Flask-DataTables

Release 0.1.2

TEKID Ltd.

Oct 04, 2023

CONTENTS

CHAPTER

ONE

DATABASE CONNECTION

We made some monkeypatch for `peewee` and `playhouse.flask_utils` in order to better support the integration with `Flask`.

class flask_datatables.database.**DataTable**(*app=None*, *database=None*, *model_class=None*)

Bases: `FlaskDB`

Server-side processing integration with DataTables.

Parameters

- **app** (Optional[Flask]) – `flask.Flask` instance to integrate DataTables.
- **database** (Union[Dict[str, Any], str, FlaskDB, None]) – Database connection configurations.
- **model_class** (Optional[Type[*Model*]]) – *flask_datatables.model.Model* class to create data models.

init_app(*app*)

Initialise the `DataTables` with the `Flask` application.

Parameters
app (Flask) – `flask.Flask` instance to integrate DataTables.

Return type
None

base_model_class: *Model*

Base class for data models.

database: **FlaskDB**

Database connection instance.

CHAPTER

TWO

BASE MODEL

We extends the `peewee.Model` class to integrate with the server-side processing logic of DataTables. Some monkey-pathes were made as certain functionalities with `peewee` are not correctly implemented.

class flask_datatables.model.**Model**(**args*, ***kwargs*)

Bases: `Model`

Extends `peewee.Model` with DataTables support.

To ease the *`Model`* class from recursively processing its field data, we added the following two attributes to save the DataTables integrated fields data.

dt_orderable: Dict[str, Field] = {}

DataTables orderable fields.

dt_searchable: Dict[str, Field] = {}

DataTables searchable fields.

classmethod validate_model()

Validates data model and dynamically insert fields.

If DataTables integration is enabled for the data model, this method will insert fields (database columns) for both *order* and *search* operations respectively on each defined fields according to the original field type definition.

By default, each field is *orderable* and/or *searchable* as long as the *`datatables`* switch is enabled. When the *`orderable`* and/or *`searchable`* attributes are set to an instance of a `Field`, `Flask-DataTables` will insert additional fields of such type with `_dt_order` and/or `_dt_search` suffix as the field names accordingly.

Return type
`None`

save(*force_insert=False*, *only=None*)

Save the data in the model instance.

The method extends the original `peewee.Model.save()` method by automatically update the *searching* and *ordering* field data with the actual data.

Parameters

- **force_insert** (`bool`) – Force `INSERT` query.
- **only** (`Optional[List[`*`Field`*`]]`) – Only save the given `Field` instances.

Return type
`int`

Returns
Number of rows modified.

classmethod search(*query=None*, *factory=None*)

Server-side processing integration with DataTables.

Parameters

- **query** (Optional[*Query*]) – Query parameters sent from the client-side.
- **factory** (Optional[Callable[[Model], Union[List[Any], *ObjectData*]]]) – Factory function to prepare the server-side data.

Return type
Response

Returns
Selected information from the database in format to be sent to DataTables.

See also:

The `factory` function takes exactly one parameter, the data record returned from `peewee` selection, and returns the converted data of fields. See *flask_datatables.utils.prepare_response()* for an example.

class flask_datatables.model.**Metadata**(*model*, *database=None*, *table_name=None*, *indexes=None*, *primary_key=None*, *constraints=None*, *schema=None*, *only_save_dirty=False*, *depends_on=None*, *options=None*, *db_table=None*, *table_function=None*, *table_settings=None*, *without_rowid=False*, *temporary=False*, *strict_tables=None*, *legacy_table_names=True*, ***kwargs*)

Bases: `Metadata`

Basic metadata for data models.

`Flask-DataTables` extends the original metadata record from `peewee` with a *datatables* switch to indicate if current data model supports and/or enables DataTables server-side processing integration.

datatables: bool = False

DataTables integration indicator flag.

CHAPTER

THREE

DATA FIELDS

We extends the field classes from peewee with two properties (*dt_orderable* and *dt_searchable*) and two conversion methods (*dt_order()* and *dt_search()*) for the integration with DataTables server-side processing.

class flask_datatables.fields.**Field**(*orderable=None*, *searchable=None*, *null=False*, *index=False*, *unique=False*, *column_name=None*, *default=None*, *primary_key=False*, *constraints=None*, *sequence=None*, *collation=None*, *unindexed=False*, *choices=None*, *help_text=None*, *verbose_name=None*, *index_type=None*, *db_column=None*, *_hidden=False*)

Bases: Field

Extending peewee.Field.

