Object Usage

Data Transfer Object - DTO

Data Transfer Object is the object returned by an API call.
A DTO is used to prevent exposing your database entity structure. This also prevent the exposing of data not being or to be used by the API client, such as linked entities. Data can also be returned as human readable if an API call is consumed by an UI client.

```
/// <summary>
/// Data Tranfer Object to transfer employee node rating information from and to the API
/// </summary>
public class EmployeeNodeRating
{
    /// <summary>
    /// The id of the rating
    /// </summary>
    public int Id { get; set; }
    /// <summary>
    /// The node id for pathway rated
    /// </summary>
    public int NodeId { get; set; }
    /// <summary>
    /// The rating value specified
    /// </summary>
    public int Rating { get; set; }
    /// <summary>
    /// The comment specified for ranking
    /// </summary>
    public string Comment { get; set; }
    /// <summary>
    /// The employee name for who rated node
    /// </summary>
    public string EmployeeName { get; set; }
    /// <summary>
    /// The rating date for ranking
    /// </summary>
```

```
    public string Date { get; set; }
    /// <summary>
    /// The image url for user rating
    /// </summary>
    public string ImageUrl { get; set; }
    /// <summary>
    /// The flag to indicate if comment is own comment
    /// </summary>
    public bool IsOwnComment { get; set; }

    /// <summary>
    /// Fetch the node rating details
    /// </summary>
    /// <param name="sessionHandler"></param>
    /// <param name="nodeId"></param>
    /// <returns></returns>
    public static EmployeeNodeRating Fetch(ISessionHandler sessionHandler, int nodeId)
    {
        var eagerLoadParms = new pwEmployeeRating.EagerLoadParameters
        {
            IncludeEmployees = true,
        };

        var empRating = pwEmployeeRating.TryFetchByNodeAndEmployee(sessionHandler, nodeId,
sessionHandler.EmployeeId.Value, eagerLoadParms);

        if (empRating == null)
            return null;
        else
            return new EmployeeNodeRating
            {
                Id = empRating.Id,
                NodeId = empRating.NodeId,
                EmployeeName = empRating.prsEmployee.NameSurname,
                Comment = empRating.Comment,
                Date = empRating.CreatedDate.ToShortDateString(),
                ImageUrl = String.Format("~/api/thumbnail/AspectRatioEmployee?id={0}&width=null&height=96",
empRating.prsEmployee.EmployeeNumber),
                IsOwnComment = true,
                Rating = empRating.Rating
```

```
        };
    }
}
```

Value/View Object - VO

A Value/View Object is generally the object received by an API call.
This is similar to a DTO in the sense that the underlying database entity is not exposed. A VO generally only contains properties for all the applicable, editable values to be submitted by the client; validation are also performed on these values prior to submitting data to data storage / database.

```
/// <summary>
/// View Object to transfer employee node rating information from the API
/// </summary>
public class EmployeeNodeRating
{
    /// <summary>
    /// The id for node
    /// </summary>
    public int NodeId { get; set; }
    /// <summary>
    /// The rating value specified
    /// </summary>
    public int Rating { get; set; }
    /// <summary>
    /// The comment specified for ranking
    /// </summary>
    public string Comment { get; set; }
    /// <summary>
    /// The flag to exclude comment from updating
    /// </summary>
    public bool IgnoreCommentChange { get; set; }

    /// <summary>
    /// Log the node rating for employee
    /// </summary>
    /// <param name="sessionHandler"></param>
    /// <returns></returns>
    public void LogRating(ISessionHandler sessionHandler)
```

```
    {
        var empRating = pwEmployeeRating.TryFetchByNodeAndEmployee(sessionHandler, this.NodeId,
sessionHandler.EmployeeId.Value);
        if (empRating == null)
        {
            empRating = new pwEmployeeRating
            {
                NodeId = this.NodeId,
                EmployeeId = sessionHandler.EmployeeId.Value,
                Rating = this.Rating,
                Comment = this.IgnoreCommentChange ? String.Empty : this.Comment.Trim(),
            };

            empRating.Create(sessionHandler);
        }
        else
        {
            empRating.Rating = this.Rating;
            empRating.Comment = this.IgnoreCommentChange ? empRating.Comment : this.Comment.Trim();
            empRating.Update(sessionHandler);
        }
    }
}
```

Revision #3
Created 17 September 2020 02:09:40 by Theuns Pretorius
Updated 8 October 2020 09:38:30