Как работают права

# Модель данных

Может возвращать права для каждого объекта. Поле [!!Permissions] – битовая маска.

А может и для ModelInfo в наборе [!$System!]. Поле [!PropertyName!Permissions].

Для объекта будет определено свойство $permissions с булевыми полями вида canXXXX.

В ModelInfo появится свойство Permissions с целым значением и добавиться аналогичное объектное свойство $permissions.

Для ModelInfo обычно используется только CanCreate/CanEdit, чтобы запрещать создание объектов. Это потому, что при создании объекта еще нет и проверять права негде. Приходится проверять в коллекции.

## Особенности LazyArray

Все то же самое, но не забываем указать поле [!!Permissions] в прототипе объекта (в основном наборе). Обратите внимание, что если вы не забудете указать [!!Permissions] в протоипе, но забудете указать в основном наборе загрузки, предупреждений не будет, но все права буду запрещены.

# UI

Свойства $permissions используются в основном для управления пользовательским интерфейсом.

Логично использовать свойства самого объекта для Edit/Delete, а свойства из ModelInfo для Create.

В привязках BindCmd есть свойство Permission. Оно будет проверять свойство $permissions аргумента команды.

# model.json

Дополнительно можно проверять права на сервере. Для этого надо загрузить набор прав во время загрузки меню. Там права на все объекты системы и битовые маски. Все это упаковывается в куки UserState.

В самом model.json можно добавить объект Permissions типа

"permissions": {  
 "Data.Element": "edit"  
}

Перед выполнением команды будет проверяться, есть ли соответствующий флаг в нужном объекте.

Чтобы это работало, нужно вызвать

services.UsePermissions() при инициализации.

В сегменте пользователя должна быть процедура [{securityschema}].PemissionObjects.

Она должна вернуть список всех защищаемых объектов системы. Они общие для всех тенантов.  
Возвращается простой список Id (bigint), Name nvarchar(255).

Name – это тот ключ, который мы будем проверять в model.json

Вместе с меню должен возвращаться массив Permissions с двумя полями – Id (bigint) и Flags – битовая маска разрешений. Id – идентификатор защищаемого объекта.

Маска разрешений соответствует перечислению

public enum PermissionFlag

{

CanView = 1,

CanEdit = 2,

CanDelete = 4,

CanApply = 8,

CanUnapply = 16,

CanCreate = 32,

CanFlag64 = 64,

CanFlag128 = 128,

CanFlag256 = 256,

}

Эти флаги кодируются в строку вида “Id1:FF,Id2:F0” и записываются в куки UserState. Если Id защищаемых объектов будут поменьше – сэкономим размер куки.

Проверка работает так.

При загрузке меню загружается фактический синглтон защищаемых объектов.

Если в model.json указано permissions, то для текущего юзера расшифровываются все объекты. Потом последовательно проверяются. Если хоть одно не разрешено – выдается исключение AccessDenied.

Если в model.json есть permissions, а вызова UsePermissions не было – выдается исключение.

Сами имена и их Id – это фактически Singleton. Они не меняются в процессе работы приложения.

# Дополнительно

В меню можно вернуть:

select [UserState!TUserState!Object] = null, [ReadOnly] = cast(0 as bit), IsAdmin = cast(1 as bit);