**SessionLog.Html**

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<title>Orders</title>

<script src="../../../AdminFilesScripts/InventoryManagement/Purchase/Orders.js"></script>

<style>

table-condensed thead > tr > th, .table-condensed tbody > tr > th, .table-condensed tfoot > tr > th, .table-condensed thead > tr > td, .table-condensed tbody > tr > td, .table-condensed tfoot > tr > td {

padding: 2px;

}

.table th, .table td {

border-top: none !important;

}

body {

font-size: 11px;

font-weight: normal;

font-family: Arial;

}

</style>

</head>

<body>

<div>

<table class="subheader table-condensed" width="100%" style="margin:1px;font-size:10px" id="subheader\_tab">

<tr style="border:1px solid black">

<td class="col-md-2" align="center" style="border:1px solid black;background-color:rgb(92,177,227)"><a href="#/JIT\_" style="color:rgb(38,63,106)"><u>JIT OC Overview</u></a></td>

<td class="col-md-2" align="center" style="border:1px solid black;"><a href="#/JIT\_O" style="color:rgb(38,63,106)"><u>JIT Item Fulfillment</u></a></td>

<td class="col-md-2" align="center" style="border:1px solid black"><a href="#/JIT\_F" style="color:rgb(38,63,106)"><u>JIT Facility Fulfillment</u></a></td>

<td class="col-md-2" align="center" style="border:1px solid black"><a href="#/JIT\_D" style="color:rgb(38,63,106)"><u>JIT Dispatched</u></a></td>

<td class="col-md-2" align="center" style="border:1px solid black"><a href="#/JIT\_Report" style="color:rgb(38,63,106)"><u>JIT Reporting</u></a></td>

<td class="col-md-2" align="center" style="border:1px solid black;background-color:rgb(92,177,227)"> <a href="" style="color:rgb(38,63,106)"><u>SessionLog</u></a></td>

</table>

<!--<table class="subheader table-condensed" width="100%" style="margin:1px;font-size:10px" id="subheader\_tab">

<tr style="border:1px solid black">

<td class="col-md-2" align="center" style="border:1px solid black;"><a href="#/Invt" style="color:rgb(38,63,106)"><u>On Hand Order Confirmations</u></a></td>

<td class="col-md-2" align="center" style=" border: 1px solid black"><a href="#/Invt\_stock" style="color:rgb(38,63,106)"><u>On Hand Items</u></a></td>

<td class="col-md-2" align="center" style="border:1px solid black"><a href="#/Invt\_transaction" style="color:rgb(38,63,106)"><u>On Hand Transaction History</u></a></td>

<td class="col-md-2" align="center" style="border:1px solid black"><a href="#/Invt\_stockReport" style="color:rgb(38,63,106)"><u>On Hand Reporting</u></a></td>

<td class="col-md-2" align="center" style="border:1px solid black"><a href="#/Invt\_allocation" style="color:rgb(38,63,106)"><u>On Hand Allocation</u></a></td>

</tr>

</table>-->

<h4><label>SessionLog</label></h4>

<div class="row">

<div class="col-md-6">

<form name="CountryForm" id="CountryForm">

<table width="100%" style="background-color: #e8e8e8">

<tr style="background-color:grey;color:white">

<td colspan="6">

<label>Search Criteria</label>

</td>

</tr>

<tr>

<td>

<label>From Date:</label>

</td>

<td>

<div class="input-group">

<input type="text" id="fromDate" ng-model="fromDate" ng-disabled="true" size="15">

<span><button type="button" id="btnPicker1" custom-calendar><span class="glyphicon glyphicon-calendar"></span></button></span>

</div>

</td>

<td>

<label>To Date:</label>

</td>

<td>

<div class="input-group">

<input type="text" id="toDate" ng-model="toDate" ng-disabled="true" size="15">

<span><button type="button" id="btnPicker2" custom-calendar><span class="glyphicon glyphicon-calendar"></span></button></span>

