Camping Le Maine Blanc

Version 1.0

Nathalie Darnaudat

15 septembre 2025

Contents:

Bienvenue dans la documentation du site du Camping Le Maine Blanc.

Cette documentation couvre les modules Python, les vues, les formulaires, les admin et les tests du projet.

Vous pouvez ajouter votre contenu en utilisant la syntaxe `reStructuredText`. Pour plus de détails, consultez la documentation : reStructuredText

CHAPITRE 1

Modules et Apps

1.1 Core

— Modèles

class core.models.**CampingInfo**(**args*, ***kwargs*)

Bases : TranslatableModel

Stores general camping rules and schedules.

Fields include reception hours, arrival and departure times, and gate opening hours.

Security :

— No direct user input, so XSS or SQL injection risk is minimal.

translations

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example :

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

Parent.children is a ReverseManyToOneDescriptor instance.

Most of the implementation is delegated to a dynamically defined manager class built by create_forward_many_to_many_manager() defined below.

exception DoesNotExist

Bases : ObjectDoesNotExist

exception MultipleObjectsReturned

Bases : MultipleObjectsReturned

arrivals_end_high

Descriptor for translated attributes.

This attribute proxies all get/set calls to the translated model.

arrivals_end_low

Descriptor for translated attributes.

This attribute proxies all get/set calls to the translated model.

arrivals_start_high

Descriptor for translated attributes.

This attribute proxies all get/set calls to the translated model.

departure_end

Descriptor for translated attributes.

This attribute proxies all get/set calls to the translated model.

id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

portal_end

Descriptor for translated attributes.

This attribute proxies all get/set calls to the translated model.

portal_start

Descriptor for translated attributes.

This attribute proxies all get/set calls to the translated model.

welcome_afternoon_end

Descriptor for translated attributes.

This attribute proxies all get/set calls to the translated model.

welcome_afternoon_start

Descriptor for translated attributes.

This attribute proxies all get/set calls to the translated model.

welcome_end

Descriptor for translated attributes.

This attribute proxies all get/set calls to the translated model.

welcome_start

Descriptor for translated attributes.

This attribute proxies all get/set calls to the translated model.

class core.models.**SwimmingPoolInfo**(**args*, ***kwargs*)

Bases : TranslatableModel

Stores swimming pool schedule information.

Security :

— No user input processed.

translations

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example :

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

exception DoesNotExist

Bases : `ObjectDoesNotExist`

exception MultipleObjectsReturned

Bases : `MultipleObjectsReturned`

id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

pool_opening_end

Descriptor for translated attributes.

This attribute proxies all get/set calls to the translated model.

pool_opening_start

Descriptor for translated attributes.

This attribute proxies all get/set calls to the translated model.

class core.models.**FoodInfo**(**args*, ***kwargs*)

Bases : `TranslatableModel`

Stores information about food services such as food trucks, pizza days, bread reservations, and bar opening hours.

Translations :
— Automatically translates burger and pizza days using DeepL API.
— Logs translation errors without interrupting save process.

Security :
— No user input is directly processed here, reducing XSS risk.
— DeepL API errors are caught and logged.

translations

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example :

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

save(**args*, ***kwargs*)

Overrides save to automatically translate certain fields into multiple languages using DeepL API. Errors are logged but do not interrupt saving.

Security :
— Only controlled default values are translated ; no user input is processed.

exception DoesNotExist

Bases : ObjectDoesNotExist

exception MultipleObjectsReturned

Bases : MultipleObjectsReturned

bar_hours_end

Descriptor for translated attributes.

This attribute proxies all get/set calls to the translated model.

bar_hours_start

Descriptor for translated attributes.

This attribute proxies all get/set calls to the translated model.

bread_hours_end

Descriptor for translated attributes.

This attribute proxies all get/set calls to the translated model.

bread_hours_reservations

Descriptor for translated attributes.

This attribute proxies all get/set calls to the translated model.

bread_hours_start

Descriptor for translated attributes.

This attribute proxies all get/set calls to the translated model.

burger_food_days

Descriptor for translated attributes.

This attribute proxies all get/set calls to the translated model.

burger_food_hours_end

Descriptor for translated attributes.

This attribute proxies all get/set calls to the translated model.

burger_food_hours_start

Descriptor for translated attributes.

This attribute proxies all get/set calls to the translated model.

id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

pizza_food_days

Descriptor for translated attributes.

This attribute proxies all get/set calls to the translated model.

class core.models.**LaundryInfo**(**args*, ***kwargs*)

Bases : TranslatableModel

Stores information about laundry services and pricing.

Security :

— Only fixed numeric values ; no user input.

translations

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example :

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

exception DoesNotExist

Bases : `ObjectDoesNotExist`

exception MultipleObjectsReturned

Bases : `MultipleObjectsReturned`

dryer_price

Descriptor for translated attributes.

This attribute proxies all get/set calls to the translated model.

id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

washing_machine_price

Descriptor for translated attributes.

This attribute proxies all get/set calls to the translated model.

class core.models.**CampingInfoTranslation**(*id*, *language_code*, *welcome_start*, *welcome_end*, *welcome_afternoon_start*, *welcome_afternoon_end*, *arrivals_start_high*, *arrivals_end_high*, *arrivals_end_low*, *departure_end*, *portal_start*, *portal_end*, *master*)

Bases : `TranslatedFieldsModel`

exception DoesNotExist

Bases : `TranslationDoesNotExist`, *DoesNotExist*, *DoesNotExist*

exception MultipleObjectsReturned

Bases : `MultipleObjectsReturned`

arrivals_end_high

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

arrivals_end_low

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

arrivals_start_high

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

departure_end

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

get_language_code_display(*, *field=<parler.utils.compat.HideChoicesCharField : language_code>*)

id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

language_code

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

master

The mandatory Foreign key field to the shared model.

master_id

objects = <django.db.models.manager.Manager object>

portal_end

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

portal_start

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

welcome_afternoon_end

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

welcome_afternoon_start

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

welcome_end

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

welcome_start

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

class core.models.**FoodInfoTranslation**(*id*, *language_code*, *burger_food_days*, *burger_food_hours_start*, *burger_food_hours_end*, *pizza_food_days*, *bread_hours_reservations*, *bread_hours_start*, *bread_hours_end*, *bar_hours_start*, *bar_hours_end*, *master*)

Bases : TranslatedFieldsModel

exception DoesNotExist

Bases : TranslationDoesNotExist, *DoesNotExist*, *DoesNotExist*

exception MultipleObjectsReturned

Bases : MultipleObjectsReturned

bar_hours_end

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

bar_hours_start

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

bread_hours_end

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

bread_hours_reservations

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

bread_hours_start

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

burger_food_days

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

burger_food_hours_end

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

burger_food_hours_start

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

get_language_code_display(*, *field=<parler.utils.compat.HideChoicesCharField : language_code>*)

id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

language_code

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

master

The mandatory Foreign key field to the shared model.

