eqweqwe Documentation

Выпуск 1

qweqweqw

25 November 2014

Оглавление

Содержание

Глава 1

Общее описание

Приложение, реализующее управление разрешениями пользователей на выполнение определенныхре-
ализующее операций

Глава 2

Установка

Пакет m3_isers подключается как приложение М3.

2.1 Установка с помощью pip

Установите пакет `m3_users` из репозитория пакетов компании `БАРС Груп`

```
pip install m3-users -i https://<PyPI_сервер_БАРС_Груп>
```

В этом случае будут установлены все необходимые пакеты.

2.2 Установка из архива

Скачайте и распакуйте архив модуля https://bitbucket.org/barsgroup/m3_users/downloads

2.3 Установка из Mercurial

Клонируйте исходный код модуля из репозитория

```
hg clone https://bitbucket.org/barsgroup/m3_users
```

2.4 Установка из каталога

```
python setup.py install
```

2.5 Настройка

Подключение пакета осуществляется в файле `settings.py` приложения. Необходимо добавить имя пакета в раздел `INSTALLED_APPS`.

```
INSTALLED_APPS = (
    m3_users,
)
```

Также необходимо добавить таблицы в СУБД. Если в Вашем проекте используется модуль South запустите команду:

```
python manage.py migrate m3_users
```

В противном случае запустите команду:

```
python manage.py syncdb
```

Глава 3

Использование

После подключения модуля к проекту, необходимо добавить окно редактирования ролей пользователей в меню и в системе появится возможность редактирования прав ролей пользователей.

Например

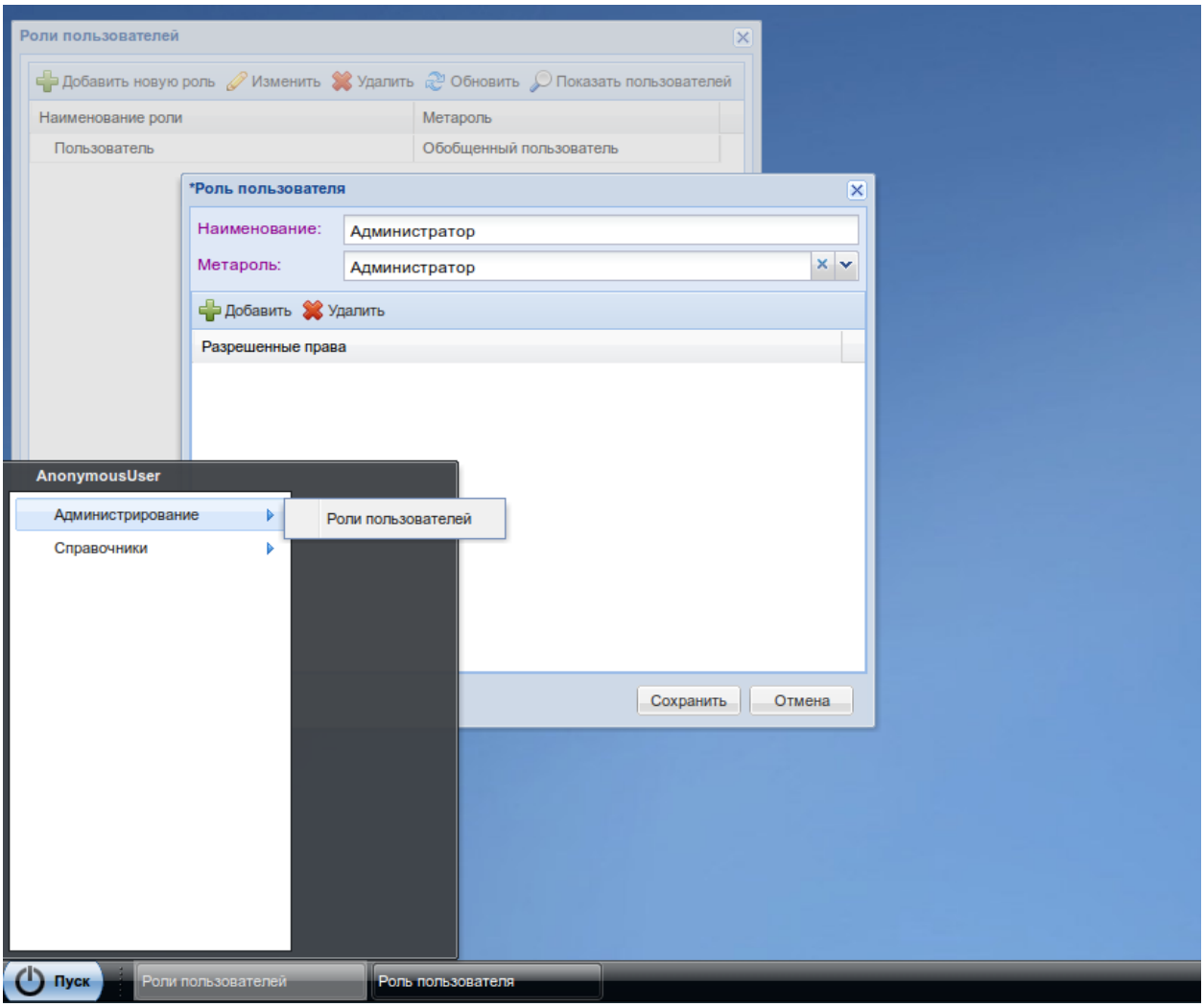
```
# app_meta.py

from m3_users.roles import RolesWindowAction
from m3_ext.ui import app_ui

admin_group = app_ui.DesktopLaunchGroup(name=u'Администрирование')

admin_group.subitems.append(
    app_ui.DesktopLauncher(name=u'Роли пользователей',
                           url=RolesWindowAction.absolute_url())
)

app_ui.DesktopLoader.add(app_ui.get_metarole(app_ui.GENERIC_USER),
                         app_ui.DesktopLoader.START_MENU,
                         admin_group)
```


