**Documentation for server side pagination with angularJs**

\*\*\* Here is a short technique to implement the server side pagination with angularJs. First of all, we need mentioned required js file.

<script src="../Scripts/angular.js"></script>

<script src="../Scripts/ui-bootstrap-tpls-1.1.2.min.js"></script>

<script src="demandOrderListController.js"></script>

<link href="../Content/bootstrap.min.css" rel="stylesheet" />

**In demand-order-list.html**

\*\*\* In html file, we need to add mentioned code that contents pagination.

<tfoot>

<tr><td>

<span class="form-group pull-left page-size form-inline">

<select style="margin-top: 20px;"

class="ddlPageSize form-control control-color" ng-model="pageSizeSelected" ng-change="changePageSize()"> <option value="25" ng-selected="true">25</option> <option value="50">50</option> <option value="100">100</option> </select>

</span>

</td>

<td colspan="9">

<div class="pull-right">

<uib-pagination total-items="totalCount" ng-change="pageChanged(pageIndex)" items-per-page="pageSizeSelected" direction-links="true" ng-model="pageIndex" max-size="maxSize" class="pagination" boundary-links="true" rotate="false" num-pages="numPages">

</uib-pagination>

</div>

</td></tr>

</tfoot>

**In demandOrderController.js:**

\*\*\* Now in controller file, we have to use mentioned code to pass the page index and page size to service.js file.

$scope.maxSize = 5; // Limit number for pagination display number.

$scope.totalCount = 0; // Total number of items in all pages. Initialize as a zero

$scope.pageIndex = 1; // Current page number. First page is 1. --> $scope.pageSizeSelected = 25; // Maximum number of items per page.

$scope.pageChanged = function (currentPage) {

$scope.pageIndex = currentPage;

getDemandOrderList();

};

//This method is calling from dropDown

$scope.changePageSize = function () {

$scope.pageIndex = 1;

$scope.pageSizeSelected = parseInt($('.ddlPageSize').val());

getDemandOrderList();

};

var getDemandOrderList = function () {

authService.loadingOn();

$scope.totalCount = 0;

var promise = salesService.getDemandOrderList($scope.pageIndex, $scope.pageSizeSelected);

promise.then(function (response) {

$scope.doList = response;

if (response.length > 0) {

$scope.totalCount = response[0].TotalCount;

$scope.numPages = parseInt($scope.totalCount / $scope.pageSizeSelected) + 1;}

authService.loadingOff();

}, function (err) {

$scope.doList = [];

authService.loadingOff();});

};

\*\*\* **In salesService.js**

In salesService.js, this function needs to take page index and page size as parameter that has passed from controller.js. now this function has to pass these parameter again to SalesController.cs.

salesServiceFactory.getDemandOrderList = function (pageIndex, pageSize) {

var deferred = $q.defer();

$http.get(serviceBase + 'api/Sales/GetDemandOrderList/' + pageIndex + '/' + pageSize).success(function (response) {

deferred.resolve(response);

}).error(function (err, status) {

deferred.reject(err);

});

return deferred.promise;

};

\*\*\* **In SalesController.cs:**

In SalesController, the page index and page size is received then it has passed to ISalesInterfaces.cs.

[Route("GetDemandOrderList/{pageIndex}/{pageSize}")]

[HttpGet]

public async Task<IHttpActionResult> GetDemandOrderList(int pageIndex, int pageSize)

{

try

{

var userVm = await UserId();

int paymentStatus = (int)PaymentStatusEnum.All;

var demandOrderList = \_salesSvc.GetDemandOrderList(userVm.Id, paymentStatus, pageIndex, pageSize);

return Ok(demandOrderList);

}

catch (Exception ex)

{

\_logger.Log(RequestContext.Principal.Identity.Name, Request.RequestUri.AbsolutePath, ex);

return ResponseMessage(Request.CreateErrorResponse(HttpStatusCode.InternalServerError, ex.Message));

}

}

\*\*\* **In ISalesInterface.cs:**

In ISalesInterface, we need to add this prototype to that interface.

IList<DemandOrderVm> GetDemandOrderList(int userId, int paymentStatus, int pageIndex, int pageSize);

\*\*\* **In SalesService.cs:**

Now in SalesService, GetDemandOrderList has to implement first.

public IList<DemandOrderVm> GetDemandOrderList(int userId, int paymentStatus, int pageIndex, int pageSize)

{

return \_salesRepository.GetDemandOrderList(userId, paymentStatus, pageIndex, pageSize);

}

\*\*\* **In ISalesRepository.cs:**

In ISalesRepository, The mentioned prototype has to add in IRepository file.