master_id

objects = <django.db.models.manager.Manager object>

pizza_food_days

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

class core.models.**LaundryInfoTranslation**(*id*, *language_code*, *washing_machine_price*, *dryer_price*, *master*)

Bases : TranslatedFieldsModel

exception DoesNotExist

Bases : TranslationDoesNotExist, *DoesNotExist*, *DoesNotExist*

exception MultipleObjectsReturned

Bases : MultipleObjectsReturned

dryer_price

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

get_language_code_display(*, *field=<parler.utils.compat.HideChoicesCharField : language_code>*)

id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

language_code

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

master

The mandatory Foreign key field to the shared model.

master_id

objects = <django.db.models.manager.Manager object>

washing_machine_price

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

class core.models.**SwimmingPoolInfoTranslation**(*id*, *language_code*, *pool_opening_start*, *pool_opening_end*, *master*)

Bases : TranslatedFieldsModel

exception DoesNotExist

Bases : TranslationDoesNotExist, *DoesNotExist*, *DoesNotExist*

exception MultipleObjectsReturned

Bases : MultipleObjectsReturned

get_language_code_display(*, *field=<parler.utils.compat.HideChoicesCharField : language_code>*)

id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

language_code

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

master

The mandatory Foreign key field to the shared model.

master_id

objects = <django.db.models.manager.Manager object>

pool_opening_end

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

pool_opening_start

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

— Vues

core.views.**home_view**(*request*)

Render the homepage.

Security :

— No user input processed ; safe from XSS or injection.

core.views.**about_view**(*request*)

Render the about page.

Security :

— No user input processed.

core.views.**infos_view**(*request*)

Render the information page with pricing, supplements, seasons, mobile homes, camping info, capacity, and other prices.

Features :

— Groups normal and worker prices
— Handles supplements and visitor prices
— Dynamically sets mobile home descriptions based on current language

Security :

— Only retrieves data from the database
— No user input is processed
— Safe against XSS and injection

core.views.**services_view**(*request*)

Render the Services page including swimming pool, food, and laundry info.

Security :

— Only reads database objects
— No user input processed

core.views.**accommodations_view**(*request*)

Render the Accommodations page.

Security :

— No user input processed

core.views.**activities_view**(*request*)

Render the Activities page.

Security :

— No user input processed

core.views.**legal_view**(*request*)

Render the Legal page.

Security :

— Static content, safe

core.views.**privacy_view**(*request*)

Render the Privacy Policy page.

Security :

— Static content, safe

core.views.**not_found_view**(*request*, *exception=None*)

Render the 404 Not Found page.

Security :

— Static content

— Exception handling passed safely

— Admin

class core.apps.**CoreConfig**(*app_name*, *app_module*)

Bases : AppConfig

Configuration for the “core” app, which manages general campsite information.

default_auto_field = 'django.db.models.BigAutoField'

name = 'core'

verbose_name = 'Informations diverses'

1.2 Tests Core

— Modèles

core.tests.test_models.**test_campinginfo_fixture**(*campinginfo_fr*)

Verify that the CampingInfo fixture is correctly created and times are set.

core.tests.test_models.**test_swimmingpoolinfo_fixture**(*swimmingpoolinfo_fr*)

Verify that the SwimmingPoolInfo fixture is correctly created and opening times are correct.

core.tests.test_models.**test_foodinfo_fixture**(*foodinfo_fr*)

Verify that the FoodInfo fixture is correctly created and food hours are set.

core.tests.test_models.**test_laundryinfo_fixture**(*laundryinfo_fr*)

Verify that the LaundryInfo fixture is correctly created and prices are correct.

— Vues

core.tests.test_views.**test_home_view**(*client*)

Home page should load successfully and use the correct template.

core.tests.test_views.**test_about_view**(*client*)

About page should load successfully and use the correct template.

core.tests.test_views.**test_accommodations_view**(*client*)

Accommodations page should load successfully and use the correct template.

core.tests.test_views.**test_activities_view**(*client*)

Activities page should load successfully and use the correct template.

core.tests.test_views.**test_legal_view**(*client*)

Legal page should load successfully and use the correct template.

core.tests.test_views.**test_privacy_view**(*client*)

Privacy policy page should load successfully and use the correct template.

core.tests.test_views.**test_not_found_view**(*client*)

Not found page should return 404 and use the correct template.

core.tests.test_views.**test_infos_view**(*client*, *campinginfo_fr*, *mobilehome_fr*)

Infos view should load successfully and provide camping info and mobilehome data.

core.tests.test_views.**test_services_view**(*client*, *swimmingpoolinfo_fr*, *foodinfo_fr*, *laundryinfo_fr*)

Services view should load successfully and provide swimming, food, and laundry info.

1.3 Reservations

— Vues

reservations.views.**reservation_request_view**(*request*)

Handles reservation requests from users :
- — GET : display empty reservation form
- — POST : validate form, translate message, send email to admin, show success message

Security measures :
- — Form validation via ReservationRequestForm
- — Escape user-provided message to prevent XSS
- — Deepl API errors are caught and logged, original message preserved
- — Email sending is wrapped, fail_silently=False
- — No sensitive data (API key) is exposed to templates

— Formulaires

class reservations.forms.**ReservationRequestForm**(*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.utils.ErrorList'>*, *label_suffix=None*, *empty_permitted=False*, *field_order=None*, *use_required_attribute=None*, *renderer=None*, *bound_field_class=None*)

Bases : Form

Form to handle reservation requests from customers.