Parameters

- **orderable** (Union[bool, *Field*, None]) – Optional[Union[bool, peewee.Field]]: DataTables orderable field.
- **searchable** (Union[bool, *Field*, None]) – Optional[Union[bool, peewee.Field]]: DataTables searchable field.
- ****kwargs** – Arbitrary arguments accepted by peewee.Field.
- **null** (*bool*) –
- **index** (*bool*) –
- **unique** (*bool*) –
- **column_name** (*Optional[str]*) –
- **default** (*Any*) –
- **primary_key** (*bool*) –
- **constraints** (*Optional[List[SQL]]*) –
- **sequence** (*Optional[str]*) –
- **collation** (*Optional[str]*) –
- **unindexed** (*bool*) –
- **choices** (*Optional[Iterable[Tuple[str, Any]]]*) –
- **help_text** (*Optional[str]*) –
- **verbose_name** (*Optional[str]*) –
- **index_type** (*Optional[str]*) –

- **db_column** (*Optional[str]*) –
- **_hidden** (*bool*) –

If `orderable` and/or `searchable` is a `bool` value, it indicates if the field supports DataTables ordering and/or searching:

- `True` means the field is orderable and/or searchable, and it will refer to its properties *dt_orderable* and/or *dt_searchable* as its default field instance;

  Note: If the property returns `None`, then the field is orderable and/or searchable by itself with its own values.

- `False` disables ordering and searching on the field;
- an instance of `peewee.Field` indicates that the current field is orderable and/or searchable through the given field instance.

Important: If `orderable` and/or `searchable` is an instance of `peewee.Field`, then its attributes *orderable* and/or *searchable* will be the corresponding instance.

If `orderable` and/or `searchable` is `True`, then it refers to its properties *dt_orderable* and/or *dt_searchable* as the actual value:

- if the properties return an instance of `peewee.Field`, then the attributes will be the returned instance; i.e. the field is orderable and/or searchable by converting to the target field instead of itself;
- if the properties return `None`, then the attributes will be `True`; i.e. the field is orderable and/or searchable by itself with its value.

If `orderable` and/or `searchable` is `False`, then the attributes will be `False` as well.

static dt_order(*value*)

Convert value for DataTables ordering operation.

Parameters
value (Any) – Source value.

Return type
Any

Returns
Converted value.

static dt_search(*value*)

Convert value for DataTables searching operation.

Parameters
value (Any) – Source value.

Return type
Any

Returns
Converted value.

property dt_orderable: *Field* **| None**

DataTables default orderable field.

property dt_searchable: *Field* | None

DataTables default searchable field.

orderable: Union[bool, *Field*]

DataTables integration orderable flag.

searchable: Union[bool, *Field*]

DataTables integration searchable flag.

class flask_datatables.fields.**AutoField**(**args*, ***kwargs*)

Bases: AutoField, *IntegerField*

Extending peewee.AutoField.

orderable: Union[bool, *Field*]

DataTables integration orderable flag.

searchable: Union[bool, *Field*]

DataTables integration searchable flag.

class flask_datatables.fields.**BareField**(*adapt=None*, **args*, ***kwargs*)

Bases: BareField, *Field*

Extending peewee.BareField.

orderable: Union[bool, *Field*]

DataTables integration orderable flag.

searchable: Union[bool, *Field*]

DataTables integration searchable flag.

class flask_datatables.fields.**BigAutoField**(**args*, ***kwargs*)

Bases: BigAutoField, *AutoField*

Extending peewee.BigAutoField.

orderable: Union[bool, *Field*]

DataTables integration orderable flag.

searchable: Union[bool, *Field*]

DataTables integration searchable flag.

class flask_datatables.fields.**BigBitField**(**args*, ***kwargs*)

Bases: BigBitField, *BlobField*

Extending peewee.BigBitField.

orderable: Union[bool, *Field*]

DataTables integration orderable flag.

searchable: Union[bool, *Field*]

DataTables integration searchable flag.

class flask_datatables.fields.**BigIntegerField**(*orderable=None*, *searchable=None*, *null=False*, *index=False*, *unique=False*, *column_name=None*, *default=None*, *primary_key=False*, *constraints=None*, *sequence=None*, *collation=None*, *unindexed=False*, *choices=None*, *help_text=None*, *verbose_name=None*, *index_type=None*, *db_column=None*, *_hidden=False*)

Bases: BigIntegerField, *IntegerField*

Extending peewee.BigIntegerField.

