msaSDK Module

.admin.admin

Attributes

Classes

AdminApp

```
Bases: PageAdmin, AdminGroup
Management Applications
Attributes
__register_lock [instance-attribute]
  __register_lock = False
db_{\,\, {\tt instance-attribute}}
  db = AsyncDatabase(self.engine)
engine [instance-attribute]
  engine = self.engine or self.app.engine
msa_app [instance-attribute]
  msa_app: MSAApp = msa_app
page_path class-attribute
  page_path = '/'
```

```
tabs_mode class-attribute
```

```
tabs_mode: TabsModeEnum = None
```

Functions

__init__

```
__init__(app: AdminApp, msa_app: MSAApp)
```

get_admin_or_create

```
get_admin_or_create(
    admin_cls: Type[_BaseAdminT], register: bool = True
) -> Optional[_BaseAdminT]
```

get_model_admin cached

```
get_model_admin(table_name: str) -> Optional[ModelAdmin]
```

This function returns a cached instance of the ModelAdmin object.



Caching is used to prevent re-reading the environment every time the ModelAdmin object is used.

get_page async

```
get_page(request: Request) -> Union[Page, App]
```

get_page_schema

```
get_page_schema() -> Optional[PageSchema]
```

register_admin

```
register_admin(
    *admin_cls: Type[_BaseAdminT],
) -> Type[_BaseAdminT]
```

register_router

```
register_router()

router_prefix property

router_prefix()

unregister_admin

unregister_admin(*admin_cls: Type[BaseAdmin])
```

AdminGroup

Bases: PageSchemaAdmin

Management Applications Group

Functions

__init__

```
__init__(app: AdminApp) -> None
```

__iter__

```
__iter__() -> Iterator[_PageSchemaAdminT]
```

append_child

```
append_child(
    child: _PageSchemaAdminT,
    group_schema: PageSchema = None,
) -> None
```

get_page_schema_child

```
get_page_schema_child(
    unique_id: str,
) -> Optional[_PageSchemaAdminT]
```

get_page_schema_children async

```
get_page_schema_children(
    request: Request,
) -> List[PageSchema]

unique_id

unique_id() -> str
```

BaseAdmin

Attributes

app instance-attribute

```
app = app
```

Functions

__init__

```
__init__(app: AdminApp)
```

site

```
site() -> BaseAdminSite
```

unique_id

```
unique_id() -> str
```

BaseAdminSite

Bases: AdminApp

Attributes

auth class-attribute

```
auth: Auth = None
```

 $db_engine {\tiny \underbrace{\tt instance-attribute}}$

```
db_engine = self.msa_app.sqlite_db_engine
msa_app [instance-attribute]
  msa_app = msa_app
router [instance-attribute]
  router = self.msa_app.router
settings instance-attribute
  settings = msa_app.settings
Functions
__init__
  __init__(msa_app: MSAApp)
mount_app
  mount_app(msa_app: MSAApp, name: str = 'admin') -> None
router_path
  router_path() -> str
```

BaseFormAdmin

Bases: PageAdmin

Attributes

form class-attribute

form: Form = None

form_init class-attribute

```
form_init: bool = None
form_path [instance-attribute]
  form_path = self.form_path or f'{self.page_path}/api'
route_init class-attribute
  route_init: Callable = None
route_submit class-attribute
  route_submit: Callable = None
router_prefix class-attribute
  router_prefix: str = '/form'
schema class-attribute
  schema: Type[MSABaseModel] = None
schema_init_out class-attribute
  schema_init_out: Type[Any] = Any
schema_submit_out class-attribute
  schema_submit_out: Type[Any] = Any
Functions
__init__
  __init__(app: AdminApp)
get_form async
  get_form(request: Request) -> Form
```

```
get_form_item async
      get_form_item(
         request: Request, modelfield: ModelField
      ) -> Union[FormItem, MSAUISchemaNode]
    get_page async
      get_page(request: Request) -> Page
    register_router
      register_router()
BaseModelAction
    Base Model Action
    Attributes
    action class-attribute
      action: Action = None
    admin instance-attribute
      admin = admin
    Functions
    __init__
      __init__(admin: ModelAdmin)
    fetch_item_scalars async
      fetch_item_scalars(
         item_id: List[str],
      ) -> List[MSABaseModel]
    register_router
```

```
register_router()
```