Fields :
- — Customer info : name, first_name, address, postal_code, city, phone, email
- — Reservation dates : start_date, end_date
- — Accommodation details : accommodation_type, tent dimensions, vehicle_length
- — Guests : adults, children_over_8, children_under_8, pets
- — Electricity info : electricity, cable_length
- — Message : optional free text

Security :
- — Fields are validated and escaped to prevent XSS
- — Conditional fields validated in *clean()*

ACCOMMODATION_CHOICES = [('', 'Choisissez un type'), ('tent', 'Tente'), ('car_tent', 'Voiture tente'), ('caravan', 'Caravane'), ('fourgon', 'Fourgon aménagé'), ('van', 'Van'), ('camping_car', 'Camping-car'), ('mobil-home', 'Mobil-home')]

ELECTRICITY_CHOICES = [('yes', 'Avec électricité'), ('no', 'Sans électricité')]

clean()

Custom validation :
- — Validate dates (arrival < departure, not in past)
- — Validate conditional fields (tent dimensions, vehicle length, cable if electricity)
- — Escape free-text message to prevent XSS

base_fields = {'accommodation_type': <django.forms.fields.ChoiceField object>, 'address': <django.forms.fields.CharField object>, 'adults': <django.forms.fields.ChoiceField object>, 'cable_length': <django.forms.fields.DecimalField object>, 'children_over_8': <django.forms.fields.ChoiceField object>, 'children_under_8': <django.forms.fields.ChoiceField object>, 'city': <django.forms.fields.CharField object>, 'electricity': <django.forms.fields.ChoiceField object>, 'email': <django.forms.fields.EmailField object>, 'end_date': <django.forms.fields.DateField object>, 'first_name': <django.forms.fields.CharField object>, 'message': <django.forms.fields.CharField object>, 'name': <django.forms.fields.CharField object>, 'pets': <django.forms.fields.ChoiceField object>, 'phone': <django.forms.fields.CharField object>, 'postal_code': <django.forms.fields.CharField object>, 'start_date': <django.forms.fields.DateField object>, 'tent_length': <django.forms.fields.DecimalField object>, 'tent_width': <django.forms.fields.DecimalField object>, 'vehicle_length': <django.forms.fields.DecimalField object>}

declared_fields = {'accommodation_type': <django.forms.fields.ChoiceField object>, 'address': <django.forms.fields.CharField object>, 'adults': <django.forms.fields.ChoiceField object>, 'cable_length': <django.forms.fields.DecimalField object>, 'children_over_8': <django.forms.fields.ChoiceField object>, 'children_under_8': <django.forms.fields.ChoiceField object>, 'city': <django.forms.fields.CharField object>, 'electricity': <django.forms.fields.ChoiceField object>, 'email': <django.forms.fields.EmailField object>, 'end_date': <django.forms.fields.DateField object>, 'first_name': <django.forms.fields.CharField object>, 'message': <django.forms.fields.CharField object>, 'name': <django.forms.fields.CharField object>, 'pets': <django.forms.fields.ChoiceField object>, 'phone': <django.forms.fields.CharField object>, 'postal_code': <django.forms.fields.CharField object>, 'start_date': <django.forms.fields.DateField object>, 'tent_length': <django.forms.fields.DecimalField object>, 'tent_width': <django.forms.fields.DecimalField object>, 'vehicle_length': <django.forms.fields.DecimalField object>}

property media

Return all media required to render the widgets on this form.

— Configuration

class reservations.apps.**ReservationsConfig**(*app_name*, *app_module*)

Bases : AppConfig

Configuration for the “reservations” app, which handles booking requests and forms.

default_auto_field = 'django.db.models.BigAutoField'

name = 'reservations'

1.4 Tests Reservations

— Vues

reservations.tests.test_views.**valid_reservation_data**()

Valid POST data for the reservation form

reservations.tests.test_views.**test_get_reservation_request_view**(*client*)

GET request to reservation_request returns the form and correct template.

reservations.tests.test_views.**test_post_valid_reservation_with_datetime_and_label**(*mock_translate*, *mock_send*, *client*, *valid_reservation_data*)

POST valid reservation sends email, shows success, and sets submission_datetime.

reservations.tests.test_views.**test_post_invalid_reservation**(*client*)

POST request with missing required fields returns form errors

— Formulaires

reservations.tests.test_forms.**valid_form_data**()

Valid data for the booking form

reservations.tests.test_forms.**test_form_valid**(*valid_form_data*)

Form with all valid data should be valid.

reservations.tests.test_forms.**test_form_missing_required_field**(*valid_form_data*)

Form missing "name" field should be invalid and report error.

reservations.tests.test_forms.**test_form_dates_invalid**(*valid_form_data*)

Start date in the past or end date not after start should be invalid.

reservations.tests.test_forms.**test_form_tent_requires_dimensions**(*valid_form_data*)

Tent accommodation requires both length and width to be filled.

reservations.tests.test_forms.**test_form_vehicle_requires_length**(*valid_form_data*)

Vehicle-type accommodation requires vehicle_length to be filled.

reservations.tests.test_forms.**test_form_cable_requires_if_electricity_yes**(*valid_form_data*)

If electricity is "yes", cable_length must be filled.

1.5 Bookings

— Modèles

class bookings.models.**SupplementPrice**(**args*, ***kwargs*)

Bases : Model

Stores additional pricing options for reservations.

Fields :

- — extra_adult_price : price per extra adult
- — child_over_8_price : price per child over 8 years
- — child_under_8_price : price per child under 8 years
- — pet_price : price per pet
- — extra_vehicle_price : price per additional vehicle
- — extra_tent_price : price per additional tent
- — visitor_price_without_swimming_pool : visitor price without pool
- — visitor_price_with_swimming_pool : visitor price with pool

Security :

- — Only stores numeric data
- — No user input processed directly

extra_adult_price

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

child_over_8_price

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

child_under_8_price

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

pet_price

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

extra_vehicle_price

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

extra_tent_price

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

visitor_price_without_swimming_pool

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

visitor_price_with_swimming_pool

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

exception DoesNotExist

Bases : `ObjectDoesNotExist`

exception MultipleObjectsReturned

Bases : `MultipleObjectsReturned`

id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

objects = <django.db.models.manager.Manager object>

prices

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example :

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

class bookings.models.**Price**(**args*, ***kwargs*)

Bases : `Model`

Stores the base pricing for different types of accommodations and seasons.

Fields :

— booking_type : main type (tent, caravan, camping_car)
— season : low/mid/high
— is_worker : boolean flag for worker rates
— price_1_person_with_electricity, price_2_persons_with_electricity
— price_1_person_without_electricity, price_2_persons_without_electricity
— supplements : related SupplementPrice

`- save()`

auto-assigns included_people and supplements

`- clean()`

validates camping-car pricing rules

Security :

— Only numeric and enum fields
— Validation ensures business rules are respected

```
SEASON_CHOICES = [('low', 'Basse Saison'), ('mid', 'Moyenne Saison'), ('high',
'Haute Saison')]
```

```
TYPE_CHOICES = [('tent', 'Tente / Voiture Tente'), ('caravan', 'Caravane / Fourgon /
Van'), ('camping_car', 'Camping-car'), ('other', 'Emplacement ouvrier weekend')]
```

`booking_type`

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

`season`

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

`is_worker`

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

`price_1_person_with_electricity`

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

`price_2_persons_with_electricity`

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

`price_1_person_without_electricity`

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

`price_2_persons_without_electricity`

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

`included_people`

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

worker_week_price

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

weekend_price_without_electricity

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

weekend_price_with_electricity

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

supplements

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example :

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

save(**args*, ***kwargs*)

Automatically assigns included_people based on booking_type and ensures a SupplementPrice is associated if missing.

clean()

Validate business rules before saving. For camping_car, ensures that "1 person" price fields are empty. Raises ValidationError on violation.

exception DoesNotExist

Bases : `ObjectDoesNotExist`

exception MultipleObjectsReturned

Bases : `MultipleObjectsReturned`

get_booking_type_display(*, *field=<django.db.models.fields.CharField : booking_type>*)

get_season_display(*, *field=<django.db.models.fields.CharField : season>*)

id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

objects = <django.db.models.manager.Manager object>

supplements_id

class bookings.models.**OtherPrice**(**args*, ***kwargs*)

Bases : `TranslatableModel`

Stores miscellaneous pricing info such as tourist tax.