Роли пользователей
Добавить новую роль
Изменить
Удалить
Обновить
Показать пользователей
Наименование роли
Метароль
Пользователь
Обобщенный пользователь
*Роль пользователя
Наименование:
Администратор
Метароль:
Администратор
Добавить
Удалить
Разрешенные права
Сохранить
Отмена
AnonymousUser
Администрирование
Справочники
Роли пользователей
Пуск
Роли пользователей
Роль пользователя

Состав модуля

4.1 Внешнее API для подсистемы m3_users

m3_users.api.clear_user_roles(*args, **kwargs)
Убирает все роли у пользователя

Параметры **user** (django.contrib.auth.models.User) – пользователь у которого снимается роль

Примечание: если пользователь суперпользователь, то снимается флаг суперпользователя

m3_users.api.get_user_by_id(*user_id*)
Возвращает экземпляр пользователя django.contrib.auth.models.User по указанному идентификатору.

Параметры **user_id** (*int*) – искомый пользователь

Примечание: если вдруг в user_id передан реальный пользователь, то он и возвращается.

m3_users.api.get_user_metaroles(*user*)
Возвращает список объектов метаролей, которые есть у пользователя.

Параметры **user** (django.contrib.auth.models.User) – пользователь системы

m3_users.api.get_user_roles(*user*)
Возвращает список ролей пользователя

Параметры **user** (django.contrib.auth.models.User) – пользователь системы

m3_users.api.remove_user_role(*args, **kwargs)
Снимает роль у пользователя.

Параметры

- **user** (django.contrib.auth.models.User) – пользователь у которого снимается роль
- **role** (m3_users.models.UserRole) – роль

Примечание: если снимаем роль у суперпользователя, и него не осталось ролей, наследовынных от метароли Супер-Администратора, то снимается с него флаг суперпользователя

m3_users.api.set_user_role(*args, **kwargs)
Устанавливает роль для пользователя

Параметры

- **user** (django.contrib.auth.models.User) – пользователь у которого снимается роль
- **role** (m3_users.models.UserRole) – роль

Примечание: если роль суперпользователя, то пользователю ставится флаг суперпользователя

---

m3_users.api.user_has_metarole(*user*, *metarole*)
Возвращает True в случае, если пользователю назначена метароль, иначе False.

