Хелперы

namespace ITino.ELMA.Common.Helpers

- ModelHelper
- ListenerHelper
- ServerHelper
- UserHelper
- WorkflowHelper

ModelHelper

```
/// <summary>
/// Прописано ли свойство в конфигурации
/// </summary>
/// <param name="pm">Метаданные свойства</param>
/// <returns></returns>
public static bool IsConfigProperty(PropertyMetadata pm)
```

```
/// <summary>
/// Получить Uid значения перечисления
/// </summary>
/// <param name="value">Значение перечисления</param>
/// <returns></returns>
public static Guid GetEnumUid(object value)
```

```
/// <summary>
/// Получить отображаемое описание значения перечисления
/// </summary>
/// <param name="value">Значение перечисления</param>
/// <returns></returns>
public static string GetEnumDescription(object value)
```

```
/// <summary>
/// Название типа сущности
/// </summary>
/// <param name="entity">Сущность</param>
/// <returns></returns>
public static string GetEntityDisplayName(IEntity entity)
```

```
/// <summary>
/// Описание типа сущности
/// </summary>
/// <param name="entity">Сущность</param>
/// <returns></returns>
public static string GetEntityDescription(IEntity entity)
```

```
/// <summary>
/// Проверить наличие элемента справочника и создать при необходимости
/// </summary>
/// <param name="uid">Uid элемента справочника</param>
/// <param name="values">Значения свойств</param>
/// <typeparam name="T">Тип сущности</typeparam>
public static void CheckEntityPresent<T>(Guid uid, object values) where T : IEntity
```

```
/// <summary>
/// Удалить элемент справочника (если есть)
/// </summary>
/// <param name="uid">Uid элемента справочника</param>
/// <typeparam name="T">Тип сущности</typeparam>
public static void RemoveEntityIfPresent<T>(Guid uid) where T : IEntity
```

> ELMA4

```
 /// <summary>
/// Обрезать часть даты и времени
/// </summary>
/// <param name="dateTime"></param>
/// <param name="timeSpan"></param>
/// <returns></returns>
public static DateTime Truncate(this DateTime dateTime, TimeSpan timeSpan)
```

```
/// <summary>
/// Обрезать миллисекунды
/// </summary>
/// <param name="dateTime"></param>
/// <returns></returns>
public static DateTime TruncateMs(this DateTime dateTime)
```

ListenerHelper

```
/// <summary>Получить старое значение</summary>
/// <typeparam name="T">Тип</typeparam>
/// <param name="event">Событие</param>
/// <param name="propertyName">Название свойства</param>
/// <returns>Старое значение</returns>
public static T GetOldValue<T>(PostUpdateEvent @event, string propertyName)
```

```
/// <summary>Получить старое значение</summary>
/// <typeparam name="T">Тип</typeparam>
/// <param name="event">Событие</param>
/// <param name="propertyName">Название свойства</param>
/// <returns>Старое значение</returns>
public static T GetOldValue<T>(PreUpdateEvent @event, string propertyName)
```

```
/// <summary>Получить значение свойства</summary>
/// <typeparam name="T">Тип</typeparam>
/// <param name="event">Событие</param>
/// <param name="propertyName">Название свойства</param>
/// <returns>Значение</returns>
public static T GetValue<T>(PostUpdateEvent @event, string propertyName)
```

```
/// <summary>Присвоить значение свойству</summary>
/// <param name="event">Событие</param>
/// <param name="propertyName">Название свойства</param>
/// <param name="value">Значение</param>
public static void SetValue(PreUpdateEvent @event, string propertyName, object value)
```

```
/// <summary>Получить значение свойства</summary>
/// <typeparam name="T">Тип</typeparam>
/// <param name="event">Событие</param>
/// <param name="propertyName">Название свойства</param>
/// <returns>Значение</returns>
public static T GetValue<T>(PostInsertEvent @event, string propertyName)
```

```
/// <summary>Присвоить значение свойству</summary>
/// <param name="event">Событие</param>
/// <param name="propertyName">Название свойства</param>
/// <param name="value">Значение</param>
public static void SetValue(PreInsertEvent @event, string propertyName, object value)
```

ServerHelper

```
/// <summary>
/// Получить среднее время запуска сервера
/// </summary>
public static string AverageStartTime
```

```
/// <summary>
/// Версия компонентов IEE
/// </summary>
public static string IEEVersion
```

```
/// <summary>
/// Возвращает признак что система в режиме тестирования
/// </summary>
public static bool IsTesting
```

```
/// <summary>
/// Перезапустить сервер
/// </summary>
public static void RestartServer()
```

```
/// <summary>
/// Логгер импорта
/// </summary>
public static ILog ImportLogger
```

```
/// <summary>
/// Запись в протокол импорта данных в справочник
/// </summary>
/// <param name="name">Имя справочника</param>
/// <param name="total">Всего записей</param>
/// <param name="new">Новых записей</param>
/// <param name="update">Обновлений</param>
/// <param name="delete">Удалений</param>
public static void ImportLogCatalog(string name, int total, int @new = 0, int update = 0, int delete = 0)
```

```
/// <summary>
/// Запись в протокол импорта что данные для справочника отсутствуют
/// </summary>
/// <param name="name">Имя справочника</param>
public static void ImportLogCatalogNone(string name)
```

```
/// <summary>
/// Время работы сервера
/// </summary>
/// <returns></returns>
public static TimeSpan UpTime
```

```
/// <summary>
/// Логгер обновления БД
/// </summary>
public static ILog DbUpdateLogger
```

```
/// <summary>
/// Логер пакетной обработки
/// </summary>
public static ILog ProcessLogger
```