</div>

</td>

<!--<td>

<label>SessionLogID:</label>

</td>

<td>

<input type="text" placeholder="LogID" ng-model="LogID" name="LogID">

</td>-->

</tr>

<tr>

<td><strong>SessionID</strong></td>

<td>

<input type="text" placeholder="SessionID" ng-model="SessionID" name="SessionID">

</td>

<td>

<label> TransactionType:</label>

</td>

<td>

<input type="text" size="20" id="headerId" placeholder="Type" ng-model="Type" name="Type">

</td>

</tr>

<tr>

<td>

<label> UserName:</label>

</td>

<td>

<input type="text" size="20" placeholder="UserName" ng-model="UserName" name="UserName">&nbsp;

</td>

<td>

<label> Email:</label>

</td>

<td>

<input type="text" id="inputEmail" placeholder="Email" ng-model="Email" name="Email">

</td>

</tr>

<tr>

<td><strong>CmpID</strong></td>

<td>

<input type="text" ng-model="CmpID" placeholder="CmpID" size="20" />

</td>

<td colspan="1">&nbsp;</td>

<td align="center"><input type="submit" value="Search" ng-click="search('SessionLogID')"></td>

</tr>

</table><br />

</form>

</div>

</div>

<div class="row">

<div class="col-md-12 col-sm-12 col-xs-12">

<div class="table-responsive">

<table class="table table-condensed table-striped table-hover" width="100%" id="country">

<thead style="background-color:gray;">

<tr style="background-color:gray;border:3px solid white">

<td colspan="8" style="line-height:10px"><label style="color:white">Search Results : {{sessions.TotalRecords}}</label></td>

</tr>

<tr style="background-color:gray;border:1px solid white">

<td colspan="8">

<button ng-disabled="sessions.TotalPages==0 || sessions.TotalPages == 1 || sessions == 1" ng-click="Firstpage()">&lt;&lt;</button>

<button ng-disabled="sessions.TotalPages==0 || sessions.TotalPages == 1 || currentpage == 1" ng-click="paging(0)">&lt;</button>

<label style="color:white">page {{currentpage}} of {{sessions.TotalPages>0?sessions.TotalPages:1}}</label>

<button ng-disabled="sessions.TotalPages==0 || sessions.TotalPages == 1 || currentpage == sessions.TotalPages" ng-click="paging(1,sessions.TotalPages)">&gt;</button>

<button ng-disabled="sessions.TotalPages==0 || sessions.TotalPages == 1 || currentpage == sessions.TotalPages" ng-click="Lastpage(sessions.TotalPages)">&gt;&gt;</button>

<!--<span ng-if="purchaseOrders.Data && (purchaseOrders.Data.length>0)">&nbsp;<input type="button" value="Export to Excel" ng-click="export()"></span>-->

<select ng-change="changeaction()" ng-model="pagesize" ng-options="opt as opt for opt in pagesizes" style="float:right;height:25px"></select>

</td>

</tr>

<tr>

<!--<td colspan="1"><a href="" style="color:white" ng-click="sort('SessionLogID')"><strong>Session LogID</strong></a></td>-->

<td colspan="1"><a href="" style="color:white" ng-click="sort('SessionID')"><strong>SessionID</strong></a></td>

<!--<td colspan="3"><a href="" style="color:white" ng-click="sort('RequestXML')"><strong>RequestXML</strong></a></td>

<td colspan="2"><a href="" style="color:white" ng-click="sort('ResponseXML')"><strong>ResponseXML</strong></a></td>-->

<td colspan="1"><a href="" style="color:white" ng-click="sort('CreatedDate')"><strong>CreatedDate</strong></a></td>

<td colspan="1"><a href="" style="color:white" ng-click="sort('TransactionType')"><strong>TransactionType</strong></a></td>

<td colspan="1"><a href="" style="color:white" ng-click="sort('CmpID')"><strong>Client</strong></a></td>

<td colspan="1"><a href="" style="color:white" ng-click="sort('UserName')"><strong>UserName</strong></a></td>

