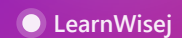


Boîtes de dialogue, validation et workflows

Boîtes de dialogue et workflows modaux · Validation et modèle

Enregistrer/Annuler



Boîtes de dialogue et workflows modaux

Ce qu'est une boîte de dialogue, quand la préférer à une page, le workflow modal en cinq étapes et l'utilité des boîtes de message.



Objectif de la leçon

Comprendre les **boîtes de dialogue** et le workflow modal d'ajout/modification sur lequel reposent les applications métier : ouvrir une fenêtre ciblée, laisser l'utilisateur accomplir une seule tâche, confirmer les actions risquées et renvoyer un résultat clair.

1. Qu'est-ce qu'une boîte de dialogue ?

Une **boîte de dialogue** est un écran plus petit qui apparaît par-dessus la page principale pour que l'utilisateur accomplisse une tâche précise. Dans ce cours, les boîtes de dialogue les plus courantes sont **Create Ticket** et **Edit Ticket**. Une boîte de dialogue ne doit pas chercher à être toute l'application — elle recueille les valeurs nécessaires, les valide et renvoie un résultat clair.



Idée clé

Une boîte de dialogue est *modale* : l'utilisateur la termine ou l'annule avant de revenir à la page qui se trouve derrière. C'est cette focalisation qui rend les workflows d'ajout/modification prévisibles.

2. Boîte de dialogue ou page

Type d'écran	Meilleur usage	Exemple
Page / zone de contenu principale	Les grands écrans de l'application où les utilisateurs passent du temps et naviguent entre les fonctionnalités.	Page Tickets, page Customers, page Settings.
Boîte de dialogue / fenêtre modale	Un petit workflow ciblé qui doit être terminé ou annulé avant de revenir en arrière.	Boîte de dialogue Create Ticket, boîte de dialogue Edit Ticket, message Confirm Delete.

3. Le pattern courant du workflow modal

Presque toutes les fonctionnalités d'ajout/modification suivent le même flux en cinq étapes. S'y tenir rend le workflow lisible et prévisible.

Clic sur un bouton → l'utilisateur choisit New ou Edit
Ouvrir la boîte → afficher le formulaire / la fenêtre modale
Valider → vérifier les champs obligatoires
Enregistrer → appeler la couche de service
Rafraîchir l'interface → mettre à jour la grille et afficher un retour

L'utilisateur clique sur **New** ou **Edit**, l'application ouvre une boîte de dialogue, celle-ci valide les champs, le bouton **Save** appelle un service et la page principale se rafraîchit *après* la fermeture réussie de la boîte de dialogue.

4. Boîtes de message et confirmations

Utilisez des boîtes de message quand l'utilisateur a besoin d'un retour immédiat ou doit confirmer une action risquée.

Situation	Retour recommandé
Enregistrement terminé	Afficher un court message de réussite et rafraîchir la liste des tickets.
Champ obligatoire manquant	Afficher un message de validation clair près du champ ou en haut de la boîte de dialogue.
Suppression du ticket sélectionné	Demander une confirmation avant de supprimer.
Annulation après modification de valeurs	Facultatif : avertir que les modifications non enregistrées seront perdues.



Vous êtes prêt pour la leçon suivante

Vous savez décrire ce qu'est une boîte de dialogue, quand l'utiliser plutôt qu'une page, le workflow modal en cinq étapes et l'utilité des boîtes de message. Ensuite : la validation des saisies et le pattern Save/Cancel cohérent.

Validation et modèle Enregistrer/Annuler

Validez au plus près de l'interface sans y enfermer les règles métier, écrivez un gestionnaire Save lisible et appliquez un pattern Save/Cancel cohérent.



Objectif de la leçon

Valider les saisies au plus près de l'interface sans y enfermer les règles métier, écrire un gestionnaire Save lisible et appliquer un pattern Save/Cancel cohérent pour que les utilisateurs sachent toujours ce qui s'est passé.

1. Validation : proche de l'interface, sans y être enfermée

La validation doit être assez proche de l'interface pour que les utilisateurs obtiennent un retour immédiat, mais les règles métier importantes doivent rester testables en dehors du clic sur le bouton. La boîte de dialogue vérifie les problèmes simples de champs, puis appelle une classe de service pour les règles réutilisables.

Type de validation	Exemple	Bon emplacement
Validation simple de l'interface	La zone de texte du titre est vide ; la priorité n'est pas sélectionnée.	Une méthode de la boîte de dialogue comme <code>ValidateForm()</code> .
Règle métier	Un ticket fermé ne peut pas être modifié par un agent de support.	<code>TicketService</code> ou un service de validation.
Règle de persistance	L'ID du ticket existe déjà dans le stockage.	La couche repository / service.

2. Gestion sûre des événements

Les événements de clic sur les boutons doivent rester lisibles. Ne cachez pas d'appels lents à la base de données ni de logique métier compliquée dans un événement d'interface anodin, sauf si c'est voulu et clairement nommé. Un bon événement Save appelle de petites méthodes utilitaires comme `ValidateForm()`, `BuildTicketFromFields()` et `ticketService.SaveTicket(ticket)`.

```
private void btnSave_Click(object sender, EventArgs e)
{
    if (!ValidateForm())
        return;

    Ticket ticket = BuildTicketFromFields();
    ticketService.SaveTicket(ticket);

    this.DialogResult = DialogResult.OK;
    this.Close();
}
```



Validez avant d'enregistrer

Le gestionnaire sort immédiatement quand `ValidateForm()` échoue — rien n'est enregistré et la boîte de dialogue reste ouverte. Seules les saisies valides atteignent le service.

3. Un pattern Save / Cancel cohérent

- **Save** valide d'abord, appelle ensuite le service, puis ferme la boîte de dialogue *uniquement en cas de réussite*.
- **Cancel** ferme la boîte de dialogue sans enregistrer.
- Les messages d'erreur doivent indiquer précisément à l'utilisateur ce qu'il doit corriger.
- La page parente ne rafraîchit la grille que lorsque la boîte de dialogue renvoie `OK`.
- Gardez des noms de boutons clairs : `btnSave` , `btnCancel` , `btnNew` , `btnEdit` .

4. Habitudes de développeur pour ce module

- Nommez clairement les contrôles pour que le code des événements soit lisible.
- Gardez la mise en page de la boîte de dialogue simple et ciblée.
- Séparez l'état de l'interface, les règles métier et les données enregistrées.
- Utilisez des services pour les opérations réutilisables sur les tickets.
- Utilisez les boîtes de message et les messages de validation de manière cohérente.



Vous êtes prêt pour le lab

Ensuite : regardez la vidéo pas à pas sur les boîtes de dialogue, réussissez la vérification des connaissances, puis construisez des boîtes de dialogue Create et Edit Ticket réutilisables — valider d'abord, enregistrer via le service, renvoyer `DialogResult.OK` et rafraîchir la grille avec un retour clair.