Хелперы

- ServerHelper
- ListenerHelper

ServerHelper

```
/// <summary>
/// Возвращает признак среды разработки
/// </summary>
public static bool IsDev
```

```
/// <summary>
/// Логгер интеграции
/// </summary>
public static ILog E365Logger { get; }
```

```
/// <summary>
/// Запустить процесс в ELMA365
/// </summary>
/// <param name="context">Контекст процесса монолита</param>
/// <param name="namespace">Пространство ELMA365</param>
/// <param name="code">Процесс ELMA365</param>
/// <param name="action">Модель контекста процесса ELMA365</param>
/// <param name="ack">Контроль выполнения в ELMA365</param>
/// <param name="async">Асинхронно</param>
/// <param name="throwException">Вызывать исключение</param>
/// <typeparam name="T">Класс контекста процесса</typeparam>
public static Guid E365StartProcess<T>(T context, string @namespace, string code,
Action<E365ProcessModel<T>> action, bool ack = false, bool async = false, bool throwException = false)
```

```
/// <summary>
/// Полная синхронизация справочника
/// </summary>
/// <param name="type">Тип сущности</param>
/// <returns></returns>
public static bool E365FullSync(Type type)
```

```
/// <summary>
/// Синхронизация системной информации (оргструктура и пользователи)
/// </summary>
```

```
/// <returns></returns>
public static bool E365SystemSync()
```

ListenerHelper

```
/// <summary>
/// Проверка на возможность мягкого удаления
/// </summary>
/// <param name="event">Событие</param>
/// <param name="action">Проверочное действие</param>
/// <typeparam name="T">Тип</typeparam>
public static void TrySoftDeleting<T>(PreUpdateEvent @event, Action<T> action) where T : class, IEntity
```

```
/// <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)
```