Fields :

- current_year : current calendar year
- tourist_tax_date : date when tourist tax applies
- price_tourist_tax : price per night/person

Security :

— Read-only for user input
— Only numeric and date fields

translations

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example :

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

exception DoesNotExist

Bases : `ObjectDoesNotExist`

exception MultipleObjectsReturned

Bases : `MultipleObjectsReturned`

current_year

Descriptor for translated attributes.

This attribute proxies all get/set calls to the translated model.

id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

price_tourist_tax

Descriptor for translated attributes.

This attribute proxies all get/set calls to the translated model.

tourist_tax_date

Descriptor for translated attributes.

This attribute proxies all get/set calls to the translated model.

class bookings.models.**Capacity**(**args*, ***kwargs*)

Bases : `Model`

Stores maximum capacity for each booking type.

Fields :

— booking_type : type of accommodation
— max_places : maximum allowed placements
— number_locations : total locations
— number_mobile_homes : number of mobile homes

Security :

— Only numeric data
— Used for internal validation (check_capacity)

booking_type

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

max_places

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

number_locations

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

number_mobile_homes

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

exception DoesNotExist

Bases : ObjectDoesNotExist

exception MultipleObjectsReturned

Bases : MultipleObjectsReturned

get_booking_type_display(*, *field=<django.db.models.fields.CharField : booking_type>*)

id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

objects = <django.db.models.manager.Manager object>

class bookings.models.**Booking**(**args*, ***kwargs*)

Bases : Model

Stores client booking information.

Fields :

- Personal info : last_name, first_name, address, postal_code, city, phone, email
- Reservation info : start_date, end_date, booking_type, booking_subtype, electricity
- Counts : adults, children_over_8, children_under_8, pets
- Extras : extra_vehicle, extra_tent, deposit_paid
- Timestamps : created_at, updated_at

- get_season()

determines season based on start_date

- calculate_total_price()

calculates total cost including supplements

- calculate_deposit()

calculates 15% deposit

- save()

auto-assigns main type, included_people, supplements

- check_capacity()

checks if capacity is available

- clean()

validates business rules and capacity

Security :

- Input validation via clean() ensures business rules are respected
- Numeric and enum fields only ; safe from injection
- check_capacity prevents overbooking
- All calculations server-side

TYPE_CHOICES = [('tent', 'Tente / Voiture Tente'), ('caravan', 'Caravane / Fourgon / Van'), ('camping_car', 'Camping-car'), ('other', 'Emplacement ouvrier weekend')]

SUBTYPE_CHOICES = [('tent', 'Tente'), ('car_tent', 'Voiture Tente'), ('caravan', 'Caravane'), ('fourgon', 'Fourgon'), ('van', 'Van'), ('camping_car', 'Camping-car')]

ELECTRICITY_CHOICES = [('yes', 'Avec électricité'), ('no', 'Sans électricité')]

last_name
: A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

first_name
: A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

address
: A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

postal_code
: A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

city
: A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

phone
: A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

email
: A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

start_date
: A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

end_date
: A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

booking_type
: A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

booking_subtype
: A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

electricity
: A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

deposit_paid
: A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

tent_length

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

tent_width

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

vehicle_length

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

cable_length

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

adults

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

children_over_8

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

children_under_8

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

pets

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

extra_vehicle

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

extra_tent

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

created_at

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

updated_at

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

MAIN_TYPE_MAP = {'camping_car': 'camping_car', 'car_tent': 'tent', 'caravan': 'caravan', 'fourgon': 'caravan', 'tent': 'tent', 'van': 'caravan'}

created_at_display()

Format creation date for admin display.

updated_at_display()

Format updated date for admin display.

get_season()

Determine season based on start_date.

calculate_total_price(*supplement=None*)

Calculate total price including extras and supplements. Ensures nights >= 1 and applies correct base price depending on booking_type and electricity.

calculate_deposit()

Calculate 15% deposit of total price, rounded to 2 decimals.

save(**args*, ***kwargs*)

Automatically sets booking_type from booking_subtype, included_people, and assigns a SupplementPrice if missing.

check_capacity()

Checks availability for given dates and booking type. Raises ValidationError if capacity exceeded.

clean()

Validates business rules : - Capacity availability - Camping_car pricing rules Raises ValidationError on violation.

exception DoesNotExist

Bases : `ObjectDoesNotExist`

exception MultipleObjectsReturned

Bases : `MultipleObjectsReturned`

get_booking_subtype_display(*, *field=<django.db.models.fields.CharField : booking_subtype>*)

get_booking_type_display(*, *field=<django.db.models.fields.CharField : booking_type>*)

get_electricity_display(*, *field=<django.db.models.fields.CharField : electricity>*)

get_next_by_created_at(*, *field=<django.db.models.fields.DateTimeField : created_at>*, *is_next=True*, ***kwargs*)

get_next_by_end_date(*, *field=<django.db.models.fields.DateField : end_date>*, *is_next=True*, ***kwargs*)

get_next_by_start_date(*, *field=<django.db.models.fields.DateField : start_date>*, *is_next=True*, ***kwargs*)

get_next_by_updated_at(*, *field=<django.db.models.fields.DateTimeField : updated_at>*, *is_next=True*, ***kwargs*)

get_previous_by_created_at(*, *field=<django.db.models.fields.DateTimeField : created_at>*, *is_next=False*, ***kwargs*)

get_previous_by_end_date(*, *field=<django.db.models.fields.DateField : end_date>*, *is_next=False*, ***kwargs*)

get_previous_by_start_date(*, *field=<django.db.models.fields.DateField : start_date>*, *is_next=False*, ***kwargs*)

get_previous_by_updated_at(*, *field=<django.db.models.fields.DateTimeField : updated_at>*, *is_next=False*, ***kwargs*)

id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

objects = <django.db.models.manager.Manager object>

class bookings.models.**MobileHome**(**args*, ***kwargs*)

Bases : Model

Stores mobile home info and translations.

