

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using WebApplication4.Controllers.WebApplication2.Controllers;

namespace WebApplication1
{
    public class ChartData
    {
        public IEnumerable<TaskInfo> GetTaskTimelineChartReport(int LocationId, int PeriodId, int tenantId, int IndependentTask)
        {
            eClose_TestingEntities context = new eClose_TestingEntities();
            IEnumerable<TaskInfo> taskList;
            IEnumerable<TaskInfo> dependandantTaskList;
            List<TaskInfo> combineList = new List<TaskInfo>();
            List<TaskInfo> finalList = new List<TaskInfo>();

            //Check the dependant and Independent task
            if (IndependentTask == 1)
            {
                taskList = (from WorkbasketTask in context.WorkbasketTasks
                    join DependetWorkbasketTask in context.DependetWorkbasketTasks
                    on WorkbasketTask.Id equals DependetWorkbasketTask.WorkbasketTaskId into DependetWorkInfo
                    from DependetWorkbasketTask in DependetWorkInfo.DefaultIfEmpty()
                    join Period in context.Periods
                    on WorkbasketTask.PeriodId equals Period.Id
                    join Team in context.Teams.AsNoTracking()
                    on WorkbasketTask.TeamId equals Team.Id
                    join WorkbasketTaskUsers in context.WorkbasketTaskUsers
                    on WorkbasketTask.Id equals WorkbasketTaskUsers.WorkbasketTaskId
                    where WorkbasketTask.PeriodId == 94 && DependetWorkbasketTask.DependsOnWorkbasketTaskId == null
                    && Period.LocationId == 172
                    //106
                    //           where (Period.LocationId == LocationId) && (Period.Id == PeriodId) && (WorkbasketTask.TenantId == tenantId)
                    group WorkbasketTaskUsers by new
                    {
                        // WorkbasketTaskUsers.EndDateInt,
                        WorkbasketTask.Id,
                        DependetWorkbasketTask.DependsOnWorkbasketTaskId,
                        WorkbasketTask.TaskName,

```

```

        Team.TeamName
    } into gcs
    let topp = gcs.Max(x => x.WFLevel)
    select new TaskInfo
    {
        //TeamId = (gcs.Key.TeamId == null) ? 0 : gcs.Key.TeamId,
        TeamName = (gcs.Key.TeamName == null) ? "" : gcs.Key.TeamName,
        TaskName = (gcs.Key.TaskName == null) ? "" : gcs.Key.TaskName,
        WorkbasketTaskId = (gcs.Key.Id == null) ? 0 : gcs.Key.Id,
        DependsOnWorkbasketTaskId = gcs.Key.DependsOnWorkbasketTaskId,
        WFLevel = topp,
        // EndDateInt = gcs.Max(x => (gcs.Key.EndDateInt == null) ? DateTime.Today : gcs.Key.EndDateInt),
        value = 0,
        extra = "",
        IsDependantTask = false
    }).ToList<TaskInfo>();
}
else if (IndependentTask == 0)
{
    taskList = (from WorkbasketTask in context.WorkbasketTasks
        join DependetWorkbasketTask in context.DependetWorkbasketTasks
        on WorkbasketTask.Id equals DependetWorkbasketTask.WorkbasketTaskId into DependetWorkInfo
        from DependetWorkbasketTask in DependetWorkInfo.DefaultIfEmpty()
        join Period in context.Periods
        on WorkbasketTask.PeriodId equals Period.Id
        join Team in context.Teams
        on WorkbasketTask.TeamId equals Team.Id
        join WorkbasketTaskUsers in context.WorkbasketTaskUsers
        on WorkbasketTask.Id equals WorkbasketTaskUsers.WorkbasketTaskId
        where WorkbasketTask.PeriodId == 94 && DependetWorkbasketTask.DependsOnWorkbasketTaskId != null //106
        && Period.LocationId == 172 //
        where (Period.LocationId == LocationId) && (Period.Id == PeriodId) &&
        (WorkbasketTask.TenantId == tenantId)
        group WorkbasketTaskUsers by new
        {
            // WorkbasketTaskUsers.EndDateInt,
            WorkbasketTask.Id,
            DependetWorkbasketTask.DependsOnWorkbasketTaskId,
            WorkbasketTask.TaskName,
            Team.TeamName
        } into gcs
    let topp = gcs.Max(x => x.WFLevel)

```