<td colspan="1"><a href="" style="color:white" ng-click="sort('Email')"><strong>Email</strong></a></td>

<td colspan="1" style="color:white"><strong>RequestXML</strong></td>

<td colspan="1" style="color:white"><strong>ResponseXML</strong></td>

</tr>

</thead>

<tbody>

<tr ng-repeat="session in sessions.Data">

<!--<td colspan="1" align="left">{{session.SessionLogID}}</td>-->

<td colspan="1" align="left">{{session.SessionID}}</td>

<!--<td colspan="3" align="left">{{session.RequestXML}}</td>

<td colspan="2" align="left">{{session.ResponseXML}}</td>-->

<td colspan="1" align="left">{{session.CreatedDate|date:medium}}</td>

<td colspan="1" align="left">{{session.TransactionType}}</td>

<td colspan="1" align="left">{{session.CmpID}}</td>

<td colspan="1" align="left">{{session.UserName}}</td>

<td colspan="1" align="left">{{session.Email}}</td>

<td colspan="1"><span ng-if="session.RequestXML != null"><input type="button" ng-click="viewrequest(session)" value="ViewRequest" /><p id="demo"></p></span></td>

<td colspan="1"><span ng-if="session.ResponseXML != null"><input type="button" ng-click="viewresponse(session)" value="ViewResponse" /></span></td>

</tr>

<tr ng-if="sessions.Data.length==0">

<td colspan="8" align="center"><strong>No Records Found</strong></td>

</tr>

</tbody>

</table>

</div>

</div>

</div>

</div>

</body>

</html>

**SessionLog.js**

app.controller('OrdersCtrl', OrdersCtrl);

OrdersCtrl.$inject = ["$scope", "$location", "FilesData", "FilesCRUDService", "$rootScope"];

function OrdersCtrl($scope, $location, FilesData, FilesCRUDService, $rootScope) {

function createdateFormat(date) {

var months = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec'];

return months[date.getMonth()] + " " + date.getDate() + ", " + date.getFullYear();

}

var currDate = new Date();

var fromDate = new Date();

fromDate = new Date(fromDate.setFullYear(fromDate.getFullYear() - 1));//createdateFormat(currDate);

$scope.fromDate = createdateFormat(fromDate);

$scope.toDate = createdateFormat(currDate);

var isSearchPerformed = false;

$scope.Statuses = ['Requested', 'Approved', 'Submitted', 'Confirmed', 'Received'];

$scope.pagesizes = [5, 10, 25, 50, 75, 100, '\*'];

$scope.pagesize = $scope.pagesizes[3];

if ($scope.pagesize == '\*') {

$scope.pagesize = 0;

}

$scope.currentpage = 1;

loaddata();

function loaddata() {

showpageloading(true);

var data = {

SessionLogID: '',

SessionID: '',

TransactionType: '',

CmpID: '',

UserName: '',

Email: '',

FromDate: fromDate.toISOString(),

ToDate: currDate.toISOString(),

pageNo: 1,

pageSize: $scope.pagesize,

sortField: 'CreatedDate',

sortAsc: false

};

var loadsessiondata = FilesCRUDService.getAllSessionLogs(data);

loadsessiondata.then(function (res) {

$scope.sessions = res.data;

showpageloading(false);

},

function (errorPl) {

$scope.error = 'failure in data retrieving', errorPl;

});

}

$scope.sort = function (columnName) {

$scope.Order = !$scope.Order;

if (!columnName)

columnName = "";

isSearchPerformed = true;

searchCountryList($scope, true, columnName);

}

$scope.search = function (columnName) {

if (!columnName)

columnName = "";

isSearchPerformed = true;

searchCountryList($scope, false, columnName);

}

function searchCountryList($scope, isPaging, colName) {

$scope.sortcol = colName;

if (!isPaging) {

$scope.currentpage = 1

}

if (!$scope.LogID)

$scope.LogID = '';

if (!$scope.SessionID)

