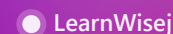


ClientProfile et propriétés responsives

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



Guide de la leçon

Comment ClientProfiles.json est évalué de haut en bas pour déterminer Application.ActiveProfile, les valeurs de propriétés responsives attribuées par profil dans le designer, ResponsiveProfileChanged pour le comportement que les propriétés responsives ne peuvent pas exprimer, et les règles de conception et de test des profils. Ce que cela signifie pour un développeur Wisej.NET — et comment aborder ce module.



Enjeu en production

Comment ClientProfiles.json est évalué de haut en bas pour déterminer Application.ActiveProfile, les valeurs de propriétés responsives attribuées par profil dans le designer, ResponsiveProfileChanged pour le comportement que les propriétés responsives ne peuvent pas exprimer, et les règles de conception et de test des profils.

1. Le problème en production

Adaptive Operations Console dispose maintenant d'un shell thématé construit avec des conteneurs Flow, Table et Flex qui résiste au redimensionnement sans gestionnaire Resize, mais il montre le même écran à tout le monde : sur un téléphone de 390 pixels, le rail de navigation, la grille des tickets et l'éditeur de détails se disputent la même largeur. Les media queries CSS ne peuvent pas régler ce problème, car les contrôles vivent sur le serveur : masquer un panneau, passer un bouton en icône seule ou ré-ancrer une région est un changement de propriété côté serveur, et un utilisateur de téléphone qui perd le panneau de détails a besoin d'une autre façon de modifier un ticket, pas d'une copie réduite de la version bureau. Ce module ajoute le mécanisme que Wisej.NET fournit pour cela : les profils client, les propriétés responsives et l'événement `ResponsiveProfileChanged`.

2. Concepts clés

Concept	Pourquoi c'est important
ClientProfile	Un ensemble nommé de règles de correspondance liées à l'appareil (largeur du navigateur, largeur d'écran, type d'appareil, user-agent, paysage) qui décrit la façon dont une catégorie de clients est utilisée, pas un nom commercial d'appareil.
ClientProfiles.json	Le fichier que lit Wisej.NET ; les profils sont évalués de haut en bas et le premier qui correspond l'emporte, l'ordre des entrées fait donc partie de la configuration.
Application.ActiveProfile	Le profil actuellement sélectionné pour cette session ; son <code>Name</code> est ce qu'affiche le label d'état et ce sur quoi se base le gestionnaire de profil.
propriété responsive	Une propriété .NET ordinaire marquée par <code>ResponsivePropertyAttribute</code> qui contient une valeur par profil client et applique la valeur correspondante chaque fois que le profil actif change.
adaptation par le designer d'abord	Les valeurs responsives ne s'attribuent que dans le designer de Visual Studio, ce qui les garde sérialisées, visibles et déterministes ; le code sert aux adaptations qui dépendent des données ou de l'état d'exécution.
ResponsiveProfileChanged	L'événement déclenché sur <code>Application</code> , sur les conteneurs comme Form, Page, Desktop et UserControl, et sur DataGridView et ListView quand le profil actif change ; <code>ResponsiveProfileChangedEventArgs.CurrentProfile</code> transporte le nouveau profil.
matrice de QA responsive	Le plan de test qui liste la largeur, la hauteur, l'orientation, le type d'appareil, le tactile, les barres de défilement, la mise à l'échelle des polices et le nom de profil attendu pour chaque taille.

3. Déroulé de la leçon

Comment un profil est retenu

Chaque entrée de `ClientProfiles.json` a un `name` et une combinaison quelconque de `device` (Mobile, Tablet ou Desktop), `landscape`, `minWidth` et `maxWidth` pour la fenêtre du navigateur, `minScreenWidth` et `maxScreenWidth` pour l'écran physique, et d'une correspondance `userAgent`. Wisej.NET lit le fichier, évalue les profils de haut en bas et retient la première correspondance comme `Application.ActiveProfile`. Rien n'est fusionné : un client correspond à un profil ou à aucun, c'est pourquoi le modèle ci-dessous se rabat sur « Default » quand `ActiveProfile` vaut null.

La correspondance peut se produire au chargement de la page ou au redimensionnement du navigateur, selon la propriété utilisée par une règle : une règle de largeur de fenêtre est réévaluée pendant qu'on étire le navigateur, tandis que la taille d'écran, le type d'appareil et le user-agent décrivent le client lui-même. Chaque fois que le profil sélectionné change, Wisej.NET déclenche `ResponsiveProfileChanged`. Pour le lab, vous écrivez six profils dans l'ordre de première correspondance : Phone, Phone Landscape, Tablet, Tablet Landscape, Small Desktop et Desktop. Les deux entrées téléphone partagent le `device` Mobile et ne diffèrent que par `landscape` ; Small Desktop est un appareil Desktop avec `maxWidth` 1024 ; Desktop

est la règle générale, en bas. Placez-la en haut et elle masque tous les profils situés en dessous, sans le moindre avertissement.

Les propriétés responsives relèvent du designer

Les propriétés responsives sont des propriétés ordinaires marquées par `ResponsivePropertyAttribute` : `Visible` , `Display` , `Size` , `Location` , `Dock` , `Anchor` , `Font` et les valeurs liées à la mise en page lorsque le contrôle les prend en charge. Dans le designer, une telle propriété contient une valeur différente pour chaque profil de `ClientProfiles.json` , et quand le profil actif change, le contrôle applique la valeur qui lui correspond. Le designer est le seul moyen pris en charge pour attribuer ces valeurs, et c'est une règle plutôt qu'une préférence : les valeurs attribuées là sont sérialisées avec le formulaire, visibles pour le développeur suivant et déterministes à l'exécution, alors que les valeurs définies en code sont invisibles à la conception et s'écartent de l'image de l'écran que montre le designer.

