

FlowLayoutPanel, TableLayoutPanel, FlexLayoutPanel et propriétés de mise en page étendues

Guide de la leçon · Guide du lab / de l'examen

● LearnWisej

Guide de la leçon

Choisir un conteneur flow, table ou flex selon le comportement du contenu, les propriétés enfants étendues FillWeight, FlowBreak, AlignX, AlignY, Row, Column, les fusions et les styles, les cas où les moteurs de mise en page respectent les marges, et les extensions de markup fluent pour les mises en page construites par code. Ce que cela signifie pour un développeur Wisej.NET — et comment aborder ce module.



Enjeu en production

Choisir un conteneur flow, table ou flex selon le comportement du contenu, les propriétés enfants étendues FillWeight, FlowBreak, AlignX, AlignY, Row, Column, les fusions et les styles, les cas où les moteurs de mise en page respectent les marges, et les extensions de markup fluent pour les mises en page construites par code.

1. Le problème en production

Le Module 4 a doté Adaptive Operations Console d'un shell de régions ancrées, mais le docking décide seulement de l'emplacement d'une région, pas du comportement de son contenu quand le navigateur fait 1400 pixels de large le lundi et 700 sur une tablette le mardi. Une rangée de cartes de métriques doit passer à la ligne, un éditeur de ticket doit garder ses labels et ses éditeurs alignés quelle que soit la longueur de la traduction allemande, et la liste et les détails doivent se partager l'espace de travail dans la proportion que vous décidez. Chacun de ces cas est un comportement de contenu différent, et Wisej.NET fournit un conteneur pour chacun : `FlowLayoutPanel`, `TableLayoutPanel` et `FlexLayoutPanel`. Choisir le mauvais, c'est ainsi que les équipes se remettent à écrire du code de redimensionnement.

2. Concepts clés

Concept	Pourquoi c'est important
FlowLayoutPanel	Dispose les enfants en lignes ou colonnes automatiques selon <code>FlowDirection</code> , passe à la ligne quand <code>WrapContents</code> est à true et ignore Location, Dock et Anchor de chaque enfant, si bien que pastilles et cartes se réorganisent d'elles-mêmes.
TableLayoutPanel	Place les enfants dans des cellules adressées par Row et Column, avec des fusions pour les cellules qui s'étendent, si bien que labels et éditeurs restent alignés tandis que chaque enfant peut encore utiliser Dock ou Anchor dans sa cellule.
FlexLayoutPanel	Dispose les enfants sur une ligne ou une colonne, remplit toujours sa zone cliente et partage l'espace restant par poids, ce qui convient aux grandes régions qui divisent l'espace de travail proportionnellement.
FillWeight	Une propriété enfant étendue lue par les moteurs flow et flex : un poids supérieur à zéro étire l'enfant dans l'espace restant, proportionnellement aux autres enfants pondérés ; zéro conserve sa propre taille.
FlowBreak	Une propriété enfant étendue de FlowLayoutPanel qui force l'enfant suivant sur une nouvelle ligne ou colonne, pour qu'une barre de filtres termine une ligne délibérément au lieu de dépendre de la largeur.
Styles de lignes et de colonnes	<code>RowStyles</code> et <code>ColumnStyles</code> déclarent si une ligne ou une colonne s'ajuste automatiquement, prend une taille fixe en pixels ou partage le reste en pourcentage ; c'est là que vivent les proportions d'un formulaire.
Margin et Padding	Padding est l'espace intérieur du conteneur autour de tous les enfants ; Margin est la distance d'un enfant à ses voisins, et seuls les moteurs qui respectent les marges l'utilisent.

3. Déroulé de la leçon

Choisir le conteneur selon le comportement du contenu

La règle de décision est assez courte pour être retenue et assez stricte pour servir de référence en revue. Utilisez `FlowLayoutPanel` pour le contenu qui peut passer à la ligne : pastilles de filtres, cartes de métriques, zones de débordement de barre d'outils, ensembles de commandes responsives. Utilisez `TableLayoutPanel` pour les formulaires structurés : labels de champs, éditeurs alignés, lignes de synthèse et toute grille de contrôles qui doit survivre à la localisation. Utilisez `FlexLayoutPanel` pour les grandes régions qui doivent remplir l'espace disponible proportionnellement, comme un rail de navigation à côté d'une région de contenu et d'un panneau de détails latéral, ou des sections mobiles empilées qui se partagent la hauteur par poids. Le docking et l'ancrage restent adaptés aux régions stables du shell et aux formulaires simples ; les trois conteneurs servent au contenu de ces régions.

