16 def has\_add\_permission(self, request):

"""

🔹 檢查當前使用者是否有權限「新增」(add) 一個物件。

- 預設回傳 True 表示允許新增。

- 可以在 `ModelAdmin` 的子類別中覆寫這個方法來自訂權限邏輯。

參數：

- request: `HttpRequest` 物件，代表當前請求。

回傳：

- True ➝ 使用者有新增權限

- False ➝ 使用者沒有新增權限

"""

# 取得當前模型的 Meta 資訊

opts = self.opts

# 取得 "add" (新增) 權限的代碼，例如 "app\_label.add\_modelname"

codename = get\_permission\_codename("add", opts)

# 檢查當前使用者是否擁有該權限

return request.user.has\_perm("%s.%s" % (opts.app\_label, codename))

16 def has\_change\_permission(self, request, obj=None):

"""

🔹 檢查當前使用者是否擁有 "修改" (change) 權限。

- 若 `obj=None`，則表示檢查是否能修改「任何」該類型的資料。

- 若 `obj` 不是 None，則可能需要進一步自訂權限來限制特定物件的修改權限。

參數：

- request: `HttpRequest` 物件，代表當前的請求。

- obj: 要檢查權限的特定物件 (如果為 `None`，則代表所有該類型的物件)。

回傳：

- True ➝ 使用者有修改權限

- False ➝ 使用者沒有修改權限

"""

# 取得當前 ModelAdmin 所管理的模型 (model) 的 Meta 資訊

opts = self.opts

# 取得 "change" (修改) 權限的代碼，例如 "app\_label.change\_modelname"

codename = get\_permission\_codename("change", opts)

# 檢查當前使用者是否擁有該權限

return request.user.has\_perm("%s.%s" % (opts.app\_label, codename))

17 def has\_delete\_permission(self, request, obj=None):

"""

🔹 檢查當前使用者是否擁有 "刪除" (delete) 權限。

- 若 `obj=None`，則表示檢查是否能刪除「任何」該類型的資料。

- 若 `obj` 不是 None，則可能需要進一步自訂權限來限制特定物件的刪除權限。

參數：

- request: `HttpRequest` 物件，代表當前的請求。

- obj: 要檢查權限的特定物件 (如果為 `None`，則代表所有該類型的物件)。

回傳：

- True ➝ 使用者有刪除權限

- False ➝ 使用者沒有刪除權限

"""

# 取得當前 ModelAdmin 所管理的模型 (model) 的 Meta 資訊

opts = self.opts

# 取得 "delete" (刪除) 權限的代碼，例如 "app\_label.delete\_modelname"

codename = get\_permission\_codename("delete", opts)

# 檢查當前使用者是否擁有該權限

return request.user.has\_perm("%s.%s" % (opts.app\_label, codename))

17 def has\_view\_permission(self, request, obj=None):

"""

🔹 檢查當前使用者是否擁有 "查看" (view) 權限。

- Django 2.1 之後，"view" 權限成為獨立權限 (以前擁有 "change" 權限即可查看)。

- 若 `obj=None`，則表示檢查是否能查看「任何」該類型的資料。

- 若 `obj` 不是 None，則可能需要進一步自訂權限來限制特定物件的查看權限。

參數：

- request: `HttpRequest` 物件，代表當前的請求。

- obj: 要檢查權限的特定物件 (如果為 `None`，則代表所有該類型的物件)。

回傳：

- True ➝ 使用者有查看權限

- False ➝ 使用者沒有查看權限

"""

# 取得當前 ModelAdmin 所管理的模型 (model) 的 Meta 資訊

opts = self.opts

# 取得 "view" (查看) 權限的代碼，例如 "app\_label.view\_modelname"

codename\_view = get\_permission\_codename("view", opts)

# 取得 "change" (修改) 權限的代碼，例如 "app\_label.change\_modelname"

codename\_change = get\_permission\_codename("change", opts)

# 檢查當前使用者是否擁有 "view" 或 "change" 權限 (有修改權限的話，預設也能查看)

return request.user.has\_perm("%s.%s" % (opts.app\_label, codename\_view)) or \

request.user.has\_perm("%s.%s" % (opts.app\_label, codename\_change))

36 def save\_model(self, request, obj, form, change):

"""

🔹 負責儲存 Django Admin 的模型 (Model) 物件。

- `change=True` 代表是 \*\*修改 (Update)\*\* 已存在的物件。

- `change=False` 代表是 \*\*新增 (Create)\*\* 新的物件。

參數：

- request: `HttpRequest` 物件，代表當前的請求。

- obj: 目前要儲存的模型物件。

- form: Admin 表單 (`ModelForm`) 物件，包含使用者輸入的資料。

- change: Boolean 值，`True` 代表是修改，`False` 代表是新增。

🚀 這個方法可以在 `ModelAdmin` 的子類別中覆寫來自訂儲存邏輯。

"""

# 使用 Django ORM 的 `save()` 方法，將物件儲存到資料庫

obj.save()

36 def delete\_model(self, request, obj):

"""