BaseModelAdmin

```
Bases: MSASQLModelCrud
Attributes
app [instance-attribute]
  app = app
bulk_update_fields class-attribute
  bulk_update_fields: List[
      Union[SQLModelListField, FormItem]
  ] = []
    Batch Edit Fields
engine instance-attribute
  engine = self.engine or self.app.db.engine
fields [instance-attribute]
  fields = self.fields or [self.model]
link_model_fields class-attribute
  link_model_fields: List[InstrumentedAttribute] = []
    Inline Fields
link_model_forms class-attribute
  link_model_forms: List[LinkModelForm] = []
    Inline Forms
list_display class-attribute
```

```
list_display: List[
      Union[SQLModelListField, TableColumn]
  ] = []
    Fields to be displayed
list_filter instance-attribute
  list_filter = self.list_filter or list_display_insfield
list_per_page class-attribute
  list_per_page: int = 10
    Data volume per page
model [instance-attribute]
  model = model
parser instance-attribute
  parser = MSASQLModelFieldParser(default_model=self.model)
search_fields class-attribute
  search_fields: List[SQLModelField] = []
    Fuzzy search fields
Functions
__init__
  __init__(app: AdminApp, model = None)
get_actions_on_bulk async
  get_actions_on_bulk(request: Request) -> List[Action]
get_actions_on_header_toolbar async
```

```
get_actions_on_header_toolbar(
      request: Request,
  ) -> List[Action]
get_actions_on_item async
  get_actions_on_item(request: Request) -> List[Action]
get_create_action [async]
  get_create_action(
     request: Request, bulk: bool = False
  ) -> Optional[Action]
get_create_form [async]
  get_create_form(
     request: Request, bulk: bool = False
  ) -> Form
get_delete_action [async]
  get_delete_action(
      request: Request, bulk: bool = False
  ) -> Optional[Action]
get_form_item async
  get_form_item(
      request: Request,
      modelfield: ModelField,
      action: MSACRUDEnum,
  ) -> Union[FormItem, MSAUISchemaNode, None]
get_form_item_on_foreign_key async
  get_form_item_on_foreign_key(
     request: Request,
      modelfield: ModelField,
      is_filter: bool = False,
  ) -> Union[Service, MSAUISchemaNode, None]
get_link_model_forms
```

Page: 10 of 22

```
get_link_model_forms() -> List[LinkModelForm]
get_list_column async
  get_list_column(
     request: Request, modelfield: ModelField
  ) -> TableColumn
get_list_columns (async)
  get_list_columns(request: Request) -> List[TableColumn]
get_list_display async
  get_list_display(
      request: Request,
  ) -> List[Union[SQLModelListField, TableColumn]]
get_list_filter async
  get_list_filter(
     request: Request,
  ) -> List[Union[SQLModelListField, FormItem]]
get_list_filter_form [async]
  get_list_filter_form(request: Request) -> Form
get_list_table async
  get_list_table(request: Request) -> TableCRUD
get_list_table_api async
  get_list_table_api(request: Request) -> MSAUIAPI
get_update_action async
 get_update_action(
      request: Request, bulk: bool = False
  ) -> Optional[Action]
```

```
get_update_form(
    request: Request, bulk: bool = False
) -> Form

router_path

router_path() -> str
```

FormAdmin

IframeAdmin

Bases: PageSchemaAdmin

Attributes

```
iframe class-attribute

iframe: Iframe = None

SrC class-attribute

src: str = ''

Functions

get_page_schema

get_page_schema() -> Optional[PageSchema]
```

LinkAdmin

Bases: PageSchemaAdmin

Management Links

Attributes

link class-attribute

```
link: str = ''
```

Functions

get_page_schema

```
get_page_schema() -> Optional[PageSchema]
```

LinkModelForm

Link Model to Form

Attributes

display_admin instance-attribute

```
display_admin = display_admin
```

Page: 13 of 22

```
item_col [instance-attribute]
  item_col = item_col
link_col [instance-attribute]
  link_col = link_col
link_model instance-attribute
  link_model = link_model
path [instance-attribute]
  path = f'/{self.display_admin.model.__name__.lower()}'
pk\_admin {\tiny \underbrace{ \text{instance-attribute} } }
  pk_admin = pk_admin
Functions
__init__
  __init__(
      pk_admin: BaseModelAdmin,
      display_admin: ModelAdmin,
      link_model: Union[SQLModel, Table],
      link_col: Column,
      item_col: Column,
bind_model_admin classmethod
  bind_model_admin(
      pk_admin: BaseModelAdmin,
      insfield: InstrumentedAttribute,
  ) -> Optional[LinkModelForm]
get_form_item async
  get_form_item(request: Request)
```

```
register_router()

route_create property

route_create()

route_delete property

route_delete()

ModelAction

Bases: BaseFormAdmin, BaseModelAction
Form and Model Actions

Attributes
```

