (so you can start by ignoring these while parsing). For example:

POST Request Body:

```
Orders

OrderBy OrderID

Take 3

Select new (OrderID, ShipCountry, CustomerID)
```

POST Response (minified):


```
[{"orderID":10248,"shipCountry":"France","customerID":"VINET"},{"orderID":10249,"shipCountry":"Germany","customerID":"TOMSP"},{"orderID":10250,"shipCountry":"Brazil","customerID":"HANAR"}]
```


Level 3 – +1 marks (total 7 marks)

We should **also** be able to work with transformations starting from the **Orders** table, which DO use the nullable fields (so now you need to parse the fields). For example:

POST Request Body:

```
Orders

Where OrderDate != null && (ShippedDate == null || Freight == null)

Take 3

Select new (OrderID, OrderDate, ShippedDate, Freight)
```

POST Response (minified):


```
[{"orderID":11008,"orderDate":"1998-04-08T00:00:00.0000000+12:00","shippedDate":null,"freight":79.4600},{"orderID":11009,"orderDate":"1998-04-08T00:00:00.0000000+12:00","shippedDate":null,"freight":59.1100},{"orderID":11010,"orderDate":"1998-04-09T00:00:00.0000000+12:00","shippedDate":null,"freight":28.7100}]
```

Level 4 – +1 marks (total 8 marks)

We should **also** be able to work with transformations that require navigation from the **Orders** table to the associated **Customer**, which means that you have to conceptually "join" the two tables and set the **Order.Customer** link. For example:

POST Request Body:

```
Orders

OrderBy OrderID

Take 3

Select new (OrderID, ShipCountry, CustomerID, Customer.ContactName,
Customer.CompanyName)
```

POST Response (minified):


```
[{"orderID":10248,"shipCountry":"France","customerID":"VINET","contact
Name":"Paul Henriot *******","companyName":"Vins et alcools
Chevalier"},{"orderID":10249,"shipCountry":"Germany","customerID":"T
OMSP","contactName":"Karin Josephs","companyName":"Toms
Spezialitäten"},{"orderID":10250,"shipCountry":"Brazil","customerID":"H
ANAR","contactName":"Mario Pontes","companyName":"Hanari Carnes"}]
```

Programs:

(A#8) one **single file** C# solution (required).

You can of course use all standard libraries extant in the labs, plus **Nancy.dll**, **Nancy.Hosting.Self.dll**, **DynamicLinqUoA.dll**.

Please don't forget to add proper **using** clauses to your source program.

Submission

Please submit to the COMSPCI **ADB** (not to automarker).

https://adb.auckland.ac.nz/