Regardez le comportement, pas l'aspect. La zone des cartes de métriques de la console ressemble à une grille, mais son comportement est « autant de cartes par ligne que possible, puis retour à la ligne » : c'est du flow. L'éditeur de ticket ressemble à une liste de champs, mais son comportement est « le bord gauche de chaque éditeur sur la même ligne » : c'est une table. L'espace de travail ressemble à deux panneaux,

mais son comportement est « la liste prend deux tiers, les détails un tiers, et aucun ne s'effondre » : c'est du flex. Le lab construit les trois pour que vous puissiez redimensionner le navigateur et observer quel conteneur résiste à quel scénario ; cette comparaison écrite est le livrable qui compte le plus.

Les propriétés enfants étendues

Les conteneurs fonctionnent grâce à des propriétés étendues : des valeurs qui appartiennent à l'enfant mais sont stockées et lues par le parent, et qui n'apparaissent dans la grille des propriétés de l'enfant que tant qu'il se trouve dans ce conteneur. `FlowLayoutPanel` lit `Margin`, `FillWeight`, `MinimumSize`, `MaximumSize` et `FlowBreak`, définis en code via le conteneur : `SetFillWeight(search, 1)` permet au champ de recherche d'utiliser toute la largeur restante de la ligne, et `SetFlowBreak(apply, true)` termine la ligne pour que les cartes en dessous commencent sur une nouvelle ligne. `FlowDirection` choisit gauche à droite, droite à gauche, haut en bas ou bas en haut ; avec `WrapContents` à `false`, les enfants sont rognés ou défilent au lieu de passer à la ligne, ce que souhaite une barre d'outils sur une seule ligne avec `AutoScroll`.

`TableLayoutPanel` lit `Row`, `Column`, `RowSpan` et `ColumnSpan` de chaque enfant et dimensionne la grille à partir de `RowStyles` et `ColumnStyles` : ajustement automatique, valeur absolue en pixels ou pourcentage. L'éditeur de ticket utilise une colonne de labels absolue de 140 pixels et une colonne d'éditeurs à 100 %, si bien que les éditeurs s'élargissent avec le panneau tandis que les labels restent en place, et l'éditeur Notes s'étend sur les deux colonnes dans une ligne qui prend la hauteur restante. `GrowStyle` décide de ce qui se passe quand un contrôle est ajouté une fois les cellules pleines : ajouter des lignes, ajouter des colonnes ou refuser de s'agrandir. Comme un enfant peut encore être ancré dans sa cellule, un éditeur ancré en `Fill` dans une colonne en pourcentage est le formulaire aligné le plus propre que produit Wisej.NET.

`FlexLayoutPanel` lit `Margin`, `MinimumSize`, `MaximumSize`, `FillWeight`, `AlignX` et `AlignY`. Son `LayoutStyle` est horizontal (une ligne) ou vertical (une colonne) ; le conteneur remplit toujours sa zone cliente et répartit l'espace restant entre les enfants pondérés. Quand un enfant ne remplit pas une dimension, `AlignX` et `AlignY` remplacent l'alignement du parent pour cet enfant seul, si bien qu'une région de détails avec `AlignY` en haut se colle au bord supérieur d'un espace de travail haut au lieu de se centrer. La règle vérifiée par le lab : chaque enfant pondéré reçoit un `MinimumSize`, sinon un navigateur étroit réduit une région à zéro et l'utilisateur voit une bande vide là où se trouvait l'éditeur de détails. `MaximumSize` plafonne la même région pour qu'un moniteur 4K n'ait pas un formulaire en un océan d'espace vide.

```
// DashboardWorkspace : FlexLayoutPanel – the list/details split
this.Dock = DockStyle.Fill;
this.LayoutStyle = FlexLayoutStyle.Horizontal;
this.Padding = new Padding(8);

var list = new Panel { MinimumSize = new Size(300, 200) };
var details = new Panel { MinimumSize = new Size(280, 200),
                          MaximumSize = new Size(520, 0) };

this.Controls.Add(list);
this.Controls.Add(details);

this.SetFillWeight(list, 2); // two thirds of the spare width
this.SetFillWeight(details, 1); // one third, never below 280 px
this.SetAlignY(details, VerticalAlignment.Top);
```

Quand les moteurs de mise en page respectent les marges

Margin est la propriété la plus souvent définie et la moins souvent respectée, car son effet dépend du moteur qui dispose l'enfant. Les moteurs flow et flex appliquent les marges : chaque enfant d'un `FlowLayoutPanel` ou d'un `FlexLayoutPanel` conserve la distance déclarée par sa Margin. Le moteur de mise en page par défaut n'utilise pas les marges pour le docking ; un Panel ancré en Left avec une Margin de 8 reste collé au bord malgré tout, c'est pourquoi le Module 4 a utilisé le Padding sur les régions ancrées pour créer les espacements. `TableLayoutPanel` tient compte des marges à l'intérieur de la cellule : la Margin d'un enfant devient sa distance à la bordure de la cellule, tandis que le Padding de la table ajoute le cadre extérieur.