$scope.SessionID = '';

if (!$scope.Type)

$scope.Type = '';

if (!$scope.UserName)

$scope.UserName = '';

if (!$scope.CmpID)

$scope.CmpID = '';

if (!$scope.Email)

$scope.Email = '';

showpageloading(true);

var searchdata = {

SessionLogID: $scope.LogID,

SessionID: $scope.SessionID,

TransactionType: $scope.Type,

CmpID: $scope.CmpID,

UserName: $scope.UserName,

Email: $scope.Email,

FromDate: $('#fromDate').val(),

ToDate: $('#toDate').val(),

pageNo: $scope.currentpage,

Records: $scope.pagesize == '\*' ? 0 : $scope.pagesize,

//pageSize: $scope.pagesize == '\*' ? 0 : $scope.pagesize,

sortField: $scope.sortcol == undefined ? 'CreatedDate' : $scope.sortcol,

sortAsc: $scope.Order == undefined ? false : $scope.Order

};

var loadsessiondata1 = FilesCRUDService.getAllSessionLogs(searchdata);

loadsessiondata1.then(function (res) {

$scope.sessions = res.data;

if ($scope.currentpage > res.data.TotalPages)

$scope.currentpage = 1;

showpageloading(false);

},

function (errorPl) {

$scope.error = 'failure in data saving', errorPl;

});

}

$scope.Firstpage = function () {

$scope.currentpage = 1;

searchCountryList($scope, true);

}

$scope.Lastpage = function (totalPages) {

$scope.currentpage = totalPages;

searchCountryList($scope, true);

}

$scope.paging = function (type, totalPages) {

if (type == 0 && $scope.currentpage > 0) {

--$scope.currentpage;

} else if (type == 1 && $scope.currentpage < totalPages) {

++$scope.currentpage;

}

searchCountryList($scope, true);

}

$scope.changeaction = function () {

var page = $scope.pagesize;

searchCountryList($scope, true);

}

$scope.viewrequest = function (requestxml) {

var viewrequestdata = FilesCRUDService.requestxmldata(requestxml);

viewrequestdata.then(function (res) {

$scope.xmldata = res.data;

punchoutdata = $scope.xmldata[0].punchdata[0].PunchOutSetup;

itemdata = $scope.xmldata[0].Myobj;

ipa.store("data1", punchoutdata);

ipa.store("itemdata", itemdata);

$location.path("/XmlDetails");

},

function (errorPl) {

$scope.error = 'failure in data saving', errorPl;

});

}

};

**FileService.js**

this.getAllSessionLogs = function (Obj, token) {

var url = rootUrl.API\_URL + "/api/PurchaseOrder/SessionSearch";

var config = {

responsetype: 'json',

headers: { 'Accept': 'application/json', 'Content-Type': 'application/x-www-form-urlencoded', 'Authorization': 'Bearer ' + token }

}

return $http.post(url, Obj, config).then(function (res) {

return res;

});

};

this.requestxmldata = function (xmlstring, token) {

var url = rootUrl.API\_URL + "/api/PurchaseOrder/requestxml/";

var config = {

responsetype: 'json',

headers: { 'Accept': 'application/json', 'Content-Type': 'application/x-www-form-urlencoded', 'Authorization': 'Bearer ' + token }

}

return $http.post(url, xmlstring, config).then(function (res) {

return res;

});

};

**PurchaseOrderController.cs**

[HttpPost]

[Route("api/PurchaseOrder/SessionSearch")]

public Result SessionSearch([FromBody]New\_Sessionlog FilterObj)

{

List<New\_Sessionlog> SearchResult = null;

Result output = null;

FilterObj.FromDate = FilterObj.FromDate.Date;