Parameters

- **orderable** (*bool | Field*) –
- **searchable** (*bool | Field*) –
- **null** (*bool*) –
- **index** (*bool*) –
- **unique** (*bool*) –
- **column_name** (*Optional[str]*) –
- **default** (*Any*) –
- **primary_key** (*bool*) –
- **constraints** (*Optional[List[SQL]]*) –
- **sequence** (*Optional[str]*) –
- **collation** (*Optional[str]*) –
- **unindexed** (*bool*) –
- **choices** (*Optional[Iterable[Tuple[str, Any]]]*) –
- **help_text** (*Optional[str]*) –
- **verbose_name** (*Optional[str]*) –
- **index_type** (*Optional[str]*) –
- **db_column** (*Optional[str]*) –
- **_hidden** (*bool*) –

orderable: Union[bool, *Field*]

DataTables integration orderable flag.

searchable: Union[bool, *Field*]

DataTables integration searchable flag.

class flask_datatables.fields.**BinaryUUIDField**(*orderable=None*, *searchable=None*, *null=False*, *index=False*, *unique=False*, *column_name=None*, *default=None*, *primary_key=False*, *constraints=None*, *sequence=None*, *collation=None*, *unindexed=False*, *choices=None*, *help_text=None*, *verbose_name=None*, *index_type=None*, *db_column=None*, *_hidden=False*)

Bases: BinaryUUIDField, *Field*

Extending peewee.BinaryUUIDField.

Parameters

- **orderable** (*bool | Field*) –
- **searchable** (*bool | Field*) –
- **null** (*bool*) –
- **index** (*bool*) –

- **unique** (*bool*) –
- **column_name** (*Optional[str]*) –
- **default** (*Any*) –
- **primary_key** (*bool*) –
- **constraints** (*Optional[List[SQL]]*) –
- **sequence** (*Optional[str]*) –
- **collation** (*Optional[str]*) –
- **unindexed** (*bool*) –
- **choices** (*Optional[Iterable[Tuple[str, Any]]]*) –
- **help_text** (*Optional[str]*) –
- **verbose_name** (*Optional[str]*) –
- **index_type** (*Optional[str]*) –
- **db_column** (*Optional[str]*) –
- **_hidden** (*bool*) –

static dt_search(*value*)

Convert value for DataTables searching operation.

Parameters
value (UUID) – Source value.

Return type
str

Returns
Converted value.

property dt_searchable: *Field* | None

DataTables default searchable field.

orderable: Union[bool, *Field*]

DataTables integration orderable flag.

searchable: Union[bool, *Field*]

DataTables integration searchable flag.

class flask_datatables.fields.**BitField**(**args*, ***kwargs*)

Bases: BitField, *BigIntegerField*

Extending peewee.BitField.

orderable: Union[bool, *Field*]

DataTables integration orderable flag.

searchable: Union[bool, *Field*]

DataTables integration searchable flag.

class flask_datatables.fields.**BlobField**(*orderable=None*, *searchable=None*, *null=False*, *index=False*, *unique=False*, *column_name=None*, *default=None*, *primary_key=False*, *constraints=None*, *sequence=None*, *collation=None*, *unindexed=False*, *choices=None*, *help_text=None*, *verbose_name=None*, *index_type=None*, *db_column=None*, *_hidden=False*)

Bases: BlobField, *Field*

Extending peewee.BlobField.

Parameters

- **orderable** (*bool | Field*) –
- **searchable** (*bool | Field*) –
- **null** (*bool*) –
- **index** (*bool*) –
- **unique** (*bool*) –
- **column_name** (*Optional[str]*) –
- **default** (*Any*) –
- **primary_key** (*bool*) –
- **constraints** (*Optional[List[SQL]]*) –
- **sequence** (*Optional[str]*) –
- **collation** (*Optional[str]*) –
- **unindexed** (*bool*) –
- **choices** (*Optional[Iterable[Tuple[str, Any]]]*) –
- **help_text** (*Optional[str]*) –
- **verbose_name** (*Optional[str]*) –
- **index_type** (*Optional[str]*) –
- **db_column** (*Optional[str]*) –
- **_hidden** (*bool*) –

orderable: Union[bool, *Field*]

DataTables integration orderable flag.

searchable: Union[bool, *Field*]

DataTables integration searchable flag.

class flask_datatables.fields.**BooleanField**(*orderable=None*, *searchable=None*, *null=False*, *index=False*, *unique=False*, *column_name=None*, *default=None*, *primary_key=False*, *constraints=None*, *sequence=None*, *collation=None*, *unindexed=False*, *choices=None*, *help_text=None*, *verbose_name=None*, *index_type=None*, *db_column=None*, *_hidden=False*)

Bases: BooleanField, *Field*

Extending peewee.BooleanField.

Parameters

- **orderable** (*bool | Field*) –
- **searchable** (*bool | Field*) –
- **null** (*bool*) –
- **index** (*bool*) –

- **unique** (*bool*) –
- **column_name** (*Optional[str]*) –
- **default** (*Any*) –
- **primary_key** (*bool*) –
- **constraints** (*Optional[List[SQL]]*) –
- **sequence** (*Optional[str]*) –
- **collation** (*Optional[str]*) –
- **unindexed** (*bool*) –
- **choices** (*Optional[Iterable[Tuple[str, Any]]]*) –
- **help_text** (*Optional[str]*) –
- **verbose_name** (*Optional[str]*) –
- **index_type** (*Optional[str]*) –
- **db_column** (*Optional[str]*) –
- **_hidden** (*bool*) –

static dt_search(*value*)

Convert value for DataTables searching operation.

