Федеральное государственное автономное

образовательное учреждение

высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт Космических и информационных технологий

институт

Кафедра «Информатика»

кафедра

**ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ №3**

Преподаватель \_\_\_\_\_\_\_\_\_\_\_ А.К. Погребников

подпись, дата инициалы, фамилия

Студент КИ15–16Б, 031510065 \_\_\_\_\_\_\_\_\_\_\_ Т.В. Радионов

номер группы, зачетной книжки подпись, дата инициалы, фамилия

Красноярск 2018

# Задача лабораторной работы

1. Для схемы данных - разработанной в практической работе №1 реализовать REST приложение. (Допускается использование собственной структуры данных - требуется предварительное согласование с преподавателем)
2. REST приложение должно корректно обрабатывать следующие методы: DELETE, GET, POST, PUT
3. Должны быть предусмотрены обработки метода GET как для конкретного элемента по ID, так и для полного списка элементов целевой таблицы

# Листинг программы

PW1.Item.cls

Class PW1.Item Extends (%Persistent, %Populate, %XML.Adaptor)  
{  
  
Property Price As %Double(MAXVAL = 10000, MINVAL = 1, SCALE = 2) [ Required ];  
  
Property ProductName As %String(MAXLEN = 12) [ Required ];  
  
Index ProductNameIndex On ProductName [ Unique ];  
  
Query GetAllItems() As %SQLQuery(CONTAINID = 1)  
{  
 SELECT \* FROM Item  
}  
  
Query GetItem(ID As %Integer) As %SQLQuery(CONTAINID = 1)  
{  
 SELECT ProductName,Price FROM Item WHERE id = :ID  
}  
  
}

PW1.Order.cls

Class PW1.Order Extends (%Persistent, %Populate, %XML.Adaptor)  
{  
  
Property Items As list Of PW1.Item [ Required ];  
  
// Property DateTime As %DateTime [ Required ];  
  
Property Price As %Double(SCALE = 2) [ Calculated, Required, SqlComputeCode = {set {Price} = ##class(PW1.Order).CalcPrice({ID})  
}, SqlComputed ];  
  
Method PriceGet() As %Double [ CodeMode = expression, ServerOnly = 1 ]  
{  
##class(PW1.Order).CalcPrice(..Items)  
}  
  
ClassMethod CalcPrice(ID As %Integer) As %Double  
{  
 set o = ..%OpenId(ID)  
 set sum = 0  
 for i=1:1:o.Items.Count()  
 {  
 s item = o.Items.GetAt(i)  
 set sum = sum + item.Price  
 }  
 Quit sum  
}  
  
Property StatusTime As %TimeStamp [ Required ];  
  
Property Status As %String(VALUELIST = "-formed-confirmed-sent-completed") [ Required ];  
  
}

REST.Broker.cls

Class REST.Broker Extends %CSP.REST  
{  
  
Parameter UseSession As Integer = 1;  
  
XData UrlMap  
{  
<Routes>  
<Route Url="/test" Method="GET" Call="Test"/>  
<Route Url="/SetSession/:sessionID" Method="PUT" Call="SetSession"/>  
<Route Url="/GetAllItems" Method="GET" Call="GetAllItems"/>  
<Route Url="/GetItem/:itemID" Method="GET" Call="GetItem"/>  
<Route Url="/AddItem/:itemID" Method="POST" Call="AddItem"/>  
<Route Url="/RemoveItem/:itemID" Method="DELETE" Call="RemoveItem"/>  
<Route Url="/FormingOrder/:itemsCount" Method="POST" Call="FormingOrder"/>  
</Routes>  
}  
  
ClassMethod SetSession(sessionID As %String) As %Status  
{  
 s %session = %session.GetSession(sessionID)  
 //set st = {"status": "OK", "sessionID": (%session.SessionId), "createTime": (%session.CreateTime)}.%ToJSON()  
 //set session = ##class(%CSP.Session).%OpenId(sessionID)  
 write %session.SessionId  
 return $$$OK  
}  
  
ClassMethod Test() As %Status  
{  
    write "{""Status"": ""OK""}"  
    return $$$OK  
}  
  
ClassMethod GetAllItems() As %Status  
{  
 //set items = ##class(PW1.Item).GetAllItemsFunc().%ToJSON()  
 //do ##class(%ZEN.Auxiliary.jsonSQLProvider).%WriteJSONFromSQL(,items)  
 do ##class(%ZEN.Auxiliary.jsonSQLProvider).%WriteJSONFromSQL(,"SELECT \* FROM PW1.Item")  
 //write items  
 return $$$OK  
}  
  
ClassMethod GetItem(itemID As %Integer) As %Status  
{  
 //set items = ##class(PW1.Item).GetAllItemsFunc().%ToJSON()  
 //do ##class(%ZEN.Auxiliary.jsonSQLProvider).%WriteJSONFromSQL(,items)  
 do ##class(%ZEN.Auxiliary.jsonSQLProvider).%WriteJSONFromSQL(,"SELECT ProductName,Price FROM PW1.Item WHERE id = " \_ itemID)  
 //write items  
 return $$$OK  
}  
  
