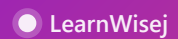


Fondamentaux de la mise en page : docking, ancrage, AutoSize, AutoScroll et composition du shell

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



Guide de la leçon

Le moteur de mise en page par défaut, l'ordre de docking et le padding, l'ancrage pour les régions de type formulaire, AutoSize et AutoScroll utilisés en toute sécurité, et la composition du shell à partir de conteneurs imbriqués et de UserControls plutôt que de gestionnaires de redimensionnement. Ce que cela signifie pour un développeur Wisej.NET — et comment aborder ce module.



Enjeu en production

Le moteur de mise en page par défaut, l'ordre de docking et le padding, l'ancrage pour les régions de type formulaire, AutoSize et AutoScroll utilisés en toute sécurité, et la composition du shell à partir de conteneurs imbriqués et de UserControls plutôt que de gestionnaires de redimensionnement.

1. Le problème en production

Adaptive Operations Console dispose désormais d'un thème et de CSS ciblé, mais son shell déplace encore les contrôles à la main : un gestionnaire **Resize** recalcule **Bounds** pour la barre d'outils, le rail de navigation, la grille des tickets et l'éditeur de détails chaque fois que le navigateur change de taille. Cela fonctionne à la largeur testée par le développeur, casse à toutes les autres, et ne peut pas survivre aux profils tablette et téléphone qu'introduit le Module 6. Dans Wisej.NET, la conception adaptative repose sur des conteneurs et des règles de mise en page, pas sur des coordonnées : des régions qui occupent des bords, un espace de travail qui remplit ce qui reste, et des éditeurs qui défilent au lieu de se chevaucher.

2. Concepts clés

Concept	Pourquoi c'est important
moteur de mise en page par défaut	Chaque contrôle Wisej.NET a un moteur de mise en page ; celui par défaut prend en charge la position absolue, le docking et l'ancrage, si bien qu'un écran peut combiner un placement précis dans le designer et des régions qui se redimensionnent d'elles-mêmes.
priorité de docking	Les enfants ancrés consomment l'espace dans l'ordre où ils apparaissent dans la collection de contrôles du parent ; c'est pourquoi les cinq mêmes panneaux produisent un shell correct ou un shell qui se chevauche selon l'ordre des enfants.
DisplayRectangle et Padding	Les contrôles ancrés se disposent dans le <code>DisplayRectangle</code> du parent, que <code>Padding</code> réduit ; le moteur par défaut ignore <code>Margin</code> pour les enfants ancrés, donc les espacements viennent du padding ou de contrôles d'espacement.
ancrage	Un ancrage conserve la distance d'un contrôle aux côtés choisis du parent, ce qui convient aux régions de type formulaire : labels en haut à gauche, zones de texte gauche-droite, boutons de commande en bas à droite.
AutoSize et mise en page circulaire	Un conteneur qui se dimensionne d'après des enfants dockés ou ancrés sur lui n'a pas de réponse stable ; cette boucle est la principale mise en garde du module concernant <code>AutoSize</code> .
AutoScroll avec barres masquées	<code>AutoScroll</code> fait défiler un conteneur quand le contenu ne tient pas, et <code>ScrollBars.Hidden</code> conserve le défilement tactile sur téléphone sans dessiner de barres de défilement.
région UserControl	Une région du shell avec sa propre mise en page locale et une petite surface publique, pour que le shell affiche, masque, ancre ou déplace la région entière selon le profil, sans connaître ses coordonnées internes.