Параметры

- **user** (django.contrib.auth.models.User) – пользователь, для которого проверяется наличие метароли
- **metarole** (*str*) – метароль

4.2 Вспомогательне методы модуля

m3_users.helpers.get_assigned_metaroles_query(*user*)
Возвращает список метаролей у пользователя

Параметры **user** (django.contrib.auth.models.User) – пользователь системы

m3_users.helpers.get_assigned_users_query(*role*, *filter=None*)
Возвращает запрос, пользователи имеющие указанную роль

Параметры

- **role** (m3_users.models.UserRole) – роль системы
- **filter** (*str*) – необязательный параметр, строка по которому будет произведена фильтрация пользователей

m3_users.helpers.get_roles_query(*filter=''*)
Возвращает запрос на получение списка ролей

Параметры **filter** (*str*) – необязательный параметр, название конкретной роли

m3_users.helpers.get_unassigned_users(*role*, *filter*)
Хелпер возвращает список пользователей (возможно, отфильтрованных по наименованию), которые еще не включены в роль

m3_users.helpers.get_users_query(*filter=''*)
Возвращает запрос на получение списка пользователей

Параметры **filter** (*str*) – необязательный параметр, строка по которому будет произведена фильтрация пользователей

4.3 Метароли пользователей

На уровне Платформы объявлены следующие метароли:

- Супер-администратор – может создавать других администраторов и назначать им права доступа

- Администратор – выполняет административные функции
- Обобщенный пользователь – любой пользователь системы

class m3_users.metaroles.MetaroleManager
Менеджер метаролей пользователя

get_metarole(*code*)
Возвращает экземпляр метароли по коду

get_registered_metaroles()
Возвращает экземпляры всех зарегистроированных в системе метаролей

register_metarole(*metarole*)
Регистрирует метароль в менеджере ролей

class m3_users.metaroles.Metaroles_DictPack
Пакет действий с метаролями

get_row(*id*)
возвращает метароль по id

get_select_window(*win*)
возвращает окно выбора метароли

Примечание: Доступно 1 событие: выбор с присвоением значения вызвавшему контролу

class m3_users.metaroles.UserMetarole(*metarole_ code*, *metarole_ name*)
Класс, описывающий метароль пользователя

code = **None**
кодовое обозначение метароли пользователя

get_owner_metaroles()
Возвращает список метаролей в которые входит наша метароль. Проще говоря список родителей, ребенком которых является наша метароль.

id
геттер необходим для лучшей маскировки объекта Метароли под объект обычной модели (критично для некоторых операций Dict_Pack'ов)

included_metaroles = **None**
список метаролей, дети данной метароли

name = **None**
название метароли пользователя

class m3_users.models.AssignedRole(**args*, ***kwargs*)
Роль, назначенная на пользователя

role
роль, ссылка m3_users.models.UserRole

user
пользователь, ссылка django.contrib.auth.models.User

class m3_users.models.RolePermission(**args*, ***kwargs*)
Разрешение, сопоставленное пользовательской роли.

disabled = **None**
булево, активность роли

`permission_code` = **None**
строка, код права доступа

`role`
роль, связь с `m3_users.models.UserRole`

`verbose_permission_name` = **None**

текстовое поле, человеческое наименование разрешения с наименованиями модулей, разделенных через запятые.

class `m3_users.models.UserRole`(**args*, ***kwargs*)
Модель хранения роли пользователя в прикладной подсистеме

`metarole` = **None**
строка, ассоциированная с ролью метароль (определяет интерфейс пользователя).