ClassMethod AddItem(itemID As %Integer) As %Status  
{  
 set itemsID = %session.Get("itemsID")  
 set itemCount = %session.Get("itemCount" \_ itemID) + 1  
 set find = $find(itemsID, itemID)  
 if (find '= 0)  
 {  
 do %session.Set("itemCount" \_ itemID, itemCount)  
 }  
 else  
 {  
 if ((itemsID = "") ! (itemsID = ","))  
 {  
 set itemsID = itemsID \_ itemID  
 }  
 else  
 {  
 set itemsID = itemsID \_ "," \_ itemID   
 }  
 do %session.Set("itemsID",itemsID)  
 do %session.Set("itemCount"\_itemID, itemCount)  
 }  
 return $$$OK  
}  
  
ClassMethod RemoveItem(itemID As %Integer) As %Status  
{  
 set itemsID = %session.Get("itemsID")  
 set find = $find(itemsID, itemID)  
 if (find '= 0)  
 {  
 set itemCount = %session.Get("itemCount" \_ itemID)  
 if (itemCount = 1 ! 0)  
 {  
 set itemCount = 0  
 set str1 = $extract(itemsID, 1, find - 3)  
 set length = $length(itemsID)  
 set str2 = $extract(itemsID, find, length)  
 set itemsID = str1 \_ str2  
 do %session.Set("itemsID", itemsID)  
 do %session.Set("itemCount" \_ itemID, itemCount)  
 }  
 else  
 {  
 do %session.Set("itemCount" \_ itemID, itemCount - 1)  
 }  
 }  
 //write itemsID  
 return $$$OK  
}  
  
ClassMethod FormingOrder(itemsCount As %String) As %Status  
{  
 set order = ##class(PW1.Order).%New()  
 for i=1:1:itemsCount  
 {  
 set itemCount = %session.Get("itemCount"\_i)  
 if (itemCount > 0)  
 {  
 set item = ##class(PW1.Item).%OpenId(i)  
 for j=1:1:itemCount  
 {  
 do order.Items.Insert(item)   
 }  
 set order.StatusTime = $ZDATETIME($NOW(), 3)  
 set order.Status = "formed"  
 }  
 }  
 set st = order.%Save()  
 return $$$OK  
}  
  
}

# Описание программы

На рисунке 1 можно ознакомиться с процессом конфигурирования REST-приложения для интернет магазина.

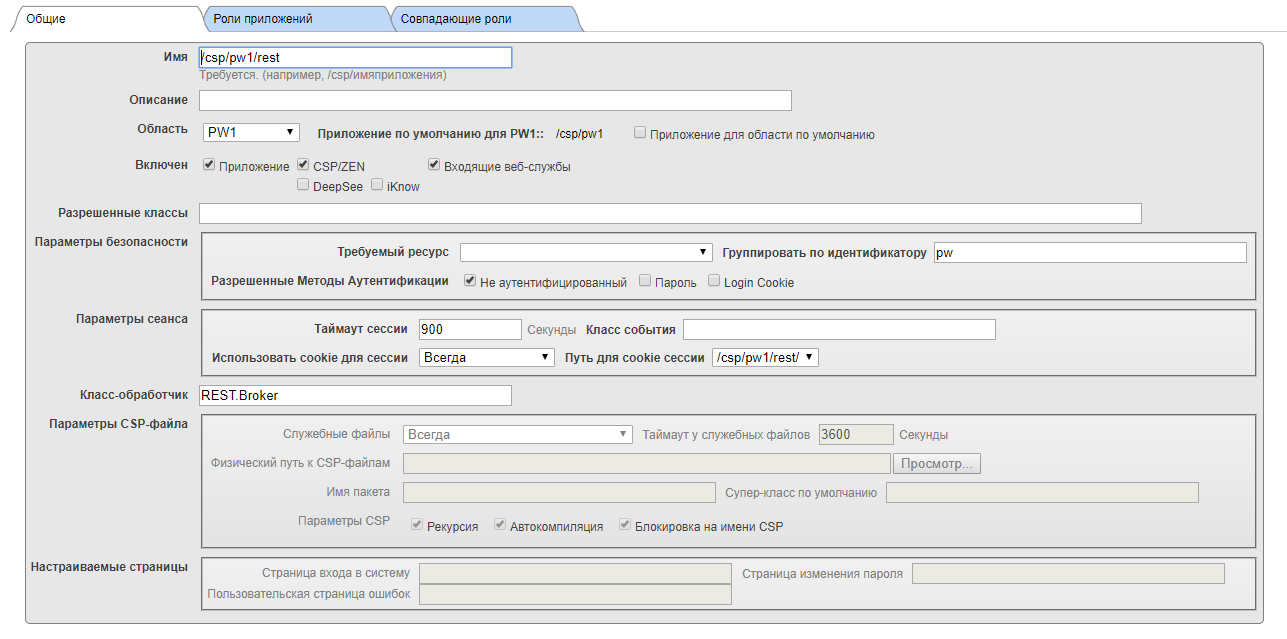


Рисунок 1 – Конфигурация REST-приложения

В разделе «[Листинг программы](#_Листинг_программы)» внутри класса REST.Broker.cls описаны различные виды методов, обозначенных в задаче, в которых используется работа как с одним элементом, так и со всеми.