3. Déroulé de la leçon

Docking, ordre des enfants et espace restant

Le docking attache un enfant à un bord de son parent (`DockStyle.Top` , `Bottom` , `Left` , `Right`) ou lui fait remplir ce qui reste (`DockStyle.Fill`). Le moteur distribue l'espace aux enfants ancrés un par un, selon leur ordre dans la collection du parent, et chaque enfant ancré réduit le rectangle disponible pour le suivant. C'est tout le secret du shell de la console : la barre d'outils prend une bande en haut, la région d'état une en bas, le rail de navigation une à gauche, le panneau de détails une à droite, et l'espace de travail remplit ce qui reste. Changez l'ordre et le panneau d'état masque la dernière ligne de la grille. Dans le designer de Visual Studio, `BringToFront` , `SendToBack` et les opérations sur l'index des enfants modifient cet ordre ; quand un shell paraît faux, vérifiez donc l'ordre des enfants avant de toucher à une taille. Le code du shell fourni est volontairement court, et le lab le reproduit dans le designer :

```
// MainPage.cs – the console shell, one Dock per region
this.toolbarPanel.Dock = DockStyle.Top;    this.toolbarPanel.Height = 56;
this.statusPanel.Dock = DockStyle.Bottom; this.statusPanel.Height = 28;
this.navigationPanel.Dock = DockStyle.Left; this.navigationPanel.Width = 240;
this.detailsPanel.Dock = DockStyle.Right;  this.detailsPanel.Width = 360;

this.workspacePanel.Dock = DockStyle.Fill;
this.workspacePanel.MinimumSize = new System.Drawing.Size(320, 240);

// gaps between docked regions come from the container, not from Margin
this.Padding = new Padding(8);
this.workspacePanel.Padding = new Padding(12, 8, 12, 8);
```

Les enfants ancrés sont placés dans le `DisplayRectangle` du parent, et `Padding` réduit ce rectangle de chaque côté. Le moteur de mise en page par défaut n'utilise pas `Margin` pour les contrôles ancrés : une marge sur le rail de navigation ne change rien à l'espace à côté de lui, alors donnez du padding au conteneur ou ancrez un fin panneau d'espacement. L'ancrage s'occupe des contrôles à l'intérieur d'une région. Un ancrage préserve la distance aux côtés choisis du parent : les labels en haut à gauche pour qu'ils restent en place, les zones de texte gauche-droite pour qu'elles s'élargissent avec le panneau de détails, le large éditeur de notes en haut-gauche-droite, et les boutons Save et Cancel en bas à droite. Un contrôle sans ancrage sur un axe flotte et garde une position proportionnelle, ce qui convient pour centrer une carte de connexion mais pas pour une grille.

Des conteneurs plutôt que des gestionnaires de redimensionnement

Le `MainPage_Resize` de la console est le signal d'alerte que ce module doit éliminer. Chaque ligne qui définit `Bounds`, `Location` ou `Size` en réponse à un redimensionnement est une règle que le moteur de mise en page aurait pu appliquer à votre place ; contrairement au moteur, elle s'exécute après que le navigateur a affiché la mauvaise mise en page, ignore tout du padding et ne se compose pas : dès que le panneau de détails devient un `UserControl`, le gestionnaire doit aussi connaître ses détails internes. La refactorisation est mécanique : listez les régions, décidez du bord que chacune occupe, ajoutez-les dans cet ordre avec `Dock`, et supprimez le gestionnaire. Les coordonnées absolues restent utiles pour un travail précis dans le designer à l'intérieur d'une boîte de dialogue de taille fixe ; elles sont le mauvais outil pour un écran qui doit aussi fonctionner sur un téléphone.

Un shell durable est un ensemble de régions imbriquées : barre de commandes en haut, navigation à gauche, espace de travail au centre, détails à droite, état en bas. Chaque région est un `Panel`, un `UserControl`, un `TabControl`, un `SplitContainer` ou une `Page` avec sa propre mise en page locale, et c'est l'imbrication qui rend le shell stable : l'espace de travail se moque de la façon dont la barre d'outils dispose ses boutons, et la barre d'outils se moque que l'espace de travail contienne des cartes de métriques au-dessus d'une grille de tickets. Ancrer un grand nombre de contrôles dans des combinaisons compliquées est aussi un signal : un `TableLayoutPanel` ou un `FlexLayoutPanel`, tous deux abordés au Module 5, exprime des champs alignés ou des régions proportionnelles plus clairement que douze réglages d'ancrage.