`metarole_name`()
возвращает название метароли

`name` = **None**
строка, наименование роли пользователя

4.4 Права ролей

class `m3_users.permissions.ActionsBackend`
Бэкенд для работы с пользователями

`authenticate`(*username=None*, *password=None*)
Авторизация, возвращает всегда None Мы не будем проверять пользователя - на это есть другие бэкенды

`get_all_permissions`(*user_obj*)
Возвращается набор строк с правами по всем группам, в которые входит пользователь и права непосредственно пользователя В нашем случае это все роли, т.к. у пользователя нет прав :)

Параметры **user_obj** (`django.contrib.auth.models.User`) – пользователь

`get_group_permissions`(*user_obj*)
Возвращается набор строк с правами по всем группам, в которые входит пользователь В нашем случае это роли и права в ролях

Параметры **user_obj** (`django.contrib.auth.models.User`) – пользователь

`get_perm_details`(*user_obj*, *perm*)
Возвращает детали прав доступа по коду, возвращает всегда None

`get_user`(*user_id*)
Возвращает пользователя системы по id

Параметры **user_id** (*int*) – идентификатор пользователя

`has_module_perms`(*user_obj*, *app_label*)
Проверка наличия каких-либо права в указанном приложении/модуле

Параметры

- **user_obj** (`django.contrib.auth.models.User`) – пользователь
- **app_label** (*str*) – название приложения

has_perm(*user_obj*, *perm*, *obj=None*)
Проверка наличия права у пользователя

Параметры

- **user_obj** (django.contrib.auth.models.User) – пользователь
- **perm** (m3_users.models.RolePermission) – право

m3_users.permissions.get_permission_details(*user_obj*, *perm*)
Возвращает параметры права

Параметры

- **user_obj** (django.contrib.auth.models.User) – пользователь
- **perm** (m3_users.models.RolePermission) – право

Created on 11.06.2010

@author: akvarats

class m3_users.roles.AddRolePermission
Выбор прав доступа для добавления в роль

class m3_users.roles.AssignUsers
Сопоставление пользователей роли

class m3_users.roles.AssignedUsersWindow(**args*, ***kwargs*)
Окно просмотра пользователей, которые включены в данную роль

class m3_users.roles.DeleteAssignedUser
Удаление связанного с ролью пользователя

class m3_users.roles.EditRoleWindowAction
Получение окна редактирования роли пользователя

class m3_users.roles.GetRolePermissionAction
Получение списка прав доступа для указанной роли

class m3_users.roles.RoleAssignedUsersDataAction
Получение данных (списка) пользователей, которые прикреплены к указанной роли

class m3_users.roles.RolesActions
Пакет действий для подсистемы прав пользователей

edit_win
псевдоним класса RolesEditWindow

select_win
псевдоним класса SelectUsersListWindow

class m3_users.roles.RolesWindowAction
Действие на получение окна показа списка пользовательских ролей

class m3_users.roles.Roles_DictPack
Справочник “Роли пользователей”.

Используется для выбора значений.

edit_window
псевдоним класса RolesEditWindow

model
псевдоним класса UserRole

class m3_users.roles.SaveRoleAction
Сохранение роли пользователя

class m3_users.roles.SelectPermissionWindow(**args*, ***kwargs*)
Окно выбора прав доступа для добавления в роль

class m3_users.roles.SelectUsersToAssignWindowAction
Показ окна с выбором сотрудников, которые не были выбраны ранее для указанной роли

class m3_users.roles.ShowAssignedUsersAction
Получение окна со списком связанных пользователей

class m3_users.roles.UsersForRoleAssignmentData
Получение списка пользователей, которые могут быть назначены на роль

m3_users.roles.get_all_permission_tree()
Общий подход к формированию дерева прав доступа (алгоритм): собирается список прав доступа исходя из наборов действий, действий, субправ наборов действий, субправ действий у каждого элемента списка прав должен быть путь в дереве (пока строкой), наименование права доступа, полное наименование права (для грида прав) путь в дереве строится из значения path элемента, а если он отсутствует, то по иерархии этого элемента в фактической структуре действий и набора действий

4.5 Паки и действия для работы с пользователями

class m3_users.users.SelectUsersListWindow(**args*, ***kwargs*)
Окно со списком пользователей

class m3_users.users.UsersActions
Пакет действий для пользователей системы

class m3_users.users.UsersDataAction
Получение списка пользователей

Indices and tables

- *genindex*
- *modindex*
- *search*

m

A

C

D

E

G

H

I

M

N

P

R

S

U

V