Parameters
value (bool) – Source value.

Return type
str

Returns
Converted value.

property dt_searchable: *Field* | None

DataTables default searchable field.

orderable: Union[bool, *Field*]

DataTables integration orderable flag.

searchable: Union[bool, *Field*]

DataTables integration searchable flag.

class flask_datatables.fields.**CharField**(*max_length=255*, **args*, ***kwargs*)

Bases: CharField, _StringField

Extending peewee.CharField.

orderable: Union[bool, *Field*]

DataTables integration orderable flag.

searchable: Union[bool, *Field*]

DataTables integration searchable flag.

class flask_datatables.fields.**DateField**(*formats=None*, **args*, ***kwargs*)

Bases: DateField, _BaseFormattedField

Extending peewee.DateField.

orderable: Union[bool, *Field*]

DataTables integration orderable flag.

searchable: Union[bool, *Field*]

DataTables integration searchable flag.

class flask_datatables.fields.**DateTimeField**(*formats=None*, **args*, ***kwargs*)

Bases: DateTimeField, _BaseFormattedField

Extending peewee.DateTimeField.

orderable: Union[bool, *Field*]

DataTables integration orderable flag.

searchable: Union[bool, *Field*]

DataTables integration searchable flag.

class flask_datatables.fields.**DecimalField**(*max_digits=10*, *decimal_places=5*, *auto_round=False*, *rounding=None*, **args*, ***kwargs*)

Bases: DecimalField, *Field*

Extending peewee.DecimalField.

static **dt_search**(*value*)

Convert value for DataTables searching operation.

Parameters
value (Decimal) – Source value.

Return type
str

Returns
Converted value.

property **dt_searchable:** *Field* | None

DataTables default searchable field.

orderable: Union[bool, *Field*]

DataTables integration orderable flag.

searchable: Union[bool, *Field*]

DataTables integration searchable flag.

class flask_datatables.fields.**DoubleField**(*orderable=None*, *searchable=None*, *null=False*, *index=False*, *unique=False*, *column_name=None*, *default=None*, *primary_key=False*, *constraints=None*, *sequence=None*, *collation=None*, *unindexed=False*, *choices=None*, *help_text=None*, *verbose_name=None*, *index_type=None*, *db_column=None*, *_hidden=False*)

Bases: DoubleField, *FloatField*

Extending peewee.DoubleField.

Parameters

- **orderable** (*bool* | Field) –
- **searchable** (*bool* | Field) –
- **null** (*bool*) –

- **index** (*bool*) –
- **unique** (*bool*) –
- **column_name** (*Optional[str]*) –
- **default** (*Any*) –
- **primary_key** (*bool*) –
- **constraints** (*Optional[List[SQL]]*) –
- **sequence** (*Optional[str]*) –
- **collation** (*Optional[str]*) –
- **unindexed** (*bool*) –
- **choices** (*Optional[Iterable[Tuple[str, Any]]]*) –
- **help_text** (*Optional[str]*) –
- **verbose_name** (*Optional[str]*) –
- **index_type** (*Optional[str]*) –
- **db_column** (*Optional[str]*) –
- **_hidden** (*bool*) –

orderable: Union[bool, *Field*]

DataTables integration orderable flag.

searchable: Union[bool, *Field*]

DataTables integration searchable flag.

class flask_datatables.fields.**FixedCharField**(*max_length=255*, **args*, ***kwargs*)

Bases: FixedCharField, *CharField*

Extending peewee.FixedCharField.

orderable: Union[bool, *Field*]

DataTables integration orderable flag.

searchable: Union[bool, *Field*]

DataTables integration searchable flag.

class flask_datatables.fields.**FloatField**(*orderable=None*, *searchable=None*, *null=False*, *index=False*, *unique=False*, *column_name=None*, *default=None*, *primary_key=False*, *constraints=None*, *sequence=None*, *collation=None*, *unindexed=False*, *choices=None*, *help_text=None*, *verbose_name=None*, *index_type=None*, *db_column=None*, *_hidden=False*)

Bases: FloatField, *Field*

Extending peewee.FloatField.