Fields :

— name, description_text : French version
— name_en/es/de/nl, description_en/es/de/nl : translated versions
— night_price, week_low/mid/high : nightly and weekly prices
— is_worker_home : reserved for workers
— worker_price_1p/2p/3p : weekly prices for workers

- save()

automatically translates name and description via DeepL API

Security :

— Only numeric/text fields
— DeepL translations handled server-side
— No user input directly sent to API

name

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

name_en

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

name_es

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

name_de

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

name_nl

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

description_text

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

description_en

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

description_es

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

description_de

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

description_nl

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

night_price

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

night_price_mid

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

week_low

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

week_mid

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

week_high

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

is_worker_home

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

worker_price_1p

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

worker_price_2p

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

worker_price_3p

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

save(**args*, ***kwargs*)

Automatically translate name and description to multiple languages using DeepL. Only executed if DEEPL_API_KEY is set and description_text is provided.

exception DoesNotExist

Bases : `ObjectDoesNotExist`

exception MultipleObjectsReturned

Bases : `MultipleObjectsReturned`

id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

objects = <django.db.models.manager.Manager object>

class bookings.models.**SupplementMobileHome**(**args*, ***kwargs*)

Bases : TranslatableModel

Stores extra charges for mobile homes.

Fields :

- mobile_home_deposit : security deposit
- cleaning_deposit : cleaning deposit
- bed_linen_rental : optional linen rental

Security :

- Numeric fields only
- Used for internal calculations, no user input processed

translations

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example :

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

Parent.children is a ReverseManyToOneDescriptor instance.

Most of the implementation is delegated to a dynamically defined manager class built by create_forward_many_to_many_manager() defined below.

exception DoesNotExist

Bases : ObjectDoesNotExist

exception MultipleObjectsReturned

Bases : MultipleObjectsReturned

bed_linen_rental

Descriptor for translated attributes.

This attribute proxies all get/set calls to the translated model.

cleaning_deposit

Descriptor for translated attributes.

This attribute proxies all get/set calls to the translated model.

id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

mobile_home_deposit

Descriptor for translated attributes.

This attribute proxies all get/set calls to the translated model.

class bookings.models.**SeasonInfo**(**args*, ***kwargs*)

Bases : TranslatableModel

Stores season start and end dates.

Fields :

- low_season_start/end
- mid_season_start_1/end_1, mid_season_start_2/end_2

— high_season_start/end

Security :

— Date fields only

— Used internally for price and availability calculations

translations

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example :

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

exception DoesNotExist

Bases : `ObjectDoesNotExist`

exception MultipleObjectsReturned

Bases : `MultipleObjectsReturned`

high_season_end

Descriptor for translated attributes.

This attribute proxies all get/set calls to the translated model.

high_season_start

Descriptor for translated attributes.

This attribute proxies all get/set calls to the translated model.

id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

low_season_end

Descriptor for translated attributes.

This attribute proxies all get/set calls to the translated model.

low_season_start

Descriptor for translated attributes.

This attribute proxies all get/set calls to the translated model.

mid_season_end_1

Descriptor for translated attributes.

This attribute proxies all get/set calls to the translated model.

mid_season_end_2

Descriptor for translated attributes.

This attribute proxies all get/set calls to the translated model.

mid_season_start_1

Descriptor for translated attributes.

This attribute proxies all get/set calls to the translated model.

mid_season_start_2

Descriptor for translated attributes.

This attribute proxies all get/set calls to the translated model.

class bookings.models.**OtherPriceTranslation**(*id*, *language_code*, *current_year*, *tourist_tax_date*, *price_tourist_tax*, *master*)

Bases : TranslatedFieldsModel

exception DoesNotExist

Bases : TranslationDoesNotExist, *DoesNotExist*, *DoesNotExist*

exception MultipleObjectsReturned

Bases : MultipleObjectsReturned

current_year

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

get_language_code_display(*, *field=<parler.utils.compat.HideChoicesCharField : language_code>*)

get_next_by_tourist_tax_date(*, *field=<django.db.models.fields.DateField : tourist_tax_date>*, *is_next=True*, ***kwargs*)

get_previous_by_tourist_tax_date(*, *field=<django.db.models.fields.DateField : tourist_tax_date>*, *is_next=False*, ***kwargs*)

id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

language_code

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

master

The mandatory Foreign key field to the shared model.

master_id

objects = <django.db.models.manager.Manager object>

price_tourist_tax

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

tourist_tax_date

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

class bookings.models.**SeasonInfoTranslation**(*id*, *language_code*, *low_season_start*, *low_season_end*, *mid_season_start_1*, *mid_season_end_1*, *mid_season_start_2*, *mid_season_end_2*, *high_season_start*, *high_season_end*, *master*)

Bases : TranslatedFieldsModel

exception DoesNotExist

Bases : TranslationDoesNotExist, *DoesNotExist*, *DoesNotExist*

exception MultipleObjectsReturned

Bases : MultipleObjectsReturned

get_language_code_display(*, *field=<parler.utils.compat.HideChoicesCharField : language_code>*)

get_next_by_high_season_end(*, *field=<django.db.models.fields.DateField : high_season_end>*, *is_next=True*, ***kwargs*)

get_next_by_high_season_start(*, *field=<django.db.models.fields.DateField : high_season_start>*, *is_next=True*, ***kwargs*)

get_next_by_low_season_end(*, *field=<django.db.models.fields.DateField : low_season_end>*, *is_next=True*, ***kwargs*)

get_next_by_low_season_start(*, *field=<django.db.models.fields.DateField : low_season_start>*, *is_next=True*, ***kwargs*)

get_next_by_mid_season_end_1(*, *field=<django.db.models.fields.DateField : mid_season_end_1>*, *is_next=True*, ***kwargs*)

get_next_by_mid_season_end_2(*, *field=<django.db.models.fields.DateField : mid_season_end_2>*, *is_next=True*, ***kwargs*)

get_next_by_mid_season_start_1(*, *field=<django.db.models.fields.DateField : mid_season_start_1>*, *is_next=True*, ***kwargs*)

get_next_by_mid_season_start_2(*, *field=<django.db.models.fields.DateField : mid_season_start_2>*, *is_next=True*, ***kwargs*)

get_previous_by_high_season_end(*, *field=<django.db.models.fields.DateField : high_season_end>*, *is_next=False*, ***kwargs*)

get_previous_by_high_season_start(*, *field=<django.db.models.fields.DateField : high_season_start>*, *is_next=False*, ***kwargs*)

get_previous_by_low_season_end(*, *field=<django.db.models.fields.DateField : low_season_end>*, *is_next=False*, ***kwargs*)

get_previous_by_low_season_start(*, *field=<django.db.models.fields.DateField : low_season_start>*, *is_next=False*, ***kwargs*)

get_previous_by_mid_season_end_1(*, *field=<django.db.models.fields.DateField : mid_season_end_1>*, *is_next=False*, ***kwargs*)

get_previous_by_mid_season_end_2(*, *field=<django.db.models.fields.DateField : mid_season_end_2>*, *is_next=False*, ***kwargs*)

get_previous_by_mid_season_start_1(*, *field=<django.db.models.fields.DateField : mid_season_start_1>*, *is_next=False*, ***kwargs*)

get_previous_by_mid_season_start_2(*, *field=<django.db.models.fields.DateField : mid_season_start_2>*, *is_next=False*, ***kwargs*)

high_season_end

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

high_season_start

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

language_code

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

low_season_end

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

low_season_start

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

master

The mandatory Foreign key field to the shared model.