```

select new TaskInfo
{
    //TeamId = (gcs.Key.TeamId == null) ? 0 : gcs.Key.TeamId,
    TeamName = (gcs.Key.TeamName == null) ? "" : gcs.Key.TeamName,
    TaskName = (gcs.Key.TaskName == null) ? "" : gcs.Key.TaskName,
    WorkbasketTaskId = (gcs.Key.Id == null) ? 0 : gcs.Key.Id,
    DependsOnWorkbasketTaskId = gcs.Key.DependsOnWorkbasketTaskId,
    WFLevel = topp,
    // EndDateInt = gcs.Max(x => (gcs.Key.EndDateInt == null) ? DateTime.Today : gcs.Key.EndDateInt),
    value = 0,
    extra = "",
    IsDependantTask = false
}).ToList<TaskInfo>();
}
else
{
    taskList = (from WorkbasketTask in context.WorkbasketTasks
                join DependetWorkbasketTask in context.DependetWorkbasketTasks
                on WorkbasketTask.Id equals DependetWorkbasketTask.WorkbasketTaskId into DependetWorkInfo
                from DependetWorkbasketTask in DependetWorkInfo.DefaultIfEmpty()
                join Period in context.Periods
                on WorkbasketTask.PeriodId equals Period.Id
                join Team in context.Teams
                on WorkbasketTask.TeamId equals Team.Id
                join WorkbasketTaskUsers in context.WorkbasketTaskUsers
                on WorkbasketTask.Id equals WorkbasketTaskUsers.WorkbasketTaskId
                where WorkbasketTask.PeriodId == 94 //106
                && Period.LocationId == 172 //                                where (Period.LocationId == LocationId) && (Period.Id == PeriodId) &&
                (WorkbasketTask.TenantId == tenantId)
                group WorkbasketTaskUsers by new
                {
                    // WorkbasketTaskUsers.EndDateInt,
                    WorkbasketTask.Id,
                    DependetWorkbasketTask.DependsOnWorkbasketTaskId,
                    WorkbasketTask.TaskName,
                    Team.TeamName
                } into gcs
                let topp = gcs.Max(x => x.WFLevel)
                select new TaskInfo
                {
                    //TeamId = (gcs.Key.TeamId == null) ? 0 : gcs.Key.TeamId,

```

```

TeamName = (gcs.Key.TeamName == null) ? "" : gcs.Key.TeamName,
TaskName = (gcs.Key.TaskName == null) ? "" : gcs.Key.TaskName,
WorkbasketTaskId = (gcs.Key.Id == null) ? 0 : gcs.Key.Id,
DependsOnWorkbasketTaskId = gcs.Key.DependsOnWorkbasketTaskId,
WFLevel = topp,
// EndDateInt = gcs.Max(x => (gcs.Key.EndDateInt == null) ? DateTime.Today : gcs.Key.EndDateInt),
value = 0,
extra = "",
IsDependantTask = false
}).ToList<TaskInfo>());
}

```

```

IEnumerable<int?> WorkbasketTaskIdList = taskList.Select(t => t.WorkbasketTaskId);
IEnumerable<int?> DependsOnWorkbasketTaskIdList = taskList.Where(x
!WorkbasketTaskIdList.Contains(x.DependsOnWorkbasketTaskId)).Select(t => t.DependsOnWorkbasketTaskId);
=>

```