Parameters

- **orderable** (*bool | Field*) –
- **searchable** (*bool | Field*) –
- **null** (*bool*) –
- **index** (*bool*) –

- **unique** (*bool*) –
- **column_name** (*Optional[str]*) –
- **default** (*Any*) –
- **primary_key** (*bool*) –
- **constraints** (*Optional[List[SQL]]*) –
- **sequence** (*Optional[str]*) –
- **collation** (*Optional[str]*) –
- **unindexed** (*bool*) –
- **choices** (*Optional[Iterable[Tuple[str, Any]]]*) –
- **help_text** (*Optional[str]*) –
- **verbose_name** (*Optional[str]*) –
- **index_type** (*Optional[str]*) –
- **db_column** (*Optional[str]*) –
- **_hidden** (*bool*) –

static dt_search(*value*)

Convert value for DataTables searching operation.

Parameters
value (float) – Source value.

Return type
str

Returns
Converted value.

property dt_searchable: *Field* | None

DataTables default searchable field.

orderable: Union[bool, *Field*]

DataTables integration orderable flag.

searchable: Union[bool, *Field*]

DataTables integration searchable flag.

class flask_datatables.fields.**ForeignKeyField**(*model*, *field=None*, *backref=None*, *on_delete=None*, *on_update=None*, *deferrable=None*, *_deferred=None*, *rel_model=None*, *to_field=None*, *object_id_name=None*, *lazy_load=True*, *constraint_name=None*, *related_name=None*, **args*, ***kwargs*)

Bases: ForeignKeyField, *Field*

Extending peewee.ForeignKeyField.

orderable: Union[bool, *Field*]

DataTables integration orderable flag.

searchable: Union[bool, *Field*]

DataTables integration searchable flag.

class flask_datatables.fields.**IdentityField**(*generate_always=False*, ***kwargs*)

Bases: IdentityField, *AutoField*

Extending peewee.IdentityField.

orderable: Union[bool, *Field*]

DataTables integration orderable flag.

searchable: Union[bool, *Field*]

DataTables integration searchable flag.

class flask_datatables.fields.**IntegerField**(*orderable=None*, *searchable=None*, *null=False*, *index=False*, *unique=False*, *column_name=None*, *default=None*, *primary_key=False*, *constraints=None*, *sequence=None*, *collation=None*, *unindexed=False*, *choices=None*, *help_text=None*, *verbose_name=None*, *index_type=None*, *db_column=None*, *_hidden=False*)

Bases: IntegerField, *Field*

Extending peewee.IntegerField.

Parameters

- **orderable** (*bool | Field*) –
- **searchable** (*bool | Field*) –
- **null** (*bool*) –
- **index** (*bool*) –
- **unique** (*bool*) –
- **column_name** (*Optional[str]*) –
- **default** (*Any*) –
- **primary_key** (*bool*) –
- **constraints** (*Optional[List[SQL]]*) –
- **sequence** (*Optional[str]*) –
- **collation** (*Optional[str]*) –
- **unindexed** (*bool*) –
- **choices** (*Optional[Iterable[Tuple[str, Any]]]*) –
- **help_text** (*Optional[str]*) –
- **verbose_name** (*Optional[str]*) –
- **index_type** (*Optional[str]*) –
- **db_column** (*Optional[str]*) –
- **_hidden** (*bool*) –

static dt_search(*value*)

Convert value for DataTables searching operation.

Parameters

value (int) – Source value.

Return type
str

Returns
Converted value.

property dt_searchable: *Field* | None

DataTables default searchable field.

orderable: Union[bool, *Field*]

DataTables integration orderable flag.

searchable: Union[bool, *Field*]

DataTables integration searchable flag.

class flask_datatables.fields.**IPField**(*orderable=None*, *searchable=None*, *null=False*, *index=False*, *unique=False*, *column_name=None*, *default=None*, *primary_key=False*, *constraints=None*, *sequence=None*, *collation=None*, *unindexed=False*, *choices=None*, *help_text=None*, *verbose_name=None*, *index_type=None*, *db_column=None*, *_hidden=False*)

Bases: IPField, *BigIntegerField*

Extending peewee.IPField.

Parameters

- **orderable** (*bool | Field*) –
- **searchable** (*bool | Field*) –
- **null** (*bool*) –
- **index** (*bool*) –
- **unique** (*bool*) –
- **column_name** (*Optional[str]*) –
- **default** (*Any*) –
- **primary_key** (*bool*) –
- **constraints** (*Optional[List[SQL]]*) –
- **sequence** (*Optional[str]*) –
- **collation** (*Optional[str]*) –
- **unindexed** (*bool*) –
- **choices** (*Optional[Iterable[Tuple[str, Any]]]*) –
- **help_text** (*Optional[str]*) –
- **verbose_name** (*Optional[str]*) –
- **index_type** (*Optional[str]*) –
- **db_column** (*Optional[str]*) –
- **_hidden** (*bool*) –

static dt_search(*value*)

Convert value for DataTables searching operation.

Parameters
value (str) – Source value.