action class-attribute

action: ActionType.Dialog = None

router [instance-attribute]

router = self.admin.router

schema class-attribute

schema: Type[MSABaseModel] = None

Functions

__init__

```
__init__(admin: ModelAdmin)
```

get_action async

```
get_action(request: Request, **kwargs) -> Action
```

```
handle async

handle(
    request: Request,
    item_id: List[str],
    data: Optional[MSABaseModel],
    **kwargs
) -> MSACRUDOut[Any]

route_submit property

route_submit()
```

ModelAdmin

```
Bases: BaseModelAdmin, PageAdmin
Model Management
Attributes
bind_model class-attribute
  bind_model: bool = True
page_path class-attribute
  page_path: str = ''
Functions
__init__
  __init__(app: AdminApp, model = None)
get_page async
  get_page(request: Request) -> Page
has_create_permission (async)
  has_create_permission(
      request: Request, data: MSABaseModel, **kwargs
```

```
) -> bool
```

has_delete_permission async

```
has_delete_permission(
    request: Request, item_id: List[str], **kwargs
) -> bool
```

has_list_permission async

```
has_list_permission(
    request: Request,
    paginator: MSACRUDPaginator,
    filters: MSABaseModel = None,
    **kwargs
) -> bool
```

has_read_permission [async]

```
has_read_permission(
    request: Request, item_id: List[str], **kwargs
) -> bool
```

has_update_permission async

```
has_update_permission(
    request: Request,
    item_id: List[str],
    data: MSABaseModel,
    **kwargs
) -> bool
```

register_router

```
register_router()
```

router_prefix property

```
router_prefix()
```

ModelFormAdmin

Bases: FormAdmin, MSASQLModelSelector, ABC

todo Read and update a model resource

Functions

```
__init__
```

```
__init__(app: AdminApp)
```

PageAdmin

Functions

```
Bases: PageSchemaAdmin, RouterAdmin
msa_ui page management
Attributes
page class-attribute
  page: Page = None
page_parser_mode class-attribute
  page_parser_mode: Literal[json, html] = 'json'
page_path class-attribute
  page_path: Optional[str] = None
page_route_kwargs class-attribute
  page_route_kwargs: Dict[str, Any] = {}
router_prefix class-attribute
  router_prefix = '/page'
template_name class-attribute
  template_name: str = ''
```

```
__init__
  __init__(app: AdminApp)
error_no_page_permission
  error_no_page_permission(request: Request)
get_page async
 get_page(request: Request) -> Page
get_page_schema
  get_page_schema() -> Optional[PageSchema]
page_parser async
  page_parser(request: Request, page: Page) -> Response
page_permission_depend async
  page_permission_depend(request: Request) -> bool
register_router
  register_router()
route_page property
  route_page() -> Callable
```

PageSchemaAdmin

Bases: BaseAdmin

Attributes

group_schema instance-attribute

```
group_schema = self.get_group_schema()
```

page_schema [instance-attribute]

```
page_schema = self.get_page_schema()
```

Functions

__init__

```
__init__(app: AdminApp)
```

get_group_schema

```
get_group_schema() -> Optional[PageSchema]
```

RAISES	DESCRIPTION
TypeError	TypeError(_StandardError)

get_page_schema

```
get_page_schema() -> Optional[PageSchema]
```

RAISES	DESCRIPTION
TypeError	TypeError(_StandardError)

has_page_permission async

```
has_page_permission(request: Request) -> bool
```

RouterAdmin

Bases: BaseAdmin, MSARouterMixin

Management Router

Functions

```
__init__
      __init__(app: AdminApp)
   register_router
      register_router()
   router_path
      router_path() -> str
TemplateAdmin
   Bases: PageAdmin
   Jinja2-Rendering Template Management
   Attributes
   page class-attribute
     page: Dict[str, Any] = {}
   page_parser_mode class-attribute
     page_parser_mode = 'html'
   page_path [instance-attribute]
     page_path = self.page_path or f'/{self.template_name}'
   templates class-attribute
      templates: Jinja2Templates = None
   Functions
    __init__
      __init__(app: AdminApp)
```

```
get_page async

get_page(request: Request) -> Dict[str, Any]

page_parser async

page_parser(
    request: Request, page: Dict[str, Any]
) -> Response
```

Last update: September 20, 2022 Created: September 20, 2022