FilterObj.ToDate = FilterObj.ToDate.Date;

var skipIndex = FilterObj.Records \* (FilterObj.PageNo - 1);

var sortField = (FilterObj.SortField == "" || FilterObj.SortField == null) ? "SessionLogID" : FilterObj.SortField;

var SortAsc = FilterObj.SortAsc == null ? true : FilterObj.SortAsc;

try

{

SearchResult = (from s in db.SessionLogs

where

(s.SessionLogID == (FilterObj.SessionLogID) || FilterObj.SessionLogID == 0 || FilterObj.SessionLogID.ToString() == null) &&

(s.SessionID.Contains(FilterObj.SessionID) || FilterObj.SessionID == "" || FilterObj.SessionID == null) &&

(s.TransactionType.Contains(FilterObj.TransactionType) || FilterObj.TransactionType == "" || FilterObj.TransactionType == null) &&

(s.CmpID.Contains(FilterObj.CmpID) || FilterObj.CmpID == "" || FilterObj.CmpID == null) &&

(s.UserName.Contains(FilterObj.UserName) || FilterObj.UserName == "" || FilterObj.UserName == null) &&

(s.Email.Contains(FilterObj.Email) || FilterObj.Email == "" || FilterObj.Email == null) &&

(s.CreatedDate >= FilterObj.FromDate && s.CreatedDate <= FilterObj.ToDate)

select new New\_Sessionlog

{

SessionLogID = s.SessionLogID,

SessionID = s.SessionID,

RequestXML = s.RequestXML,

ResponseXML = s.ResponseXML,

CreatedDate = s.CreatedDate,

TransactionType = s.TransactionType,

CmpID = s.CmpID,

UserName = s.UserName,

Email = s.Email

}).OrderByField(sortField, SortAsc).ToList();

if (FilterObj.Records == 0)

{

output = new Result

{

Data = SearchResult,

TotalRecords = SearchResult.Count(),

TotalPages = 1

};

}

else

{

var pages = SearchResult.Count() % FilterObj.Records == 0 ? SearchResult.Count() / FilterObj.Records : (SearchResult.Count() / FilterObj.Records) + 1;

if (FilterObj.PageNo > pages)

{

skipIndex = 0;

}

output = new Result

{

Data = SearchResult.Skip(skipIndex).Take(FilterObj.Records),

TotalRecords = SearchResult.Count(),

TotalPages = pages

};

}

}

catch (Exception)

{

}

return output;

}

[HttpPost]

[Route("api/PurchaseOrder/requestxml")]

public dynamic XmlRequestSession([FromBody]New\_Sessionlog xmlstring)

{

//List<New\_PunchOutSetup> SearchResult = null;

//Result output = null;

var xmlstr = xmlstring.RequestXML;

var resstr = xmlstring.ResponseXML;

//List<dynamic> Myobj = null;

var Myobj = new List<dynamic>();

var MyDataobj = new List<dynamic>();

List<dynamic> New\_PunchOutSetup = null;

var punchdata = new List<dynamic>();

var xDoc = XDocument.Parse(xmlstr);

if (xmlstring.TransactionType == "PunchOutSetup") {

var from = xDoc.Root.Element("Header");

var Fromvalue = from.Element("From");

var FromCredential = Fromvalue.Element("Credential");

var FromCredential1 = FromCredential.Attribute("domain");

var FromCredentialValue = FromCredential1.Value;

var FromIdentity = FromCredential.Element("Identity");

var FromIdentityValue = FromIdentity.Value;

var Tovalue = from.Element("To");

var ToCredential = Tovalue.Element("Credential");

var ToCredential1 = ToCredential.Attribute("domain");

var ToCredentialValue = ToCredential1.Value;

var ToIdentity = ToCredential.Element("Identity");

var ToIdentityValue = ToIdentity.Value;

var Sendervalue = from.Element("Sender");

var SenderCredential = Sendervalue.Element("Credential");

var SenderCredential1 = SenderCredential.Attribute("domain");

var SenderCredentialValue = SenderCredential1.Value;

var SenderIdentity = SenderCredential.Element("Identity");

var SenderIdentityValue = SenderIdentity.Value;

var SS = SenderCredential.Element("SharedSecret");

var SSValue = SS.Value;

var SUserAgent = Sendervalue.Element("UserAgent");

var SUserAgentValue = SUserAgent.Value;

var request = xDoc.Root.Element("Request");

var PunchOutSetupRequest = request.Element("PunchOutSetupRequest");

var Operation = PunchOutSetupRequest.Attribute("operation");

var OperationValue = Operation.Value;

var BuyerCookie = PunchOutSetupRequest.Element("BuyerCookie");

var BuyerCookieValue = BuyerCookie.Value;