🔹 刪除 Django Admin 中的單一模型實例 (單筆資料)。

參數：

- request: `HttpRequest` 物件，代表當前的請求。

- obj: 目前要刪除的模型物件。

Django 會自動呼叫這個方法來刪除物件。

🚀 這個方法可以在 `ModelAdmin` 子類別中覆寫來自訂刪除邏輯。

"""

# 呼叫 Django ORM 的 `delete()` 方法來刪除物件

obj.delete()

36 def delete\_queryset(self, request, queryset):

"""

🔹 在 Django Admin 中執行「批量刪除」操作 (刪除多筆資料)。

- 當管理員在 Django Admin 的「變更列表」(changelist) 中選擇多個物件並執行「刪除」時，這個方法會被調用。

- 預設情況下，這個方法會對 `queryset` 呼叫 `.delete()`，刪除所有選取的物件。

參數：

- request: `HttpRequest` 物件，代表當前的請求。

- queryset: 包含要刪除物件的 QuerySet。

🚀 這個方法可以在 `ModelAdmin` 子類別中覆寫來自訂批量刪除的邏輯，例如記錄刪除日誌或限制刪除條件。

"""

# 執行批量刪除，將 queryset 中的所有物件從資料庫刪除

queryset.delete()

38 def response\_add(self, request, obj, post\_url\_continue=None):

"""

🔹 新增物件後的回應 (決定跳轉位置)

當管理員在 Django Admin 介面新增一個物件後，這個方法會決定下一步該導向哪個頁面。

可能的跳轉位置：

- 📝 編輯該物件的詳細頁面 (如果選擇 "Save and continue editing")

- ➕ 新增新物件的頁面 (如果選擇 "Save and add another")

- 📜 物件列表頁面 (如果選擇 "Save")

參數：

- request: `HttpRequest` 物件，代表當前的請求。

- obj: 剛剛新增的模型物件。

- post\_url\_continue: (可選) 決定 "Save and continue editing" 時的 URL。

回傳：

- `HttpResponseRedirect` ➝ 依照管理員選擇的按鈕決定跳轉位置。

"""

# 設定成功訊息，顯示「The <model\_name> "<object>" was added successfully.」

msg = \_("The {name} “{obj}” was added successfully.") # 訊息模板

self.message\_user(

request,

format\_html(msg, name=self.opts.verbose\_name, obj=obj), # 格式化訊息

messages.SUCCESS # 設定訊息類型為 "成功"

)

# 呼叫 `response\_post\_save\_add` 來決定要跳轉到哪個頁面

return self.response\_post\_save\_add(request, obj)

41 def response\_change(self, request, obj):

"""

🔹 處理 Django Admin 物件修改後的回應 (決定跳轉位置)

當管理員在 Django Admin 介面修改物件後，這個方法會決定下一步該導向哪個頁面。

可能的跳轉位置：

- 📝 繼續編輯該物件 (如果選擇 "Save and continue editing")

- 📜 返回物件列表頁 (如果選擇 "Save")

- ➕ 新增新物件的頁面 (如果選擇 "Save and add another")

參數：

- request: `HttpRequest` 物件，代表當前的請求。

- obj: 剛剛修改的模型物件。

回傳：

- `HttpResponseRedirect` ➝ 依照管理員選擇的按鈕決定跳轉位置。

"""

# 設定成功訊息，顯示「The <model\_name> "<object>" was changed successfully.」

msg = \_("The {name} “{obj}” was changed successfully.") # 訊息模板

self.message\_user(

request,

format\_html(msg, name=self.opts.verbose\_name, obj=obj), # 格式化訊息

messages.SUCCESS # 設定訊息類型為 "成功"

)

# 呼叫 `response\_post\_save\_change` 來決定要跳轉到哪個頁面

return self.response\_post\_save\_change(request, obj)

46 def response\_delete(self, request, obj\_display, obj\_id):

"""

🔹 處理 Django Admin 物件刪除後的回應 (決定跳轉位置)

當管理員在 Django Admin 介面刪除物件後，這個方法會決定下一步該導向哪個頁面。

預設情況下：

- 📜 刪除物件後，回到該模型的列表頁 (`changelist` 頁面)。

參數：

- request: `HttpRequest` 物件，代表當前的請求。

- obj\_display: 被刪除物件的「顯示名稱」，通常是 `\_\_str\_\_()` 的回傳值。

- obj\_id: 被刪除物件的 ID。

回傳：

- `HttpResponseRedirect` ➝ 導向該模型的列表頁 (`changelist`)。

"""

# 設定成功訊息，顯示「The <model\_name> "<object>" was deleted successfully.」

msg = \_("The %(name)s “%(obj)s” was deleted successfully.") % {

"name": self.opts.verbose\_name, # 取得模型的 `verbose\_name`

"obj": obj\_display # 取得被刪除物件的顯示名稱

}

# 在 Django Admin 介面顯示成功訊息

self.message\_user(request, msg, messages.SUCCESS)

# 刪除後，跳轉到該模型的列表頁 (`changelist`)

return HttpResponseRedirect(reverse("admin:%s\_%s\_changelist" % (self.opts.app\_label, self.opts.model\_name)))