Djangoのクラスベースビュー よくオーバーライドするメソッドとアトリビュート一覧

TemplateView

クラス用途

静的ページを表示する。

アトリビュート

名称	用途	使用例
template_name	テンプレート名を指定する。	template_name = ‘foo.html’

メソッド

名称	用途	使用例

get_context_data	テンプレートに辞書データ	def get_context_data(self):
		  context = super().get_context_data()
		  context['foo'] = FooModel.objects.get(user=self.request.user)
		  return context

	を渡す。	

ListView

クラス用途

モデルデータを一覧で表示する。

アトリビュート

名称	用途	使用例
template_name	テンプレート名を指定する。	template_name = ‘foo.html’
model	モデルを指定する。 querysetを設定していない場合は必須	model = FooModel
paginate_by	1ページに表示する件数を指定する。	paginate_by = 10
queryset	テンプレートにクエリーセットを渡す。	queryset = FooModel.objects.filter(foo=bar)

	(get_querysetと違い、いつも同じクエリーの時に使える) modelを設定しない場合は必須	

メソッド

名称	用途	使用例
get_context_data	テンプレートに辞書データを渡す。	def get_context_data(self): context = super().get_context_data() context['foo'] = FooModel.objects.get(user=self.request.user) return context
get_queryset	テンプレートにクエリーセットを渡す。 (querysetと違い、クエリーが都度変わる時に使える)	def get_queryset(self): foo = FooModel.objects.filter(user=self.request.user) return foo

FormView

クラス用途

汎用的にFormを扱う。

アトリビュート

名称	用途	使用例
template_name	テンプレート名を指定する。	template_name = ‘foo.html’
form_class	フォームクラス名を指定する。	form_class = FooForm
success_url	処理成功時にリダイレクトさせるURLを指定する。 (get_success_urlとの使い分けはこちらの記事に記載)	success_url = reverse_lazy(‘app:index’)

メソッド

名称	用途	使用例

get_context_data	テンプレートに辞書データ	def get_context_data(self):
		  context = super().get_context_data()
		  context['foo'] = FooModel.objects.get(user=self.request.user)
		  return context

	を渡す。	
form_valid	フォームバリデーションに問題がなかった時に行う処理を記述する。	def form_valid(self, form): 　messages.success(self.request, '成功しました') 　return super().form_valid(form)
form_invalid	フォームバリデーションに問題があった時に行う処理を記述する。	def form_invalid(self, form): 　message.error(self.request, "失敗しました") 　return super().form_invalid(form)

CreateView

クラス用途

モデルデータを作成する。

アトリビュート

名称	用途	使用例
template_name	テンプレート名を指定する。	template_name = ‘foo.html’
model	モデルを指定する。 querysetを設定していない場合は必須	model = FooModel
queryset	テンプレートにクエリーセットを渡す。 (get_querysetと違い、いつも同じクエリーの時に使える) modelを設定していない場合は必須	queryset = FooModel.objects.filter(foo=bar)
success_url	処理成功時にリダ	success_url = reverse_lazy(‘app:index’)

	イレクトさせるURLを指定する。 (get_success_urlとの使い分けはこちらの記事に記載)	
fields	ビューで使うフォームのフィールドを指定する。	fields = (‘field1’, field2,)

メソッド

名称	用途	使用例
get_context_data	テンプレートに辞書データを渡す。	def get_context_data(self): 　context = super().get_context_data() 　context['foo'] = FooModel.objects.get(user=self.request.user) 　return context
form_valid	フォームバリデーションに問題がなかった時に行	def form_valid(self, form): 　messages.success(self.request, '成功しました') 　return super().form_valid(form)

	う処理を記述する。	
form_invalid	フォームバリデーションに問題があった時に行う処理を記述する。	def form_invalid(self, form): 　message.error(self.request, "失敗しました") 　return super().form_invalid(form)

DetailView

クラス用途

モデルデータを表示する。

アトリビュート

名称	用途	使用例

template_name	テンプレート名を指定する。	template_name = 'foo.html'
model	モデルを指定する。 querysetを設定していない場合は必須	model = FooModel
queryset	テンプレートにクエリーセットを渡す。 (get_querysetと違い、いつも同じクエリーの時に使える) modelを設定していない場合は必須	queryset = FooModel.objects.filter(foo=bar)

メソッド

名称	用途	使用例

get_context_data	テンプレートに辞書データを渡す。	def get_context_data(self):
		  context = super().get_context_data()
		  context['foo'] = FooModel.objects.get(user=self.request.user)
		  return context

get_queryset	テンプレートにクエリーセ	def get_queryset(self):
		  foo = FooModel.objects.filter(user=self.request.user)
		  return foo

	ットを渡す。 (querysetと違い、クエリーが都度変わる時に使える)	

UpdateView

クラス用途

モデルデータを更新する。

アトリビュート

名称	用途	使用例
template_name	テンプレート名を指定する。	template_name = 'foo.html'
model	モデルを指定する。 querysetを設定していない場合は必須	model = FooModel
queryset	テンプレートにクエリーセットを渡	queryset = FooModel.objects.filter(foo=bar)

	す。 (get_querysetと違い、いつも同じクエリーの時に使える) modelを設定していない場合は必須	
success_url	処理成功時にリダイレクトさせるURLを指定する。 (get_success_urlとの使い分けはこちらの記事に記載)	success_url = reverse_lazy('app:index')
fields	ビューで使うフォームのフィールドを指定する。	fields = ('field1', field2,)

メソッド

名称	用途	使用例

get_context_data	テンプレートに辞書データを渡す。	def get_context_data(self):
		  context = super().get_context_data()
		  context['foo'] = FooModel.objects.get(user=self.request.user)
		  return context

form_valid	フォームバリデーションに問題がなかった時	def form_valid(self, form):
		  messages.success(self.request, '成功しました')
		  return super().form_valid(form)

	に行う処理を記述する。	

form_invalid	フォームバリデーションに問題があった時に行う処理を記述する。	def form_invalid(self, form):
		  message.error(self.request, "失敗しました")
		  return super().form_invalid(form)

get_success_url	処理成功時にリダイレクトさせるURLを指定する。 (success_urlとの使い分けはこちらの記事に記載)	def get_success_url(self): return reverse(‘app:detail’, kwargs={‘pk’: self.kwargs[‘pk’]})

DeleteView

クラス用途

モデルデータを削除する。

アトリビュート

名称	用途	使用例

template_name	テンプレート名を指定する。	template_name = ‘foo.html’
model	モデルを指定する。 querysetを設定していない場合は必須	model = FooModel
queryset	テンプレートにクエリーセットを渡す。 (get_querysetと違い、いつも同じクエリーの時に使える) modelを設定していない場合は必須	queryset = FooModel.objects.filter(foo=bar)
success_url	処理成功時にリダイレクトさせるURLを指定する。 (get_success_urlとの使い分けはこちらの記事に記載)	success_url = reverse_lazy(‘app:index’)

メソッド

名称	用途	使用例
get_context_data	テン	def get_context_data(self):

	プレートに辞書データを渡す。	context = super().get_context_data() context['foo'] = FooModel.objects.get(user=self.request.user) return context
delete	削除処理時に何か処理を追加する。	def delete(self, request, *args, **kwargs): messages.success(self.request, “削除しました。”) return super().delete(request, *args, **kwargs)