Управление правами доступа

Управление правами доступа – сложная вопрос, который не имеет однозначного ответа. Поскольку платформа A2v10 работает на достаточно низком уровне, возможная реализация управления правами полностью зависит от разработчика прикладного приложения.

Рассмотрим одну из возможностей, которая встроена в систему “из коробки”. Она основывается на группах пользователей и списках контроля доступа. Все таблицы и хранимые процедуры, связанные с этой реализацией находятся в схеме a2security.

# Общая информация

## Пользователи и группы

В системе имеются учетные записи пользователей (таблица a2security.Users), группы пользователей (таблица a2security.Groups) и таблица принадлежности пользователя к группе (таблица a2security.UserGroups).

Имеется две предопределенных группы с “магическими” идентификаторами – “Все пользователи” (GroupId = 1) и “Администраторы” (GroupId = 77).

## Списки контроля доступа (ACL)

Списки контроля доступа построены на трехзначной логике. Конкретное действие может быть “разрешено”, “запрещено” или “не установлено”. Если “не установлено”, это означает, что право выполнять это действие будет получено из других источников. Это может быть, например, группа пользователей или верхний элемент иерархии элементов.

Все права хранятся в одной таблице a2security.Acl.

В существующей реализации используется “запретительная логика” – все, что не разрешено явно – запрещено.

## Flattening

Поскольку при выполнении хранимых процедур очень важна производительность, то в системе выполняется “уплощение (flattening)” таблиц контроля доступа. Этот термин означает, что создается таблица «плоских» прав доступа, в которой каждому пользователю (не группе!) и каждому проверяемому элементу (например пункту меню) соответствует одна запись.

Такую плоскую таблицу очень просто объединять с таблицами сущностей и фильтровать результат по нужным флажкам.

Для каждого из защищаемых элементов системы создается своя плоская таблица. По соглашению имеет имя a2security.[Element.Acl]. В текущей реализации есть две таких таблицы:

* a2security.[Module.Acl] – работа с модулями
* a2security.[Menu.Acl] – работа с меню

Каждая такая плоская таблица имеет ссылку на пользователя (не группу!), защищаемый элемент и плоский набор флажков.

## Права на действия

Фактически право на действие это просто один из флажков (логика трехзначная). В примере реализации используются 5 флажков:

* **CanView** – пользователь видит элемент
* **CanEdit** – пользователь может редактировать элемент
* **CanDelete** – пользователь может удалять элемент
* **CanApply** – пользователь может «проводить или применять» (что бы это не значило) элемент
* **CanUnapply** - пользователь может отменять «проведение» (что бы это не значило) элемент

Два последних флажка связаны с реализацией учетных систем.

Отметим, что перечисленные ниже права всего лишь одна из реализаций. Разработчик прикладного решения вправе расширять их по своему усмотрению. Есть одно ограничение – флажков может быть до 32, поскольку все они упаковываются в целое 32-битное значение.

# Общий алгоритм работы

* Создаются пользователи, группы и пользователи включаются в группы. Это выполняется стандартными средствами платформы.
* Заполняются списки контроля доступа
* При любом изменении, затрагивающем права доступа (изменились права, добавлен/удален пользователь, пользователь включен/исключен в группу) – вызывается хранимая процедура a2security.[Permission.UpdateUserInfo].
* Эта хранимая процедура последовательно вызывает все процедуры "уплощения" таблиц доступа по маске Permission.UpdateAcl.%. Каждая из этих процедур обновляет соответствующую плоскую таблицу контроля прав.

Обратите внимание! Предложенных алгоритм хорошо работает при относительно небольшом количестве пользователей и защищаемых элементов (100 пользователей и 200-300 элементов – хорошая ориентировочная цифра). Если больше – потребуется оптимизация или вообще другая реализация работы с правами.

# Признак модели Read-only

При загрузке любой модели, помимо анализа флажка CanView, который будет отвечать за видимость элемента можно использовать флажок CanEdit.

Если необходимо, чтобы элемент можно было видеть, но нельзя было редактировать, то в элементе нужно вернуть специальное поле c признаком [!!ReadOnly] (значение может быть bit или int). Если это поле установлена, платформа автоматически выведет все элементы управления в режиме Read-оnly (это поведение можно изменить, указав в элементе атрибут Render=”Show”. Это может быть необходимо, чтобы, например, разрешить какую-то кнопку).

Обратите внимание, что этот флажок не ограничивается работой с правами. Можно, например, запретить редактирование проведенного документа.

Не полагайтесь на флажок Read-оnly. Не забывайте проверять права доступа в хранимой процедуре.

# Поле элемента Permissions и работа на клиенте

В некоторых случаях желательно знать конкретные права доступа на клиенте. Можно вернуть соответствующий набор данных стандартными способами, но существует еще один вариант.

Если в наборе данных [!$System!] существует поле c суффиксом [!Permissions], то в соответствующий элемент (или массив) модели будет добавлено свойство $permissions, возвращающее объект с булевыми свойствами canView, canEdit, canDelete, canApply, canUnapply.

Поле [!Permissions] задает права в виде битовой маски:

|  |  |
| --- | --- |
| canView | 0x01 |
| canEdit | 0x02 |
| canDelete | 0x04 |
| CanApply | 0x08 |
| CanUnapply | 0x10 |

Например, если вернуть такой набор,

select [!$System!] = null, [!Documents!Permissions] = 1 + 2

то в коллекцию Documents будет добавлено свойство

$permissions: {  
 canView: true,  
 canEdit: true,  
 canDelete: false,  
 canApply: false,  
 canUnapply: false  
}

Используя это свойство можно соответствующим образом перестраивать пользовательский интерфейс.

Заметим, что для получения битовой маски можно добавить в таблицу прав (плоскую, разумеется) вычисляемое поле

[Permissions] as cast(CanView as int) +   
 cast(CanEdit as int) \* 2 +   
 cast(CanDelete as int) \* 4 +   
 cast(CanApply as int) \* 8

# Реализация работы с меню

В поставке платформы имеется реализация работы с правами доступа к элементам меню. Поскольку меню иерархическое, механизм “уплощения” использует это обстоятельство.

Если, например, пользователю разрешен доступ к элементу меню, то ему автоматически разрешается доступ к дочерним элементам меню по всему дереву. Таким образом, чтобы разрешить пользователю доступ ко всему меню, достаточно вставить в список контроля доступа одну запись, разрешающую доступ к корню дерева меню.

Процедура a2security.[Permission.UpdateAcl.Module] учитывает иерархию меню и нахождение пользователей в группах и обновляет таблицу a2security.[Menu.Acl].

Процедура загрузки меню просто использует объединение

from a2ui.Menu m   
 inner join a2security.[Menu.Acl] a on m.Id = a.Menu   
where a.UserId = @UserId and a.CanView = 1

# Реализация работы с модулями

Работа с модулями – проще. Они не имеют иерархии. Это означает, что процедура a2security.[Permission.UpdateAcl.Module] просто присваивает флажки исходя из вхождения пользователей в группы и обновляет таблицу a2security.[Module.Acl].