Considérez que les deux propriétés appartiennent à des propriétaires différents : Padding est la décision du conteneur sur l'espace intérieur autour de tous ses enfants, Margin la décision de l'enfant sur sa distance aux contrôles adjacents. Un système de mise en page cohérent pour la console définit des valeurs standard une seule fois, par exemple des modes dense, normal et aéré de 4, 8 et 16 pixels, et les applique partout via ces deux propriétés. Quand un espacement n'apparaît pas, demandez-vous quel moteur possède l'enfant avant de changer le moindre nombre.

Extensions de markup fluent

L'espace de noms `Wisej.Web.Markup` ajoute des méthodes d'extension fluent qui se lisent comme une description de mise en page. `FlowLayoutPanelExtensions` permet de définir FillWeight et FlowBreak, `FlexLayoutPanelExtensions` FillWeight, AlignX et AlignY, et `TableLayoutPanelExtensions` le nombre de colonnes, le GrowStyle et les réglages de disposition. Chaque méthode renvoie le conteneur, si bien qu'un shell construit par code devient une seule chaîne : `panel.FillWeight(nav, 0).FillWeight(content, 1).AlignY(nav, VerticalAlignment.Top)`. Le lab vous demande de construire une région de cette façon ; le candidat naturel est une région que vous créez déjà en code, comme un tableau de bord dont les cartes dépendent du rôle de l'utilisateur connecté.

Utilisez le style fluent là où il rend lisible une mise en page construite par code, et laissez les écrans conçus dans le designer au designer. Mélanger dans une même classe la sérialisation du designer, des appels impératifs à `SetFillWeight` et des chaînes fluent oblige le développeur suivant à lire trois notations pour un seul panneau ; gardez donc une seule notation par conteneur. Dans tous les cas, la checklist s'applique :

des conteneurs plutôt que des gestionnaires de redimensionnement, l'ordre de docking vérifié, du Padding sur les régions ancrées, flow pour le retour à la ligne, table pour les formulaires, flex pour les régions proportionnelles, et un MinimumSize sur chaque contrôle pondéré par FillWeight.



Pièges courants

Définir Location, Dock ou Anchor sur un enfant de FlowLayoutPanel et se demander pourquoi c'est ignoré. Des régions pondérées sans MinimumSize qui s'effondrent sur téléphone. S'attendre à ce que Margin crée des espacements entre des Panels ancrés sous le moteur de mise en page par défaut. Choisir FlexLayoutPanel pour un formulaire aux labels alignés, ou TableLayoutPanel pour une rangée de cartes qui doit passer à la ligne.



Réflexe du système visuel

Utilisez le mécanisme Wisej.NET qui porte le problème : le thème pour l'identité des contrôles standard, AppearanceKey pour les variantes sémantiques, CssClass pour le CSS ciblé propre à l'application, CssStyle uniquement pour les valeurs dynamiques ponctuelles, les conteneurs de mise en page plutôt que du code de redimensionnement, les propriétés responsives pour les valeurs de profil définies dans le designer, et ResponsiveProfileChanged pour le comportement qui n'est pas simplement visuel.



Objectifs d'apprentissage

À la fin de ce module, vous serez capable de :

- ✓ Choisir FlowLayoutPanel, TableLayoutPanel ou FlexLayoutPanel selon le comportement du contenu et la finalité de l'écran.
- ✓ Utiliser FillWeight, FlowBreak, AlignX, AlignY, Row, Column, RowSpan, ColumnSpan, RowStyles, ColumnStyles, GrowStyle, MinimumSize et MaximumSize.
- ✓ Comprendre quand les moteurs de mise en page utilisent les marges et quand ils ne les utilisent pas.
- ✓ Utiliser les méthodes d'extension de markup pour des définitions de mise en page fluente, lorsque c'est approprié.

Guide du lab / de l'examen

Ce que vous allez construire dans le lab pratique, l'approche suggérée et les livrables attendus.



Lab pratique

Construisez trois versions de la zone de contenu d'Adaptive Operations Console : une zone de filtres et de cartes de métriques en `FlowLayoutPanel` qui passe à la ligne et utilise `FillWeight` et `FlowBreak`, un éditeur de ticket en `TableLayoutPanel` dont les labels et les éditeurs sont alignés grâce à `RowStyles` et `ColumnStyles`, et un shell de tableau de bord en `FlexLayoutPanel` où la liste et les détails se partagent l'espace par `FillWeight`, avec `AlignX`, `AlignY`, `MinimumSize` et `MaximumSize` définis. Construisez l'une des trois en code avec les extensions fluent de `Wisej.Web.Markup` et comparez la façon dont chacune résiste au redimensionnement du navigateur.