XElement FirstNameValue = xDoc.Descendants("Extrinsic")

.Where(x => (string)x.Attribute("name") == "FirstName").FirstOrDefault();

XElement LastNameValue = xDoc.Descendants("Extrinsic")

.Where(x => (string)x.Attribute("name") == "LastName").FirstOrDefault();

XElement UniqueNameValue = xDoc.Descendants("Extrinsic")

.Where(x => (string)x.Attribute("name") == "UniqueName").FirstOrDefault();

XElement EmailValue = xDoc.Descendants("Extrinsic")

.Where(x => (string)x.Attribute("name") == "UserEmail").FirstOrDefault();

XElement Uservalue = xDoc.Descendants("Extrinsic")

.Where(x => (string)x.Attribute("name") == "User").FirstOrDefault();

XElement BusinessValue = xDoc.Descendants("Extrinsic")

.Where(x => (string)x.Attribute("name") == "BusinessUnit").FirstOrDefault();

var FirstName = FirstNameValue.Value;

var LastName = LastNameValue.Value;

string UniqueName = UniqueNameValue.Value;

string Email = EmailValue.Value;

var User = Uservalue.Value;

var BusinessUnit = BusinessValue.Value;

var BrowserFormPost = PunchOutSetupRequest.Element("BrowserFormPost");

var BrowserFormPostValue = BrowserFormPost.Value;

var Contact = PunchOutSetupRequest.Element("Contact");

var ContactName = Contact.Element("Name");

var ContactNameValue = ContactName.Value;

var ContactEmail = Contact.Element("Email");

var ContactEmailValue = ContactEmail.Value;

var SupplierSetup = PunchOutSetupRequest.Element("SupplierSetup");

var SupplierSetupValue = SupplierSetup.Value;

var yDoc = XDocument.Parse(resstr);

var Response = yDoc.Root.Element("Response");

var PunchOutSetupResponse = Response.Element("PunchOutSetupResponse");

var startpage = PunchOutSetupResponse.Element("StartPage");

var startpageValue = startpage.Value;

New\_PunchOutSetup PunchOutSetup = new New\_PunchOutSetup();

PunchOutSetup.FromDomain = FromCredentialValue;

PunchOutSetup.ToDomain = ToCredentialValue;

PunchOutSetup.SenderDomain = SenderCredentialValue;

PunchOutSetup.FromIdentity = FromIdentityValue;

PunchOutSetup.ToIdentity = ToIdentityValue;

PunchOutSetup.SenderIdentity = SenderIdentityValue;

PunchOutSetup.SharedSecret = SSValue;

PunchOutSetup.UserAgent = SUserAgentValue;

PunchOutSetup.Operation = OperationValue;

PunchOutSetup.BuyerCookie = BuyerCookieValue;

PunchOutSetup.FirstName = FirstName;

PunchOutSetup.LastName = LastName;

PunchOutSetup.UniqueName = UniqueName;

PunchOutSetup.UserEmail = Email;

PunchOutSetup.User = User;

PunchOutSetup.BusinessUnit = BusinessUnit;

PunchOutSetup.BrowserFormPost = BrowserFormPostValue;

PunchOutSetup.ContactName = ContactNameValue;

PunchOutSetup.ContactEmail = ContactEmailValue;

PunchOutSetup.SupplierSetup = SupplierSetupValue;

PunchOutSetup.StartPage = startpageValue;

PunchOutSetup.TransactionType = xmlstring.TransactionType;

punchdata.Add(new { PunchOutSetup });