Return type
str

Returns
Converted value.

orderable: Union[bool, *Field*]

DataTables integration orderable flag.

searchable: Union[bool, *Field*]

DataTables integration searchable flag.

class flask_datatables.fields.**ManyToManyField**(*model*, *backref=None*, *through_model=None*, *on_delete=None*, *on_update=None*, *prevent_unsaved=True*, *_is_backref=False*)

Bases: ManyToManyField, *Field*

Extending peewee.ManyToManyField.

orderable: Union[bool, *Field*]

DataTables integration orderable flag.

searchable: Union[bool, *Field*]

DataTables integration searchable flag.

class flask_datatables.fields.**PrimaryKeyField**(**args*, ***kwargs*)

Bases: PrimaryKeyField, *AutoField*

Extending peewee.PrimaryKeyField.

orderable: Union[bool, *Field*]

DataTables integration orderable flag.

searchable: Union[bool, *Field*]

DataTables integration searchable flag.

class flask_datatables.fields.**SmallIntegerField**(*orderable=None*, *searchable=None*, *null=False*, *index=False*, *unique=False*, *column_name=None*, *default=None*, *primary_key=False*, *constraints=None*, *sequence=None*, *collation=None*, *unindexed=False*, *choices=None*, *help_text=None*, *verbose_name=None*, *index_type=None*, *db_column=None*, *_hidden=False*)

Bases: SmallIntegerField, *IntegerField*

Extending peewee.SmallIntegerField.

Parameters

- **orderable** (*bool* | Field) –
- **searchable** (*bool* | Field) –
- **null** (*bool*) –

- **index** (*bool*) –
- **unique** (*bool*) –
- **column_name** (*Optional[str]*) –
- **default** (*Any*) –
- **primary_key** (*bool*) –
- **constraints** (*Optional[List[SQL]]*) –
- **sequence** (*Optional[str]*) –
- **collation** (*Optional[str]*) –
- **unindexed** (*bool*) –
- **choices** (*Optional[Iterable[Tuple[str, Any]]]*) –
- **help_text** (*Optional[str]*) –
- **verbose_name** (*Optional[str]*) –
- **index_type** (*Optional[str]*) –
- **db_column** (*Optional[str]*) –
- **_hidden** (*bool*) –

orderable: Union[bool, *Field*]

DataTables integration orderable flag.

searchable: Union[bool, *Field*]

DataTables integration searchable flag.

class flask_datatables.fields.**TextField**(*orderable=None*, *searchable=None*, *null=False*, *index=False*, *unique=False*, *column_name=None*, *default=None*, *primary_key=False*, *constraints=None*, *sequence=None*, *collation=None*, *unindexed=False*, *choices=None*, *help_text=None*, *verbose_name=None*, *index_type=None*, *db_column=None*, *_hidden=False*)

Bases: `TextField`, `_StringField`

Extending `peewee.TextField`.

Parameters

- **orderable** (*bool | Field*) –
- **searchable** (*bool | Field*) –
- **null** (*bool*) –
- **index** (*bool*) –
- **unique** (*bool*) –
- **column_name** (*Optional[str]*) –
- **default** (*Any*) –
- **primary_key** (*bool*) –
- **constraints** (*Optional[List[SQL]]*) –
- **sequence** (*Optional[str]*) –

- **collation** (*Optional[str]*) –
- **unindexed** (*bool*) –
- **choices** (*Optional[Iterable[Tuple[str, Any]]]*) –
- **help_text** (*Optional[str]*) –
- **verbose_name** (*Optional[str]*) –
- **index_type** (*Optional[str]*) –
- **db_column** (*Optional[str]*) –
- **_hidden** (*bool*) –

orderable: Union[bool, *Field*]

DataTables integration orderable flag.

searchable: Union[bool, *Field*]

DataTables integration searchable flag.

class flask_datatables.fields.**TimeField**(*formats=None*, **args*, ***kwargs*)

Bases: TimeField, _BaseFormattedField

Extending peewee.TimeField.

orderable: Union[bool, *Field*]

DataTables integration orderable flag.

searchable: Union[bool, *Field*]

DataTables integration searchable flag.

class flask_datatables.fields.**TimestampField**(**args*, ***kwargs*)

Bases: TimestampField, *BigIntegerField*

Extending peewee.TimestampField.

static dt_search(*value*)

Convert value for DataTables searching operation.

Parameters
value (datetime) – Source value.

Return type
str

Returns
Converted value.

orderable: Union[bool, *Field*]

DataTables integration orderable flag.

searchable: Union[bool, *Field*]

DataTables integration searchable flag.

class flask_datatables.fields.**UUIDField**(*orderable=None*, *searchable=None*, *null=False*, *index=False*, *unique=False*, *column_name=None*, *default=None*, *primary_key=False*, *constraints=None*, *sequence=None*, *collation=None*, *unindexed=False*, *choices=None*, *help_text=None*, *verbose_name=None*, *index_type=None*, *db_column=None*, *_hidden=False*)

Bases: UUIDField, *Field*

Extending peewee.UUIDField.