AutoSize, AutoScroll et les garde-fous de dimensionnement

`AutoSize` est précieux quand le contenu décide de la taille : un label qui doit contenir un texte traduit, un badge qui grandit avec son nombre, une carte qui enveloppe un contenu dynamique ; `AutoSizeMode` décide si le contrôle peut seulement grandir ou aussi rétrécir. Le danger est la mise en page circulaire. Si un conteneur se dimensionne automatiquement d'après des enfants dockés ou ancrés sur ce même conteneur, le parent demande aux enfants quelle taille prendre, les enfants demandent au parent quelle taille ils ont, et le moteur n'a pas de réponse stable. L'accident typique est un panneau avec `AutoSize` qui remplit aussi son parent. Gardez le dimensionnement automatique sur les contrôles feuilles et sur les conteneurs dont les enfants ont des tailles fixes ou dictées par leur contenu.

`AutoScroll` est la réponse sûre quand le contenu ne tient pas : le conteneur défile au lieu d'étirer le shell. L'éditeur de détails de la console compte plus de champs qu'un écran de téléphone ne peut en afficher ; le lab active donc `AutoScroll` et, aux largeurs étroites, règle `ScrollBars` sur `Hidden`, ce qui conserve le défilement tactile sans dessiner de barres qui volent de la largeur. `AutoScrollMargin` ajoute de l'espace après le dernier contrôle et `AutoScrollMinSize` garantit une zone défilante minimale. `MinimumSize` et `MaximumSize` sont les autres garde-fous : un `MinimumSize` sur l'espace de travail et sur la grille garantit que la région en remplissage ne s'effondre jamais en une mince bande quand le navigateur se rétrécit, et un `MaximumSize` empêche le panneau de détails d'engloutir un moniteur ultra-large.

Des régions UserControl que le shell peut réorganiser

La dernière étape du lab est celle qui porte ses fruits au Module 6. Une fois le rail de navigation et l'éditeur de détails transformés en `UserControl`, chacun avec son propre docking et son propre ancrage à l'intérieur, le shell ne connaît plus les coordonnées internes d'aucun éditeur. Il connaît cinq régions et, selon le profil, il affiche, masque, ancre ou déplace des régions entières : le rail peut devenir une étroite bande d'icônes, l'éditeur de détails peut passer sous la grille sur tablette ou s'ouvrir dans un `Form` modal sur téléphone, et aucun de ces déplacements ne touche à une zone de texte. La région expose quelques membres significatifs, comme une propriété `Ticket` et un événement `Saved`, et garde ses contrôles enfants privés, pour qu'un renommage interne ne casse jamais la page.

La checklist de mise en page du module est courte : préférer les conteneurs aux gestionnaires de redimensionnement, vérifier l'ordre de docking, mettre du `Padding` sur les régions ancrées, définir `MinimumSize` sur les régions en remplissage, et laisser défiler le contenu qui ne tient pas. La liste des signaux d'alerte de refactorisation est le miroir de cette checklist, et confronter un écran à cette liste est une bonne habitude avant chaque test de profil.



Signaux d'alerte de refactorisation

Du code `Resize` manuel qui définit `Bounds` à répétition sur de nombreux contrôles. `Dock` et `Anchor` combinés sur des contrôles où ils se contredisent. Des coordonnées absolues dans un écran qui doit fonctionner sur téléphone. Des grilles sans `MinimumSize`. Des panneaux avec `AutoSize` et un docking `Fill` qui créent des boucles. Des formulaires qui masquent des contrôles au lieu de déplacer le contenu de faible priorité dans une section défilante.



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 :

- ✓ Utiliser le docking et l'ancrage de façon intentionnelle et comprendre comment l'ordre des enfants influe sur la priorité de docking.
- ✓ Composer des écrans à partir de conteneurs imbriqués plutôt que de gestionnaires de redimensionnement et de mises à jour manuelles de coordonnées.
- ✓ Utiliser en toute sécurité AutoSize, AutoSizeMode, AutoScroll, ScrollBars, MinimumSize, MaximumSize, Padding et Margin.
- ✓ Créer des régions UserControl réutilisables qui peuvent être réorganisées entre les profils bureau, tablette et téléphone.