UserHelper

```
/// <summary>
/// Системный пользователь
/// </summary>
public static IUser SystemUser
```

```
/// <summary>
/// Пользователь подсистемы обмена
/// </summary>
public static IUser ExchangeUser
```

```
/// <summary>
/// Получить непосредственного руководителя для пользователя
/// </summary>
/// <param name="user">Пользователь</param>
/// <returns>Руководитель (если есть, иначе null)</returns>
public static IUser GetImmediateChiefByUser(IUser user)
```

```
/// <summary>
/// Получить всех активных пользователей группы
/// </summary>
/// <param name="gUidStr">Строковый Uid группы</param>
/// <returns>Список пользователей</returns>
public static IEnumerable<IUser> GetUsersInGroup(string gUidStr)
```

```
/// <summary>
/// Получить всех активных пользователей группы
/// </summary>
/// <param name="gUid">Uid группы</param>
/// <returns>Список пользователей</returns>
public static IEnumerable<IUser> GetUsersInGroup(Guid gUid)
```

```
/// <summary>
/// Преобразовать полные инициалы в сокращенные
/// </summary>
```

```
/// <param name="fullName">Полные инициалы</param>
/// <param name="lastNameOrder">Фамилия в начале</param>
/// <returns>Сокращенные инициалы</returns>
public static string ConvertToShortName(string fullName, bool lastNameOrder = true)
```

```
/// <summary>
/// Входит ли пользователь в группу Администраторы
/// </summary>
/// <returns></returns>
public static bool IsAdmin
```

> Только ELMA4

```
/// <summary>
/// Отправить push уведомление текущему пользователю
/// </summary>
/// <param name="title">Заголовок уведомления</param>
/// <param name="message">Сообщение</param>
/// <param name="clickUrl">Ссылка в браузере при нажатии</param>
/// <param name="tag">Тэг (для группировки)</param>
/// <returns></returns>
public static bool WebPush(string message, string title = null, string clickUrl = null, string tag = null)
```

```
/// <summary>
/// Отправить push уведомление пользователю
/// </summary>
/// <param name="user">Пользователь</param>
/// <param name="title">Заголовок уведомления</param>
/// <param name="message">Сообщение</param>
/// <param name="clickUrl">Ссылка в браузере при нажатии</param>
/// <param name="tag">Тэг (для группировки)</param>
/// <returns></returns>
public static bool WebPush(IUser user, string message, string title = null, string clickUrl = null, string tag = null)
```

WorkflowHelper

```
/// <summary>
/// Получить запускаемые процессы по входному типу сущности
/// </summary>
/// <param name="typeUid">UID типа сущности</param>
/// <returns>Список процессов Workflow</returns>
public static IEnumerable<IWorkflowProcess> GetStartableProcessesByInputEntity(Guid typeUid)
```

```
/// <summary>
/// Комментарий для процессной задачи
/// </summary>
/// <param name="element">Uid элемента задачи на схеме</param>
/// <param name="task">Задача</param>
/// <param name="author">Автор комментария</param>
/// <param name="date">Дата комментария</param>
/// <param name="comment">Комментарий</param>
/// <param name="type">Тип комментария</param>
public static void CommentForTask(Guid element, ITaskBase task, IUser author, DateTime date, string comment,
EOWorkflowTaskCommentType type = EOWorkflowTaskCommentType.Info)
```

```
/// <summary>
/// Сохранить уведомление для последующего показа
/// </summary>
/// <param name="message">Текст сообщения</param>
/// <param name="type">Тип сообщения</param>
public static void Notify(string message, NotifyType type = NotifyType.Info)
```

```
/// <summary>
/// Убрать замещение у задачи процесса
/// </summary>
/// <param name="task">Задача процесса</param>
/// <param name="uid">Uid задачи на схеме</param>
/// <param name="ignoreUsers">Список игнорируемых пользователей</param>
public static void RemoveReplacement([NotNull] this ITaskBase task, Guid uid, IEnumerable<IUser> ignoreUsers
= null)
```

```
/// <summary>
/// Завершить процесс с выводом ошибки пользователю
/// </summary>
/// <param name="context"></param>
/// <param name="error"></param>
public static void CompleteWithError(object context, string error)
```

```
/// <summary>
/// Выполнить длительное действие в отдельном потоке при создании задачи
/// </summary>
/// <param name="element">Id задачи на диаграмме</param>
/// <param name="task">Созданная задача в OnTaskCreate</param>
/// <param name="action">Действие</param>
public static void ProcessLongAction(Guid element, ITaskBase task, Action<LongActionModel, ISession> action)
```