}

else

{

var from = xDoc.Root.Element("Header");

var Fromvalue = from.Element("From");

var FromCredential = Fromvalue.Element("Credential");

var FromCredential1 = FromCredential.Attribute("domain");

var FromCredentialValue = FromCredential1.Value;

var FromIdentity = FromCredential.Element("Identity");

var FromIdentityValue = FromIdentity.Value;

var Tovalue = from.Element("To");

var ToCredential = Tovalue.Element("Credential");

var ToCredential1 = ToCredential.Attribute("domain");

var ToCredentialValue = ToCredential1.Value;

var ToIdentity = ToCredential.Element("Identity");

var ToIdentityValue = ToIdentity.Value;

var Sendervalue = from.Element("Sender");

var SenderCredential = Sendervalue.Element("Credential");

var SenderCredential1 = SenderCredential.Attribute("domain");

var SenderCredentialValue = SenderCredential1.Value;

var SenderIdentity = SenderCredential.Element("Identity");

var SenderIdentityValue = SenderIdentity.Value;

var SUserAgent = Sendervalue.Element("UserAgent");

var SUserAgentValue = SUserAgent.Value;

var message = xDoc.Root.Element("Message");

var deply = message.Attribute("deploymentMode");

var deplyValue = deply.Value;

var PunchOutOrderMessage = message.Element("PunchOutOrderMessage");

var BuyerCookie = PunchOutOrderMessage.Element("BuyerCookie");

var BuyerCookieValue = BuyerCookie.Value;

var PunchOutOrderMessageHeader = PunchOutOrderMessage.Element("PunchOutOrderMessageHeader");

var Operation = PunchOutOrderMessageHeader.Attribute("operationAllowed");

var OperationValue = Operation.Value;

var Quotestatus = PunchOutOrderMessageHeader.Attribute("quoteStatus");

var QuotestatusValue = Quotestatus.Value;

var Total = PunchOutOrderMessageHeader.Element("Total");

var TotalMoney = Total.Element("Money");

var TotalMoneyValue = TotalMoney.Value;

var Shipping = PunchOutOrderMessageHeader.Element("Shipping");

var ShippingMoney = Shipping.Element("Money");

var ShippingMoneyValue = ShippingMoney.Value;

var ShippingDescr = Shipping.Element("Description");

var ShippingDescrValue = ShippingDescr.Value;

var Tax = PunchOutOrderMessageHeader.Element("Tax");

var TaxMoney = Tax.Element("Money");

var TaxMoneyValue = TaxMoney.Value;

var TaxDescr = Tax.Element("Description");

var TaxDescrValue = TaxDescr.Value;

New\_PunchOutSetup PunchOutSetup = new New\_PunchOutSetup();

PunchOutSetup.FromDomain = FromCredentialValue;

PunchOutSetup.ToDomain = ToCredentialValue;

PunchOutSetup.SenderDomain = SenderCredentialValue;

PunchOutSetup.FromIdentity = FromIdentityValue;

PunchOutSetup.ToIdentity = ToIdentityValue;

PunchOutSetup.SenderIdentity = SenderIdentityValue;

PunchOutSetup.UserAgent = SUserAgentValue;

PunchOutSetup.DeploymentMode = deplyValue;

PunchOutSetup.Operation = OperationValue;

PunchOutSetup.QuoteStatus = QuotestatusValue;

PunchOutSetup.BuyerCookie = BuyerCookieValue;

PunchOutSetup.TotalMoney = TotalMoneyValue;

PunchOutSetup.ShippingMoney = ShippingMoneyValue;

PunchOutSetup.ShippingDescr = ShippingDescrValue;

PunchOutSetup.TaxMoney = TaxMoneyValue;

PunchOutSetup.TaxDescr = TaxDescrValue;

PunchOutSetup.TransactionType = xmlstring.TransactionType;

punchdata.Add(new { PunchOutSetup });

var ItemIn = PunchOutOrderMessage.Element("ItemIn");

foreach (XElement item in PunchOutOrderMessage.Elements("ItemIn"))

