

# msaSDK Module

## .auth.auth.models

---

### Classes

#### BaseGroup

Bases: BaseRBAC

##### Attributes

\_\_tablename\_\_ class-attribute

```
__tablename__ = 'auth_group'
```

parent\_id class-attribute

```
parent_id: int = Field(  
    None, title=_("Parent"), foreign_key="auth_group.id"  
)
```

#### BaseRBAC

Bases: PkMixin

##### Attributes

\_\_table\_args\_\_ class-attribute

```
__table_args__ = {'extend_existing': True}
```

desc class-attribute

```
desc: str = Field(  
    default="",  
    title=_("Description"),  
    max_length=400,
```

```
msa_ui_form_item="textarea",
)
```

key class-attribute

```
key: str = Field(
    Ellipsis,
    title=_("Identify"),
    max_length=20,
    sa_column=Column(
        String(20), unique=True, index=True, nullable=False
    ),
)
```

name class-attribute

```
name: str = Field(Ellipsis, title=_('Name'), max_length=20)
```

## BaseUser

Bases: PkMixin, UsernameMixin, PasswordMixin, EmailMixin, CreateTimeMixin

### Attributes

\_\_table\_args\_\_ class-attribute

```
__table_args__ = {'extend_existing': True}
```

\_\_tablename\_\_ class-attribute

```
__tablename__ = 'auth_user'
```

avatar class-attribute

```
avatar: str = Field(
    None,
    title=_("Avatar"),
    max_length=100,
    msa_ui_form_item=InputImage(
        maxLength=1,
        maxSize=2 * 1024 * 1024,
        receiver="post:/admin/file/upload",
    ),
    msa_ui_table_column=ColumnImage(
        width=50, height=50, enlargeAble=True
    )
)
```

```
),  
)
```

is\_active class-attribute

```
is_active: bool = Field(default=True, title=_('Is Active'))
```

nickname class-attribute

```
nickname: str = Field(  
    None, title=_("Nickname"), max_length=32  
)
```

**Classes**

**Config**

**Attributes**

use\_enum\_values class-attribute

```
use_enum_values = True
```

**Functions**

display\_name property

```
display_name() -> str
```

**has\_requires**

```
has_requires(  
    session: Session,  
    *,  
    roles: Union[str, Sequence[str]] = None,  
    groups: Union[str, Sequence[str]] = None,  
    permissions: Union[str, Sequence[str]] = None  
) -> bool
```

Check if the user has the specified RBAC privileges

PARAMETER	DESCRIPTION
-----------	-------------

PARAMETER	DESCRIPTION
<code>session</code>	sqlalchemy <code>Session</code> ; asynchronous <code>AsyncSession</code> , please use <code>run_sync</code> method. <b>TYPE:</b> <code>Session</code>
<code>roles</code>	list of roles <b>TYPE:</b> <code>Union[str, Sequence[str]]</code> <b>DEFAULT:</b> <code>None</code>
<code>groups</code>	list of user groups <b>TYPE:</b> <code>Union[str, Sequence[str]]</code> <b>DEFAULT:</b> <code>None</code>
<code>permissions</code>	list of permissions <b>TYPE:</b> <code>Union[str, Sequence[str]]</code> <b>DEFAULT:</b> <code>None</code>

RETURNS	DESCRIPTION
<code>bool</code>	Return <code>True</code> for successful detection.

**identity** `property`

```
identity() -> str
```

**is\_authenticated** `property`

```
is_authenticated() -> bool
```

CreateTimeMixin

Bases: `SQLModel`

Attributes

`create_time` `class-attribute`

```
create_time: datetime = Field(
    default_factory=datetime.now, title=_("Create Time")
)
```

## EmailMixin

Bases: `SQLModel`

### Attributes

email `class-attribute`

```
email: EmailStr = Field(
    None,
    title=_("Email"),
    sa_column=Column(
        String(50), unique=True, index=True, nullable=True
    ),
    msa_ui_form_item="input-email",
)
```

## Group

Bases: `BaseGroup`

Group

### Attributes

roles `class-attribute`

```
roles: List[Role] = Relationship(
    back_populates="groups", link_model=GroupRoleLink
)
```

## GroupRoleLink

Bases: `SQLModel`

### Attributes

\_\_table\_args\_\_ `class-attribute`

```
__table_args__ = {'extend_existing': True}
```