IList<DemandOrderVm> GetDemandOrderList(int userId, int paymentStatus, int pageIndex, int pageSize);

\*\*\* **In SalesRepository.cs:**

Now it time to implement the add prototype in IRepository. Since page index and page size is passed from controller. So we have to make query by using page index and page size to get the specific data.

public IList<DemandOrderVm> GetDemandOrderList(int userId, int paymentStatus, int pageIndex, int pageSize)

{

var totalCount = 0;

var user = \_ppsDbContext.User.FirstOrDefault(x => x.Id == userId);

var employee = \_ppsDbContext.Employee.FirstOrDefault(x => x.Id == user.EmployeeId);

List<DemandOrder> doList;

if (employee != null && employee.DepartmentId == (int)DepartmentEnum.SalesAndMarketing)

{

var employeeRepository = new EmployeeRepository();

var employeeIdList = employeeRepository.GetManagedEmployee(employee.Id).Select(x => x.Item1);

if (paymentStatus == 0)

{

doList = \_ppsDbContext.DemandOrder.Where(x => employeeIdList.Contains(x.EmployeeId)).OrderBy(y => y.Id).Skip((pageIndex - 1) \* pageSize).Take(pageSize).ToList();

totalCount = \_ppsDbContext.DemandOrder.Where(x => employeeIdList.Contains(x.EmployeeId)).Count();

}

else

{

doList = \_ppsDbContext.DemandOrder.Where(x => x.PaymentStatusId == paymentStatus && employeeIdList.Contains(x.EmployeeId)).OrderBy(y => y.Id).Skip((pageIndex - 1) \* pageSize).Take(pageSize).ToList();

totalCount = \_ppsDbContext.DemandOrder.Where(x => x.PaymentStatusId == paymentStatus && employeeIdList.Contains(x.EmployeeId)).Count();

}

} else {

if (paymentStatus == 0)

{

doList = \_ppsDbContext.DemandOrder.OrderBy(y => y.Id).Skip((pageIndex - 1) \* pageSize).Take(pageSize).ToList();

totalCount = \_ppsDbContext.DemandOrder.Count();

} else {

doList = \_ppsDbContext.DemandOrder.Where(x => x.PaymentStatusId == paymentStatus).OrderBy(y => y.Id).Skip((pageIndex - 1) \* pageSize).Take(pageSize).ToList();

totalCount = \_ppsDbContext.DemandOrder.Where(x => x.PaymentStatusId == paymentStatus).Count();

}}

var doVm = new ConcurrentBag<DemandOrderVm>();

doList.ForEach(d =>

{

var totalPaid = d.DemandOrderTransaction.ToList().Sum(x => x.TransactionAmount);

var totalDue = d.TotalGrandAmount - totalPaid;

int maturityDays = (DateTime.Today - d.DODate).Days;

int maturityLabel = (int)DoMaturityLabelEnum.Normal;

if (maturityDays > d.SaleType?.DurationInDays)

{

maturityLabel = (int)DoMaturityLabelEnum.OverDue;

}

else if (maturityDays > d.SaleType?.WarningInDays)

{

maturityLabel = (int)DoMaturityLabelEnum.Warning;

}

doVm.Add(new DemandOrderVm

{

Id = d.Id,

DODate = d.DODate,

SaleTypeName = d.SaleType?.SaleTypeName,

DemandOrderTypeName = d.DemandOrderType?.DemandOrderTypeName,

MaturityDays = maturityDays,

MaturityLabel = maturityLabel,

ReferenceNo = d.ReferenceDONo,

CustomerName = d.Customer.CustomerName,

DOStatusName = d.DemandOrderStatus.Status,

TotalGrandAmount = d.TotalGrandAmount,

CreatedByName = StringExtension.ToFullName(d.User.FirstName, d.User.LastName),

CreatedDate = d.CreatedOn,

Submitted = d.SubmittedBy != null,

DOPaymentStatusId = GetDemandOrderTransactionStatus(d, totalDue),

DOPaymentStatus = d.PaymentStatus.PaymentStatusName,

ProductTypeGroupId = d.DemandOrderDetail.First().Product.ProductType.ProductTypeGroupId,

ProductTypeGroupName = d.DemandOrderDetail.First().Product.ProductType.ProductTypeGroup.ProductTypeGroupName,

**TotalCount = totalCount**

});

});

return doVm.ToList();

}

\*\*\* **Note: highlights color is a modification into my existing code.**