Point de départ

Ouvrez la solution `AdaptiveOperationsConsole` telle que vous l'avez laissée après le Module 4 : la Page principale est un shell de conteneurs imbriqués avec une barre d'outils, un rail de navigation, un espace de travail en remplissage, un panneau de détails et une région d'état, les régions de navigation et de détails sont des `UserControls`, et l'espace de travail est encore un simple `Panel` dont les cartes de métriques, l'éditeur de ticket et la liste sont positionnés à la main. Rangez les trois nouvelles régions dans un dossier `Layout` pour que le projet de synthèse puisse retenir le gagnant de chaque comparaison.

Approche suggérée

- ✓ Créez une `FilterBar` qui dérive de `FlowLayoutPanel` avec `FlowDirection` de gauche à droite et `WrapContents` activé ; ajoutez le champ de recherche, la liste déroulante de statut et le bouton Apply, donnez au champ de recherche un `FillWeight` de 1 et un `MinimumSize`, définissez `FlowBreak` sur Apply, puis placez les cartes de métriques en dessous et observez-les passer à la ligne.
- ✓ Construisez un `UserControl` `TicketEditor` autour d'un `TableLayoutPanel` à deux colonnes : une colonne de labels absolue de 140 pixels, une colonne d'éditeurs à 100 %, une ligne à ajustement automatique par champ, une ligne Notes qui s'étend sur les deux colonnes avec la hauteur restante, et chaque éditeur ancré en Fill dans sa cellule.
- ✓ Remplacez le `Panel` de l'espace de travail par un `DashboardWorkspace` qui dérive de `FlexLayoutPanel` en disposition horizontale ; ajoutez la liste et les détails avec des `FillWeight` de 2 et 1, un `MinimumSize` sur les deux, un `MaximumSize` sur les détails et `AlignY` en haut sur les détails.
- ✓ Construisez l'une des trois régions entièrement en code avec `using Wisej.Web.Markup;` et les méthodes d'extension fluent `FillWeight`, `FlowBreak`, `AlignX` et `AlignY` enchaînées sur le conteneur, et gardez cette classe exempte de sérialisation du designer.
- ✓ Exécutez la console, redimensionnez le navigateur de 1400 à 1024 puis à 700 pixels pour chaque région, prenez une capture à chaque largeur et notez, pour chaque région, quel conteneur a tenu et quelle propriété étendue a fait le travail.

Livrables attendus

- ✓ Zone de filtres et de cartes de métriques en FlowLayoutPanel utilisant WrapContents, FillWeight et FlowBreak
- ✓ Éditeur de ticket en TableLayoutPanel avec RowStyles, ColumnStyles et fusions pour des champs alignés
- ✓ Shell de tableau de bord en FlexLayoutPanel avec FillWeight, AlignX, AlignY, MinimumSize et MaximumSize
- ✓ Une région construite en code avec les extensions fluent de Wisej.Web.Markup
- ✓ Captures comparatives au redimensionnement et note indiquant quel conteneur convient à chaque région

Questions de révision

- ✓ Vous avez donné un FillWeight de 1 à la région de détails mais aucun MinimumSize, et à 700 pixels elle a disparu. Expliquez ce que le moteur flex a fait de l'espace et pourquoi la correction est une contrainte de taille plutôt qu'un autre poids.
- ✓ Le bouton Apply passe à la deuxième ligne à 1024 pixels alors qu'il y a de la place pour lui. Quelle propriété étendue en est la cause, et comment la FilterBar se comporterait-elle si vous la retiriez et mettiez WrapContents à false ?
- ✓ Une Margin de 8 sur les cartes du FlowLayoutPanel a créé des espacements, mais la même Margin sur le rail de navigation ancré n'a rien fait. Quel moteur respecte les marges dans chaque cas, et quelle propriété crée l'espacement pour le rail ?



Réflexe du système visuel

Utilisez le mécanisme Wisej.NET qui porte le problème : le thème pour l'identité des contrôles standard, AppearanceKey pour les variantes sémantiques, CssClass pour le CSS ciblé propre à l'application, CssStyle uniquement pour les valeurs dynamiques ponctuelles, les conteneurs de mise en page plutôt que du code de redimensionnement, les propriétés responsives pour les valeurs de profil définies dans le designer, et ResponsiveProfileChanged pour le comportement qui n'est pas simplement visuel.