Parameters

- **orderable** (*bool | Field*) –
- **searchable** (*bool | Field*) –
- **null** (*bool*) –
- **index** (*bool*) –
- **unique** (*bool*) –
- **column_name** (*Optional[str]*) –
- **default** (*Any*) –
- **primary_key** (*bool*) –
- **constraints** (*Optional[List[SQL]]*) –
- **sequence** (*Optional[str]*) –
- **collation** (*Optional[str]*) –
- **unindexed** (*bool*) –
- **choices** (*Optional[Iterable[Tuple[str, Any]]]*) –
- **help_text** (*Optional[str]*) –
- **verbose_name** (*Optional[str]*) –
- **index_type** (*Optional[str]*) –
- **db_column** (*Optional[str]*) –
- **_hidden** (*bool*) –

static dt_search(*value*)

Convert value for DataTables searching operation.

Parameters
value (UUID) – Source value.

Return type
str

Returns
Converted value.

property dt_searchable: *Field* | None

DataTables default searchable field.

orderable: Union[bool, *Field*]

DataTables integration orderable flag.

searchable: Union[bool, *Field*]

DataTables integration searchable flag.

CHAPTER

FOUR

UTILITIES & AUXILIARIES

We provided some auxiliary functions for `Flask-DataTables`.

- *render_macro()* renders a given `macro` from the Jinja templates
- *prepare_response()* is the default built-in method for `factory` parameter of *Model.search*
- *parse_request()* is the utility function to parse DataTables client-side query parameters from the URL

`flask_datatables.utils.`**`render_macro`**(*template_name_or_list*, *macro*, ***context*)

Evaluates and renders a **macro** from the template.

Parameters

- **`template_name_or_list`** (Union[str, List[str]]) – The name of the template to be rendered, or an iterable with template names the first one existing will be rendered.
- **`macro`** (str) – The name of macro to be called.
- **`context`** (*Any*) –

Keyword Arguments

`**context` – The variables that should be available in the context of the template.

Return type

str

Returns

The rendered macro.

`flask_datatables.utils.`**`prepare_response`**(*template*)

Prepare response object data.

The function returns a wrapper function to use the `template` as a factory to render HTML response blocks. The Jinja templates should have **macro** blocks for each target field named after `render_{field_name}` and takes only one argument `record` as the selected data model record.

Parameters

`template` (Union[str, List[str]]) – Path to the macro template.

Return type

Callable[[Model], *ObjectData*]

Returns

Prepared response object data.

See also:

See *flask_datatables.utils.render_macro()* for more information.

flask_datatables.utils.**parse_request**(*args=None*)

Parse flask.request.args as Query.

Parameters
args (Optional[ImmutableMultiDict]) – Original request arguments. The default value is inferred from request.args.

Return type
Query

Returns
Parsed query dictionary.

CHAPTER

FIVE

TYPE HINTS

As described in the DataTables documentation, the client side sends specific format of query parameters to the server side.

When making a request to the server using server-side processing, DataTables will send data as described in *Query* in order to let the server know what data is required.

Once DataTables has made a request for data, with the above parameters sent to the server, it expects JSON data to be returned to it, with the parameters set as described in *Response*.

class flask_datatables.typing.**Column**

Bases: TypedDict

Column data.

data: str

Column's data source, as defined by `columns.data`.

name: str

Column's name, as defined by `columns.name`.

orderable: bool

Flag to indicate if this column is orderable (`True`) or not (`False`). This is controlled by `columns.orderable`.

search: *Search*

Search information.

searchable: bool

Flag to indicate if this column is searchable (`True`) or not (`False`). This is controlled by `columns.searchable`.

class flask_datatables.typing.**ObjectData**

Bases: TypedDict

Server-side processing return using objects.

DT_RowAttr: Dict[str, Any]

Add the data contained in the object to the row `tr` node as attributes. The object keys are used as the attribute keys and the values as the corresponding attribute values. This is performed using using the **`jQuery`_** `param()` method. Please note that this option requires **DataTables 1.10.5** or newer.

DT_RowClass: str

Add this class to the `tr` node.

DT_RowData: `Dict[str, Any]`

Add the data contained in the object to the row using the `jQuery`_ `data()` method to set the data, which can also then be used for later retrieval (for example on a click event).