Guide du lab / de l'examen

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



Lab pratique

Refactorisez le shell d'Adaptive Operations Console en conteneurs imbriqués : une barre d'outils en haut, un rail de navigation à gauche, un espace de travail en remplissage, un panneau de détails à droite et une région d'état, chacun étant un Panel ou un UserControl avec sa propre mise en page locale. Supprimez tous les gestionnaires Resize qui définissent Bounds, corrigez l'ordre de docking dans le designer, utilisez Padding pour les espacements entre les régions ancrées, donnez un MinimumSize à l'espace de travail et à la grille, et faites défiler l'éditeur de détails avec AutoScroll et des barres de défilement masquées aux largeurs étroites.

Point de départ

Ouvrez la solution `AdaptiveOperationsConsole` telle que vous l'avez laissée après le Module 3 : la `Page` principale porte le thème AdaptiveOps, la feuille de style ciblée et le basculement de thème par session, mais ses cinq régions sont encore de simples panneaux dont les `Bounds` sont recalculés dans `MainPage_Resize`, et l'éditeur de détails chevauche la grille des tickets sous environ 1100 pixels. Exécutez-la une fois à la largeur bureau, puis réduisez le navigateur et notez ce qui est rogné, se chevauche ou disparaît avant de modifier quoi que ce soit.

Approche suggérée

- ✓ Listez chaque méthode `Resize`, `Layout` ou de chargement qui définit `Bounds`, `Location` ou `Size`, notez dans vos notes de lab les cinq régions et le bord que chacune occupe, puis supprimez ce code avant de toucher au designer.
- ✓ Dans le designer, définissez `Dock` sur les cinq panneaux (barre d'outils Top, état Bottom, navigation Left, détails Right, espace de travail Fill) et corrigez l'ordre des enfants avec `BringToFront` et `SendToBack` jusqu'à ce que les bords soient occupés en premier et que l'espace de travail prenne ce qui reste.
- ✓ Donnez à la page et aux régions un `Padding` pour les espacements entre zones ancrées, définissez `MinimumSize` sur l'espace de travail et la grille des tickets, et vérifiez que le moteur par défaut ignore `Margin` sur les enfants ancrés.
- ✓ Dans le panneau de détails, ancrez les labels en haut à gauche, les éditeurs gauche-droite, l'éditeur de notes en haut-gauche-droite et les boutons de commande en bas à droite ; activez `AutoScroll`, réglez `ScrollBars` sur `Hidden` pour les largeurs étroites et ajustez `AutoScrollMargin`.
- ✓ Extrayez `NavigationRail` et `DetailsEditor` sous forme de `UserControl` avec leur propre mise en page locale et une petite surface publique, remplacez-les dans le shell, et redimensionnez le navigateur de la largeur bureau à la largeur téléphone pour vérifier que rien ne se chevauche ni n'est rogné.

Livrables attendus

- ✓ Shell reconstruit à partir de conteneurs imbriqués avec les régions barre d'outils, navigation, espace de travail, détails et état
- ✓ Tout le code Resize manuel qui définissait Bounds supprimé
- ✓ Ordre de docking, Padding et MinimumSize définis intentionnellement sur chaque région
- ✓ Éditeur de détails défilant via AutoScroll avec barres de défilement masquées aux largeurs étroites
- ✓ Régions de navigation et de détails extraites dans des UserControls

Questions de révision

- ✓ Si le panneau d'espace de travail était ancré avant le panneau d'état dans l'ordre des enfants, que verrait l'utilisateur en bas de la grille, et quelle commande du designer corrige le problème sans changer une seule taille ?
- ✓ Vous avez défini une `Margin` de 12 pixels sur le rail de navigation et l'espace n'est pas apparu. Pourquoi, et quels sont les deux mécanismes qui produisent bien un espace entre des régions ancrées ?
- ✓ L'éditeur de détails est un `UserControl` avec `AutoScroll`. Quels membres le shell doit-il connaître pour le déplacer sous la grille sur tablette, et lesquels ne doit-il jamais avoir besoin de connaître ?



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.