Класс ContentItem

Класс ContentItem реализует CRUD-функциональность для статей (создание, чтение, обновление, удаление) по той же логике, которая используется в бэкенде. Это бывает необходимо для создания пользовательских админок, а также для расширения QP7 с помощью пользовательских вкладок и действий. Для использования этого API необходим экземпляр класса DBConnector. В отличие от CRUD-методов класса DBConnector, методы класса ContentItem не кэшируют данные статей и поддерживают отправку уведомлений. Кроме того эти методы всегда работают с текущими версиями статей, а не с опубликованными.

Методы

New

```
public static ContentItem New(int contentId, DBConnector cnn)
```

Создание новой статьи в заданном контенте. Если контент не найден, генерируется исключение.

Read

```
public static ContentItem Read(int id, DBConnector cnn)
```

Чтение существующей статьи по ее идентификатору. Если статья не найдена, генерируется исключение.

Remove

```
public static void Remove(int id, DBConnector cnn)
```

Удаление существующей статьи по ее идентификатору. Если статья не найдена, ничего не происходит.

Save

```
public void Save()
```

Сохранение статьи (как новой, так и существующей). Внутри вызывает метод AddFormToContent класса DBConnector, а кроме этого отправляет уведомления.

LoadLastModifiedFromCustomTab

```
public void LoadLastModifiedFromCustomTab()
```

Попытка загрузить свойство LastModifiedBy из сессионной переменной, которая устанавливается при аутентификации пользователя в пользовательской вкладке методом CheckCustomTabAuthentication. Если она не удалась, остается предыдущее значение.

Свойства

FieldValues

```
public Dictionary<string, ContentItemValue> FieldValues {get; internal set;
}
```

Хэш-таблица значений полей статьи. В отличие от хэш-таблицы *Values*, использующейся в методах класса DBConnector, в данном случае в качестве ключей используются реальные имена полей, задаваемые в QP7. Регистр имеет значение! В качестве значения используется экземпляр класса *ContentItemValue*, в котором есть 2 свойства, отвечающие за данные статей. Данные полей типа M2M и M2O доступны через свойство *LinkedItems*, а всех полей остальных типов – через свойство *Data*:

```
public string Data { get; set; }
public HashSet<int> LinkedItems {get; internal set; }
```

При создании новой или чтения существующей статьи хэш-таблица с информациях о всех полях и их значениях загружается автоматически. Для новой статьи значения по умолчанию не подгружаются, но при сохранении учитываются, если соответствующие поля пришли пустыми.

Id

```
public int Id { get; set; }
```

Идентификатор статьи. Для новой статьи – 0. Чтобы создать копию статьи в том же контенте, достаточно прочитать статью методом Read, сбросить значение Id в 0 и сохранить статью методом Save.

Visible

```
public bool Visible { get; set; }
```

Флаг видимости статьи. По умолчанию - true;

Archive

```
public bool Archive { get; set; }
```

Флаг, определяющий, находится ли статья в архиве. По умолчанию - false;

DelayedSchedule

```
public bool DelayedSchedule { get; set; }
```

Флаг отложенной публикации. По умолчанию - false

LastModifiedBy

```
public int LastModifiedBy { get; set; }
```

Идентификатор пользователя, последним модфицировавшим статью. По умолчанию - 1. Для интеграции с пользовательскими вкладками можно воспользоваться методом LoadLastModifiedFromCustomTab.

StatusName

```
public string StatusName { get; set; }
```

Имя статуса. По умолчанию - *Published*, но если на контент назначено workflow, то – *None*.

ContentId

```
public int ContentId { get; set; }
```

Идентификатор контента. При изменении статья будет перемещена.

Created

```
public DateTime Created { get; internal set; }
```

Дата создания статьи.

Modified

```
public DateTime Modified { get; internal set; }
```

Дата последней модификации статьи.

Splitted

```
public bool Splitted { get; internal set; }
```

Флаг расщепления статьи. Изменяется косвенно при изменении статуса.

Пример

```
ContentItem item = ContentItem.New(284, cnn);
item.FieldValues["Title"].Data = "Test";
item.FieldValues["Date"].Data = DateTime.Today.ToString();
item.FieldValues["Category"].LinkedItems.Add(1660);
item.LoadLastModifiedFromCustomTab();
item.Save();
```