master_id

mid_season_end_1

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

mid_season_end_2

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

mid_season_start_1

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

mid_season_start_2

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

objects = <django.db.models.manager.Manager object>

class bookings.models.**SupplementMobileHomeTranslation**(*id*, *language_code*, *mobile_home_deposit*, *cleaning_deposit*, *bed_linen_rental*, *master*)

Bases : TranslatedFieldsModel

exception DoesNotExist

Bases : TranslationDoesNotExist, *DoesNotExist*, *DoesNotExist*

exception MultipleObjectsReturned

Bases : MultipleObjectsReturned

bed_linen_rental

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

cleaning_deposit

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

get_language_code_display(*, *field=<parler.utils.compat.HideChoicesCharField : language_code>*)

id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

language_code

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

master

The mandatory Foreign key field to the shared model.

master_id

mobile_home_deposit

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

objects = <django.db.models.manager.Manager object>

— Vues

bookings.views.**booking_form**(*request*)

Display and process the initial booking form where the user selects : - Booking subtype (tent, caravan, etc.) - Start and end dates - Initial booking preferences (electricity, number of people…)

Security : - Use Django forms for input validation. - Convert decimals and dates to safe formats before saving in session. - Validate capacity with Booking.check_capacity to prevent overbooking.

bookings.views.**booking_summary**(*request*)

Display a booking summary before payment. Validates session data and shows : - Total price - Deposit amount - Remaining balance

bookings.views.**booking_details**(*request*)

Collect customer's personal details. Once valid, create a Stripe Checkout session for deposit payment.

Security : - Never trust session directly, always validate with Django form. - Stripe key is read from Django settings.

bookings.views.**booking_confirm**(*request*)

Final step : - Verify data integrity - Save booking in DB - Send confirmation emails (admin + customer) - Clear session

Security : - Validate required fields before saving. - Remove all booking data from session after confirmation.

— Formulaires

class bookings.forms.**BookingFormClassic**(*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.utils.ErrorList'>*, *label_suffix=None*, *empty_permitted=False*, *instance=None*, *use_required_attribute=None*, *renderer=None*)

Bases : ModelForm

Client booking form.

Main fields :

- booking_type : campsite type (tent, caravan, camper, etc.)
- start_date / end_date : arrival and departure dates
- adults, children_over_8, children_under_8, pets : number of people and pets
- electricity : whether electricity is required
- cable_length : required if electricity is selected
- tent_width / tent_length, vehicle_length : specific dimensions depending on type

Security :

- Dates cannot be in the past.
- Conditional fields are validated based on campsite type.
- All data is cleaned via clean() method.

```
BOOKING_TYPE_CHOICES_FOR_FORM = [('tent', 'Tente'), ('car_tent', 'Voiture Tente'),
('caravan', 'Caravane'), ('fourgon', 'Fourgon'), ('van', 'Van'), ('camping_car',
'Camping-car')]

SUBTYPE_TO_MAIN_TYPE = {'camping_car': 'camping_car', 'car_tent': 'tent', 'caravan':
'caravan', 'fourgon': 'caravan', 'tent': 'tent', 'van': 'caravan'}

ELECTRICITY_CHOICES = [('yes', 'Avec électricité'), ('no', 'Sans électricité')]
```

class Meta

Bases : object

model

alias de *Booking*

```
fields = ['booking_type', 'vehicle_length', 'tent_width', 'tent_length',
'adults', 'children_over_8', 'children_under_8', 'pets', 'electricity',
'cable_length', 'start_date', 'end_date']

labels = {'cable_length': 'Longueur du câble électrique (m)', 'children_over_8':
'Enfants +8 ans', 'children_under_8': 'Enfants -8 ans', 'pets': 'Animaux',
'tent_length': 'Longueur de la tente (m)', 'tent_width': 'Largeur de la tente
(m)', 'vehicle_length': 'Longueur du véhicule (m)'}

widgets = {'cable_length': <django.forms.widgets.NumberInput object>,
'children_over_8': <django.forms.widgets.Select object>, 'children_under_8':
<django.forms.widgets.Select object>, 'pets': <django.forms.widgets.Select
object>, 'tent_length': <django.forms.widgets.NumberInput object>, 'tent_width':
<django.forms.widgets.NumberInput object>, 'vehicle_length':
<django.forms.widgets.NumberInput object>}
```

clean()

Custom validation :

- Arrival date cannot be in the past.
- Departure date must be after arrival.
- Conditional fields are required depending on campsite type and electricity.

base_fields = {'adults': <django.forms.fields.IntegerField object>, 'booking_type': <django.forms.fields.ChoiceField object>, 'cable_length': <django.forms.fields.DecimalField object>, 'children_over_8': <django.forms.fields.IntegerField object>, 'children_under_8': <django.forms.fields.IntegerField object>, 'electricity': <django.forms.fields.ChoiceField object>, 'end_date': <django.forms.fields.DateField object>, 'pets': <django.forms.fields.IntegerField object>, 'start_date': <django.forms.fields.DateField object>, 'tent_length': <django.forms.fields.DecimalField object>, 'tent_width': <django.forms.fields.DecimalField object>, 'vehicle_length': <django.forms.fields.DecimalField object>}

declared_fields = {'adults': <django.forms.fields.IntegerField object>, 'booking_type': <django.forms.fields.ChoiceField object>, 'electricity': <django.forms.fields.ChoiceField object>, 'end_date': <django.forms.fields.DateField object>, 'start_date': <django.forms.fields.DateField object>}

property media

Return all media required to render the widgets on this form.

class bookings.forms.**BookingDetailsForm**(*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.utils.ErrorList'>*, *label_suffix=None*, *empty_permitted=False*, *field_order=None*, *use_required_attribute=None*, *renderer=None*, *bound_field_class=None*)

Bases : Form

Client personal details form.