DT_RowId: `str`

Set the ID property of the `tr` node to this value.

class flask_datatables.typing.**Order**

Bases: `TypedDict`

Ordering information.

column: `int`

Column to which ordering should be applied. This is an index reference to the `columns` array of information that is also submitted to the server.

dir: `Literal['asc', 'desc']`

Ordering direction for this column. It will be `asc` or `desc` to indicate ascending ordering or descending ordering, respectively.

class flask_datatables.typing.**Query**

Bases: `TypedDict`

Sent parameters.

When making a request to the server using server-side processing, DataTables will send the following data in order to let the server know what data is required.

columns: `List[`*Column*`]`

Column data.

draw: `int`

Draw counter. This is used by DataTables to ensure that the Ajax returns from server-side processing requests are drawn in sequence by DataTables (Ajax requests are asynchronous and thus can return out of sequence). This is used as part of the `draw` return parameter (see below).

length: `int`

Number of records that the table can display in the current draw. It is expected that the number of records returned will be equal to this number, unless the server has fewer records to return. Note that this can be -1 to indicate that all records should be returned (although that negates any benefits of server-side processing!)

order: `List[`*Order*`]`

Ordering information.

search: *Search*

Search information.

start: `int`

Paging first record indicator. This is the start point in the current data set (0 index based - i.e. 0 is the first record).

class flask_datatables.typing.**Response**

Bases: `TypedDict`

Returned data.

Once DataTables has made a request for data, with the above parameters sent to the server, it expects JSON data to be returned to it, with the following parameters set.

data: `List[Union[List[Any], ObjectData]]`

The data to be displayed in the table. This is an array of data source objects, one for each row, which will be used by DataTables. Note that this parameter's name can be changed using the `ajax` option's `dataSrc` property.

draw: `int`

The draw counter that this object is a response to - from the `draw` parameter sent as part of the data request. Note that it is **strongly recommended for security reasons** that you cast this parameter to an integer, rather than simply echoing back to the client what it sent in the `draw` parameter, in order to prevent Cross Site Scripting (XSS) attacks.

error: `Optional[str]`

If an error occurs during the running of the server-side processing script, you can inform the user of this error by passing back the error message to be displayed using this parameter. Do not include if there is no error.

Type
Optional

recordsFiltered: `int`

Total records, after filtering (i.e. the total number of records after filtering has been applied - not just the number of records being returned for this page of data).

recordsTotal: `int`

Total records, before filtering (i.e. the total number of records in the database)

class flask_datatables.typing.**Search**

Bases: `TypedDict`

Search information.

regex: `bool`

`True` if the global filter should be treated as a regular expression for advanced searching, `False` otherwise. Note that normally server-side processing scripts will not perform regular expression searching for performance reasons on large data sets, but it is technically possible and at the discretion of your script.

value: `str`

Global search value. To be applied to all columns which have `searchable` as `True`.

flask_datatables.typing.**ArrayData**

Server-side processing return using arrays as the data source for the table.

alias of `List[Any]`

As DataTables is a quite power and useful JavaScript library for manipulating and displaying data, we intended to make integration of the client-side DataTables scripts with the server-side processing based on `Flask` and `peewee`.

CHAPTER

SIX

INSTALLATION

Note: As we have noticed, there's already a `Flask-DataTables` library available on PyPI. However, this package was intended for integration with `SQLAlchemy` instead of `peewee`.

To start with, simply install the `Flask-DataTables` package from PyPI:

```
pip install Flask-DataTables-peewee
```

or should you prefer the bleeding edge version:

```
git clone https://github.com/tekidltd/Flask-DataTables.git
cd Flask-DataTables
pip install .
```

CHAPTER

SEVEN

USAGE

`Flask-DataTables` is quite simple to use, just declare your data model in the preferable way from `peewee` and voilà, that's it.

Taking examples from the `peewee` documentation, we can have a DataTables integrated data model just as below:

```
from flask import Flask
from flask_datatables import (CharField, DataTable, DateTimeField,
                              ForeignKeyField, Metadata, TextField)

DATABASE = 'postgresql://postgres:password@localhost:5432/my_database'

app = Flask(__name__)
app.config.from_object(__name__)

db_wrapper = DataTable(app)


class User(db_wrapper.Model):
    username = CharField(unique=True)

    class Meta(Metadata):
        datatables = True


class Tweet(db_wrapper.Model):
    user = ForeignKeyField(User, backref='tweets')
    content = TextField()
    timestamp = DateTimeField(default=datetime.datetime.now)

    class Meta(Metadata):
        datatables = True
```

And now, you may simply call *`Tweet.search`* to perform the server-side processing queries.

See also:

It is also possible to customise the orderable and/or searchable fields through *`Field`* parameters, and their corresponding behaviours by subclassing the *`Field`* classes.

CHAPTER

EIGHT

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

f

INDEX

E

F

I

L

M

N

O

P