\_\_tablename\_\_ `class-attribute`

```
__tablename__ = 'auth_group_roles'
```

group\_id class-attribute

```
group_id: Optional[int] = Field(
    default=None,
    foreign_key="auth_group.id",
    primary_key=True,
    nullable=False,
)
```

role\_id class-attribute

```
role_id: Optional[int] = Field(
    default=None,
    foreign_key="auth_role.id",
    primary_key=True,
    nullable=False,
)
```

## PasswordMixin

Bases: SQLModel

### Attributes

password class-attribute

```
password: PasswordStr = Field(
    title=_("Password"),
    max_length=128,
    sa_column=Column(String(128), nullable=False),
    msa_ui_form_item="input-password",
)
```

## PasswordStr

Bases: SecretStr, str

## Permission

Bases: BaseRBAC

Permission

### Attributes

\_\_tablename\_\_ class-attribute

```
__tablename__ = 'auth_permission'
```

roles class-attribute

```
roles: List[Role] = Relationship(
    back_populates="permissions",
    link_model=RolePermissionLink,
)
```

## PkMixin

Bases: SQLModel

### Attributes

id class-attribute

```
id: int = Field(
    default=None, primary_key=True, nullable=False
)
```

## Role

Bases: BaseRBAC

Roles

### Attributes

\_\_tablename\_\_ class-attribute

```
__tablename__ = 'auth_role'
```

groups class-attribute

```
groups: List[Group] = Relationship(
    back_populates="roles", link_model=GroupRoleLink
)
```

permissions class-attribute

```
permissions: List[Permission] = Relationship(
    back_populates="roles", link_model=RolePermissionLink
)
```

## RolePermissionLink

Bases: `SQLModel`

### Attributes

`__table_args__` class-attribute

```
__table_args__ = {'extend_existing': True}
```

`__tablename__` class-attribute

```
__tablename__ = 'auth_role_permissions'
```

`permission_id` class-attribute

```
permission_id: Optional[int] = Field(  
    default=None,  
    foreign_key="auth_permission.id",  
    primary_key=True,  
    nullable=False,  
)
```

`role_id` class-attribute

```
role_id: Optional[int] = Field(  
    default=None,  
    foreign_key="auth_role.id",  
    primary_key=True,  
    nullable=False,  
)
```

## User

Bases: `BaseUser`

### Attributes

`__table_args__` class-attribute

```
__table_args__ = {'extend_existing': True}
```

`__tablename__` class-attribute



```
__tablename__ = 'auth_user'
```

groups class-attribute

```
groups: List[Group] = Relationship(link_model=UserGroupLink)
```

roles class-attribute

```
roles: List[Role] = Relationship(link_model=UserRoleLink)
```

## UserGroupLink

Bases: SQLModel

### Attributes

\_\_table\_args\_\_ class-attribute

```
__table_args__ = {'extend_existing': True}
```

\_\_tablename\_\_ class-attribute

```
__tablename__ = 'auth_user_groups'
```

group\_id class-attribute

```
group_id: Optional[int] = Field(
    default=None,
    foreign_key="auth_group.id",
    primary_key=True,
    nullable=False,
)
```

user\_id class-attribute

```
user_id: Optional[int] = Field(
    default=None,
    foreign_key="auth_user.id",
    primary_key=True,
    nullable=False,
)
```

## UserRoleLink

Bases: `SQLModel`

### Attributes

`__table_args__` class-attribute

```
__table_args__ = {'extend_existing': True}
```

`__tablename__` class-attribute

```
__tablename__ = 'auth_user_roles'
```

`role_id` class-attribute

```
role_id: Optional[int] = Field(
    default=None,
    foreign_key="auth_role.id",
    primary_key=True,
    nullable=False,
)
```

`user_id` class-attribute

```
user_id: Optional[int] = Field(
    default=None,
    foreign_key="auth_user.id",
    primary_key=True,
    nullable=False,
)
```

## UsernameMixin

Bases: `SQLModel`

### Attributes

`username` class-attribute

```
username: str = Field(
    title=_("Username"),
    max_length=32,
    sa_column=Column(
        String(32), unique=True, index=True, nullable=False
```

```
)  
)
```

---

Last update: September 13, 2022

Created: September 13, 2022