Security :

— Email field validated automatically.
— Maximum length for all text fields to prevent injection.
— All data cleaned via cleaned_data.

clean()

Centralized cleaning and validation : - Strips leading/trailing spaces for all text fields. - Normalizes email to lowercase. - Validates phone number contains only digits, spaces, +, -, ().

base_fields = {'address': <django.forms.fields.CharField object>, 'city': <django.forms.fields.CharField object>, 'email': <django.forms.fields.EmailField object>, 'first_name': <django.forms.fields.CharField object>, 'last_name': <django.forms.fields.CharField object>, 'phone': <django.forms.fields.CharField object>, 'postal_code': <django.forms.fields.CharField object>}

declared_fields = {'address': <django.forms.fields.CharField object>, 'city': <django.forms.fields.CharField object>, 'email': <django.forms.fields.EmailField object>, 'first_name': <django.forms.fields.CharField object>, 'last_name': <django.forms.fields.CharField object>, 'phone': <django.forms.fields.CharField object>, 'postal_code': <django.forms.fields.CharField object>}

property media

Return all media required to render the widgets on this form.

— Admin

class bookings.admin.**CapacityAdmin**(*model*, *admin_site*)

Bases : ModelAdmin

Admin for the Capacity model : displays booking types and maximum capacity.

```
list_display = ('booking_type', 'max_places', 'number_locations',
'number_mobile_homes')

list_filter = ('booking_type',)

search_fields = ('booking_type',)

fieldsets = (("Nombre d'emplacements par type", {'description': "Nombre maximum
d'emplacements disponibles pour chaque type de réservation.", 'fields':
('booking_type', 'max_places')}), ('Capacités totales du camping', {'fields':
('number_locations', 'number_mobile_homes')}))

property media
```

class bookings.admin.**SupplementPriceAdmin**(*model*, *admin_site*)

Bases : ModelAdmin

Admin for the SupplementPrice model : display and manage extra pricing.

```
list_display = ('extra_adult_price', 'child_over_8_price', 'child_under_8_price',
'pet_price', 'extra_vehicle_price', 'extra_tent_price',
'visitor_price_without_swimming_pool', 'visitor_price_with_swimming_pool')

search_fields = ()

fieldsets = (('Suppléments', {'description': "Tarifs des suppléments quelque soit la
saison et le type d'hébergement.", 'fields': ('extra_adult_price',
'child_over_8_price', 'child_under_8_price', 'pet_price', 'extra_vehicle_price',
'extra_tent_price', 'visitor_price_without_swimming_pool',
'visitor_price_with_swimming_pool')}),)

property media
```

class bookings.admin.**PriceAdminForm**(*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.utils.ErrorList'>*, *label_suffix=None*, *empty_permitted=False*, *instance=None*, *use_required_attribute=None*, *renderer=None*)

Bases : ModelForm

Custom form for PriceAdmin to add validation logic.

class **Meta**

Bases : object

model

alias de *Price*

```
fields = '__all__'
```

clean()

Hook for doing any extra form-wide cleaning after Field.clean() has been called on every field. Any ValidationError raised by this method will not be associated with a particular field ; it will have a special-case association with the field named "__all__".

base_fields = {'booking_type': <django.forms.fields.TypedChoiceField object>, 'is_worker': <django.forms.fields.BooleanField object>, 'price_1_person_with_electricity': <django.forms.fields.DecimalField object>, 'price_1_person_without_electricity': <django.forms.fields.DecimalField object>, 'price_2_persons_with_electricity': <django.forms.fields.DecimalField object>, 'price_2_persons_without_electricity': <django.forms.fields.DecimalField object>, 'season': <django.forms.fields.TypedChoiceField object>, 'supplements': <django.forms.models.ModelChoiceField object>, 'weekend_price_with_electricity': <django.forms.fields.DecimalField object>, 'weekend_price_without_electricity': <django.forms.fields.DecimalField object>, 'worker_week_price': <django.forms.fields.DecimalField object>}

declared_fields = {}

property media

Return all media required to render the widgets on this form.

class bookings.admin.**PriceAdmin**(*model*, *admin_site*)

Bases : ModelAdmin

Admin interface for Price model.
— Uses custom form PriceAdminForm to validate fields
— Displays pricing info for different booking types, seasons, and workers
— Organizes form fields into sections with descriptions
— List view shows both 1-person and 2-person prices, weekend and worker prices

form

alias de *PriceAdminForm*

list_display = ('booking_type', 'season', 'is_worker', 'price_1_person_with_electricity', 'price_2_persons_with_electricity', 'price_1_person_without_electricity', 'price_2_persons_without_electricity', 'worker_week_price', 'weekend_price_without_electricity', 'weekend_price_with_electricity')

list_filter = ('booking_type', 'season', 'is_worker')

search_fields = ('booking_type', 'season')

ordering = ('booking_type', 'season', 'is_worker')

exclude = ('included_people',)

fieldsets = (('Tarifs classiques', {'description': "Pour les tentes, caravanes et fourgons, saisir les tarifs 1 et 2 personnes.
Pour les camping-cars, le tarif est identique pour 1 ou 2 personnes.Merci de renseigner uniquement la ligne 2 personnes et laisser l'autre vide.", 'fields': ('booking_type', 'season', 'price_1_person_with_electricity', 'price_2_persons_with_electricity', 'price_1_person_without_electricity', 'price_2_persons_without_electricity')}), ('Tarifs ouvriers semaine', {'description': 'Prix spéciaux pour les ouvriers, électricité incluse.', 'fields': ('is_worker', 'worker_week_price')}), ('Tarifs ouvriers weekend', {'description': "Tarifs réduits le week-end quelque soit le type d'hébergement.", 'fields': ('weekend_price_without_electricity', 'weekend_price_with_electricity')}))

property media

class bookings.admin.**OtherPriceAdmin**(*model*, *admin_site*)

Bases : TranslatableAdmin

Admin for the OtherPrice model : manage current year and tourist tax.

list_display = ('current_year', 'tourist_tax_date', 'price_tourist_tax')

fieldsets = (('Année en cours', {'fields': ('current_year',)}), ('Taxe de séjour', {'fields': ('tourist_tax_date', 'price_tourist_tax')}))

property media

class bookings.admin.**BookingAdmin**(*model*, *admin_site*)