{

var Quantity = item.Attribute("quantity");

var QuantityValue = Quantity.Value;

var ItemID = item.Element("ItemID");

var SupplierPartID = ItemID.Element("SupplierPartID");

var SupplierPartIDValue = SupplierPartID.Value;

var SupplierPartAuxiliaryID = ItemID.Element("SupplierPartAuxiliaryID");

var SupplierPartAuxiliaryIDValue = SupplierPartAuxiliaryID.Value;

var ItemDetail = item.Element("ItemDetail");

var UnitPrice = ItemDetail.Element("UnitPrice");

var UnitPriceValue = UnitPrice.Value;

var ItemDescr = ItemDetail.Element("Description");

var ItemDescrValue = ItemDescr.Value;

var UOM = ItemDetail.Element("UnitOfMeasure");

var UOMValue = UOM.Value;

var Classification = ItemDetail.Element("Classification");

var ClassificationValue = Classification.Value;

var LeadTime = ItemDetail.Element("LeadTime");

var LeadTimeValue = LeadTime.Value;

New\_QuantityOrder QuantityOrder = new New\_QuantityOrder();

QuantityOrder.Quantity = QuantityValue;

QuantityOrder.SupplierPartID = SupplierPartIDValue;

QuantityOrder.SupplierPartAuxillaryID = SupplierPartAuxiliaryIDValue;

QuantityOrder.UnitPrice = UnitPriceValue;

QuantityOrder.Description = ItemDescrValue;

QuantityOrder.UnitofMeasure = UOMValue;

QuantityOrder.Classification = ClassificationValue;

QuantityOrder.LeadDetail = LeadTimeValue;

var QuantityData = QuantityOrder;

Myobj.Add( new{ QuantityData});

};

}

Myobj.ToArray();

MyDataobj.Add(new { Myobj, punchdata });

return MyDataobj;

}

}

public class New\_Sessionlog

{

public int SessionLogID { get; set; }

public string SessionID { get; set; }

public string RequestXML { get; set; }

public string ResponseXML { get; set; }

public Nullable<System.DateTime> CreatedDate { get; set; }

public string TransactionType { get; set; }

public string CmpID { get; set; }

public string UserName { get; set; }

public string Email { get; set; }

public DateTime FromDate { get; set; }

public DateTime ToDate { get; set; }

public int PageNo { get; set; }

public int Records { get; set; }

public string SortField { get; set; }

public bool SortAsc { get; set; }

}

public class New\_PunchOutSetup

{

public string FromDomain { get; set; }

public string TransactionType { get; set; }

public string ToDomain { get; set; }

public string SenderDomain { get; set; }

public string FromIdentity { get; set; }

public string ToIdentity { get; set; }

public string SenderIdentity { get; set; }

public string SharedSecret { get; set; }

public string UserAgent { get; set; }

public string BuyerCookie { get; set; }

public string FirstName { get; set; }

public string LastName { get; set; }

public string UniqueName { get; set; }

public string UserEmail { get; set; }

public string User { get; set; }

public string BusinessUnit { get; set; }

public string BrowserFormPost { get; set; }

public string ContactName { get; set; }

public string ContactEmail { get; set; }

public string SupplierSetup { get; set; }

public string ShippingMoney { get; set; }

public string ShippingDescr { get; set; }

public string TaxMoney { get; set; }

public string TotalMoney { get; set; }

public string TaxDescr { get; set; }

public string Operation { get; set; }

public string QuoteStatus { get; set; }

public string DeploymentMode { get; set; }

public string StartPage { get; set; }

}

public class New\_QuantityOrder

{

public string Quantity { get; set; }

public string SupplierPartID { get; set; }

public string SupplierPartAuxillaryID { get; set; }

public string UnitPrice { get; set; }

public string Description { get; set; }

public string UnitofMeasure { get; set; }

public string Classification { get; set; }

public string LeadDetail { get; set; }

}