```

dependantTaskList = (from WorkbasketTask in context.WorkbasketTasks
join Period in context.Periods
on WorkbasketTask.PeriodId equals Period.Id into Periodinfo
join Team in context.Teams
on WorkbasketTask.TeamId equals Team.Id into Teaminfo
from Team in Teaminfo.DefaultIfEmpty()
join WorkbasketTaskUsers in context.WorkbasketTaskUsers
on WorkbasketTask.Id equals WorkbasketTaskUsers.WorkbasketTaskId into WorkbasketTaskUsersinfo
from WorkbasketTaskUsers in WorkbasketTaskUsersinfo.DefaultIfEmpty()
join record in DependsOnWorkbasketTaskIdList
on WorkbasketTask.Id equals record
group WorkbasketTaskUsers by new
{
// WorkbasketTaskUsers.EndDateInt,
WorkbasketTask.Id,
WorkbasketTask.TaskName,
Team.TeamName
} into gcs
let topp = gcs.Max(x => x.WFLevel)
select new TaskInfo
{
//TeamId = (gcs.Key.TeamId == null) ? 0 : gcs.Key.TeamId,
TeamName = (gcs.Key.TeamName == null) ? "" : gcs.Key.TeamName,
TaskName = (gcs.Key.TaskName == null) ? "" : gcs.Key.TaskName,
WorkbasketTaskId = (gcs.Key.Id == null) ? 0 : gcs.Key.Id,

```

```

        DependsOnWorkbasketTaskId = (gcs.Key.Id == null) ? 0 : gcs.Key.Id,
        // EndDateInt = gcs.Max(x => (gcs.Key.EndDateInt == null) ? DateTime.Today : gcs.Key.EndDateInt),
        WFLevel = topp,
        value = 0,
        extra = "",
        IsDependantTask = true
    }).ToList<TaskInfo>();

```

```

combineList.AddRange(taskList);
combineList.AddRange(dependandantTaskList);

```

```

IEnumerable<int?> workBasketIdList = combineList.Select(x => x.WorkbasketTaskId).ToList();

```

```

IEnumerable<TaskInfo> workbasketTaskUserList = context.WorkbasketTaskUsers.
    Where ( x=> workBasketIdList.Contains( x.WorkbasketTaskId)).Select( x=> new TaskInfo
    {
        WFLevel= x.WFLevel,
        EndDateInt= x.EndDateInt,
        WorkbasketTaskId=x.WorkbasketTaskId
    }
    ).ToList<TaskInfo>() ;

```

```

try

```

```

{

```

```

    foreach (var task in combineList)

```

```

    {

```

```

        var _taskList = workbasketTaskUserList.Where(x => x.WorkbasketTaskId == task.WorkbasketTaskId && x.WFLevel == task.WFLevel &&
x.EndDateInt != null);

```

```

        DateTime? taskDate = new DateTime();

```

```

        int _maxWLevel = 0;

```

```

        int _maxWorkDay = 0;

```

```

        if (_taskList.Count() == 0)

```

```

        {

```

```

            var listForFindMaxLevel = (from u in workbasketTaskUserList

```

```

                where

```

```

                    (u.WorkbasketTaskId == task.WorkbasketTaskId

```

```

                    && u.EndDateInt != null)

```

```

                select new

```

```

        { WFLevel = u.WFLevel == null ? 0 : u.WFLevel }).ToList();
    if (listForFindMaxLevel.Count > 0)
        _maxWLevel = listForFindMaxLevel.Max(x => x.WFLevel);

    if (_maxWLevel != 0)
    {
        taskDate = workbasketTaskUserList.Where(x => x.WorkbasketTaskId == task.WorkbasketTaskId && x.WFLevel ==
_maxWLevel).First().EndDateInt;
    }
    else
    {
        var listWorkDay = context.PeriodDays.Where(x => x.Id == PeriodId).Select(x => x.WorkDay).ToList();

        _maxWorkDay = listWorkDay.Count() > 0 ? listWorkDay.Max(x => x.Value) : 0;

        taskDate = context.PeriodDays.Where(x => x.Id == PeriodId && x.WorkDay == _maxWorkDay).FirstOrDefault().CalenderDate;
    }
}
task.EndDateInt = _taskList.Count() == 0 ? taskDate : _taskList.FirstOrDefault().EndDateInt;
finalList.Add(task);
}
}
catch (Exception ex)
{
}

return (finalList);

}

}
}

```