Bases : ModelAdmin

Admin interface for Booking model.
— Displays client info, booking details, capacities, and extra options
— Allows editing of deposit status directly from list view
— Provides filters by type, electricity, deposit, and dates
— Readonly fields : created_at_display, updated_at_display

list_display = ('last_name', 'first_name', 'start_date', 'end_date', 'booking_type', 'booking_subtype', 'electricity', 'deposit_paid', 'created_at_display', 'updated_at_display')

list_editable = ('deposit_paid',)

list_filter = ('booking_type', 'booking_subtype', 'electricity', 'deposit_paid', 'start_date', 'end_date')

search_fields = ('last_name', 'first_name', 'email', 'phone')

ordering = ('-created_at',)

readonly_fields = ('created_at_display', 'updated_at_display')

fieldsets = (('Informations client', {'fields': ('last_name', 'first_name', 'address', 'postal_code', 'city', 'phone', 'email')}), ('Détails de la réservation', {'fields': ('start_date', 'end_date', 'booking_type', 'booking_subtype', 'electricity', 'deposit_paid')}), ('Capacités et options', {'fields': ('adults', 'children_over_8', 'children_under_8', 'pets', 'extra_vehicle', 'extra_tent')}), ('Détails supplémentaires', {'description': "Ces champs apparaissent uniquement pour certains types d'hébergements.", 'fields': ('tent_length', 'tent_width', 'vehicle_length', 'cable_length')}), ('Dates de création et de mise à jour', {'fields': ('created_at_display', 'updated_at_display')}))

property media

class bookings.admin.**MobileHomeAdmin**(*model*, *admin_site*)

Bases : ModelAdmin

Admin for the MobileHome model : display pricing, information, and options.

list_display = ('name', 'night_price', 'night_price_mid', 'week_low', 'week_mid', 'week_high', 'is_worker_home')

search_fields = ('name',)

list_filter = ('is_worker_home',)

fieldsets = (('Informations générales', {'fields': ('name', 'description_text')}), ('Prix à la nuitée', {'description': 'Prix à la nuitée uniquement en basse et moyenne saison (si le prix moyenne saison diffère, utilisez night_price_mid).', 'fields': ('night_price', 'night_price_mid')}), ('Prix par semaine', {'fields': ('week_low', 'week_mid', 'week_high')}), ('Prix ouvriers', {'description': 'Uniquement pour le mobil-home ouvrier. Les prix sont fixes selon le nombre de personnes.', 'fields': ('is_worker_home', 'worker_price_1p', 'worker_price_2p', 'worker_price_3p')}))

property media

class bookings.admin.**SupplementMobileHomeAdmin**(*model*, *admin_site*)

Bases : TranslatableAdmin

Admin for the SupplementMobileHome model : manage deposits and linen rental.

list_display = ('mobile_home_deposit', 'cleaning_deposit', 'bed_linen_rental')

fieldsets = (('Cautions et location de linge', {'fields': ('mobile_home_deposit', 'cleaning_deposit', 'bed_linen_rental')}),)

property media

class bookings.admin.**SeasonInfoAdmin**(*model*, *admin_site*)

Bases : TranslatableAdmin

Admin for the SeasonInfo model : manage low, mid, and high season dates.

list_display = ['low_season_start', 'low_season_end', 'mid_season_start_1', 'mid_season_end_1', 'mid_season_start_2', 'mid_season_end_2', 'high_season_start', 'high_season_end']

fieldsets = (('Saisons', {'fields': ('low_season_start', 'low_season_end', 'mid_season_start_1', 'mid_season_end_1', 'mid_season_start_2', 'mid_season_end_2', 'high_season_start', 'high_season_end')}),)

property media

— Configuration App

class bookings.apps.**BookingsConfig**(*app_name*, *app_module*)

Bases : AppConfig

Configuration for the "bookings" app, which manages pricing and reservations.

default_auto_field = 'django.db.models.BigAutoField'

name = 'bookings'

verbose_name = 'Prix et Réservations'

1.6 Tests Bookings

— Modèles

bookings.tests.test_models.**test_supplementprice_creation**()

Test creation of SupplementPrice object and its string representation.

bookings.tests.test_models.**test_price_save_and_clean**()

Test saving a Price object, automatic field population, and validation rules.

bookings.tests.test_models.**test_booking_save_total_and_deposit**()

Test booking save, total price calculation, and deposit computation.

bookings.tests.test_models.**test_booking_capacity_validation**()

Test that capacity validation prevents overbooking.

bookings.tests.test_models.**test_capacity_str**()

Test the string representation of Capacity.

bookings.tests.test_models.**test_mobilehome_creation**()

Test MobileHome object creation and string representation.

bookings.tests.test_models.**test_supplementmobilehome_creation**()

Test SupplementMobileHome object creation and string representation.

bookings.tests.test_models.**test_seasoninfo_creation**()

Test SeasonInfo object creation and translation handling (French).

— Vues

bookings.tests.test_views.**valid_booking_data**()

Returns a dictionary of valid booking data for form submission tests.

bookings.tests.test_views.**client_details_data**()

Returns a dictionary of valid client personal details for form submission tests.

bookings.tests.test_views.**supplements**()

Creates and returns a SupplementPrice object for price calculation tests.

bookings.tests.test_views.**set_booking_session**(*client*, *booking_data*)

Stores booking data in client session for views that rely on session data.

bookings.tests.test_views.**test_booking_form_valid**(*mock_check_capacity*, *client*)

Submitting a valid booking form should redirect to booking_summary and call check_capacity.

bookings.tests.test_views.**test_booking_summary_displays_correct_prices**(*mock_deposit*, *mock_total*, *client*, *valid_booking_data*)

Booking summary view should correctly display total, deposit, and remaining balance.

bookings.tests.test_views.**test_booking_details_creates_stripe_session**(*mock_deposit*, *mock_stripe*, *client*, *valid_booking_data*, *client_details_data*, *supplements*)

Booking details view should create a Stripe session for deposit payment and redirect to Stripe.

bookings.tests.test_views.**test_booking_confirm_saves_booking_and_sends_emails**(*mock_send*, *client*, *valid_booking_data*, *client_details_data*)

Booking confirmation should save the booking, mark deposit as paid, send emails, and clear session.

— Formulaires

class bookings.tests.test_forms.**TestBookingFormClassic**

Bases : object

test_valid_data_tent(*mock_check_capacity*)

Valid data for tent type with electricity should pass

test_missing_required_field(*mock_check_capacity*)

Missing conditional field (cable_length) should raise error

test_past_start_date(*mock_check_capacity*)

Start date in the past should raise error

test_invalid_type_field(*mock_check_capacity*)

Invalid booking type should raise error

class bookings.tests.test_forms.**TestBookingDetailsForm**

Bases : object

test_valid_data()

All valid fields should pass and normalize email/strip spaces

test_invalid_phone()

Phone with invalid characters should raise error

test_missing_required_fields()

Required fields missing should raise errors

pytestmark = [Mark(name='django_db', args=(), kwargs={})]

Index des modules Python

b

c

r