Dans le lab, chaque différence déterministe de la console relève du designer. Sur les deux profils Phone, le UserControl du rail de navigation et le panneau de détails reçoivent `Visible` à false. Les boutons de la barre d'outils reçoivent un `Display` icône seule sur Phone et Tablet pour que la barre de commandes tienne sur une ligne. Sur les profils Tablet, le `Dock` de la région de détails passe de Right à Bottom pour qu'elle se place sous la grille des tickets et non à côté. Aucun de ces changements ne dépend des données, aucun n'a donc sa place dans le code ; le code sert aux adaptations qui dépendent des données, des permissions, d'un calcul à l'exécution ou de l'événement de profil lui-même.

ResponsiveProfileChanged pour le comportement

Certaines adaptations ne sont pas des valeurs. Sur un téléphone, l'éditeur de détails ne peut pas du tout être un panneau latéral ; il doit s'ouvrir comme `Form` modal via `ShowDialog` , et le label d'état doit afficher et journaliser le nom du profil pour que le support sache quelle mise en page voit un utilisateur. C'est à cela que sert `ResponsiveProfileChanged` . Il est déclenché sur `Application` , sur les types `ContainerControl` comme `Form`, `Page`, `Desktop` et `UserControl`, et sur certains contrôles complexes comme `DataGridView` et `ListView`. Gérez-le sur `Application` pour un comportement global à l'application et sur un conteneur clé quand la logique est locale à cette région ; les arguments de l'événement exposent `CurrentProfile` .

Le modèle ci-dessous est la forme attendue par le lab : s'abonner une seule fois, appliquer le profil une fois au démarrage à partir d' `Application.ActiveProfile` pour que la mise en page initiale soit correcte, faire passer l'événement par la même méthode `ApplyProfile` , et se désabonner dans `Dispose` , car `Application` survit à la page. Gardez `ApplyProfile` léger et idempotent : un téléphone qui pivote entre Phone et Phone Landscape la déclenche à répétition, et un éditeur déjà ouvert doit être réutilisé, jamais rouvert. Travail typique de l'événement : charger moins de colonnes dans la grille, changer l'intervalle d'anti-rebond d'une recherche, transformer un panneau de détails en formulaire modal, appliquer un autre modèle de données.

```
// MainPage.cs – one method applies the profile at startup and on every change
Application.ResponsiveProfileChanged += Application_ResponsiveProfileChanged;
ApplyProfile(Application.ActiveProfile?.Name);

private void Application_ResponsiveProfileChanged(object sender,
ResponsiveProfileChangedEventArgs e)
{
    ApplyProfile(e.CurrentProfile?.Name);
}

private void ApplyProfile(string profileName)
{
    var phone = profileName != null && profileName.StartsWith("Phone");
    this.lblProfile.Text = profileName ?? "Default"; // shown and logged
    if (phone && this.ticketEditor == null) OpenDetailsAsModal(); // side panel → modal,
    once
}
```

Des variantes de conception qui changent le comportement

Les profils doivent décrire un usage, pas des appareils. Phone, Tablet Landscape, Small Desktop, Kiosk et Touch Workstation nomment chacun une façon d'utiliser la console, et chaque variante doit changer ce que fait la console : le téléphone reçoit un éditeur modal et un bouton de menu, la tablette empile la région de détails sous la grille, le petit bureau garde le shell avec une barre d'outils en icônes seules, le bureau affiche tout. Utilisez les règles de largeur pour la mise en page, les règles device pour les hypothèses de saisie comme la taille des zones tactiles et les barres de défilement masquées, `userAgent` seulement en cas de nécessité, et les règles landscape seulement quand l'orientation change vraiment l'interface. Un profil qui ne change qu'une dimension CSS n'avait pas besoin d'exister.

Testez des deux façons. Le vrai redimensionnement du navigateur exerce les règles de largeur ; l'émulation d'appareil exerce les règles device, de largeur d'écran et de user-agent, et après avoir changé d'appareil dans l'émulateur mobile de Chrome, vous devez rafraîchir la page pour que ces profils soient réévalués. La matrice de QA responsive a une ligne par scénario, de Desktop en 1440 sur 900 jusqu'à Phone en 390 sur 844 et Phone Landscape en 844 sur 390, chacune avec le nom de profil attendu et les vérifications qui prouvent le comportement : navigation visible, détails empilés, éditeur modal accessible, aucun débordement horizontal.

- Les profils étroits ou spécifiques avant les profils larges ; la première correspondance l'emporte et rien n'est fusionné.
- Les valeurs responsives dans le designer ; `ResponsiveProfileChanged` seulement quand une valeur ne suffit pas, et alors léger et idempotent.
- Afficher le nom du profil dans les builds de test et documenter chaque contrôle masqué avec son alternative mobile.



Pièges courants

Lister le profil général Desktop en premier, si bien que les profils téléphone et tablette ne sont jamais retenus. Attribuer les valeurs responsives en code au lieu du designer. Placer toutes les adaptations dans le gestionnaire d'événements, y compris celles qu'une valeur **Visible** ou **Dock** aurait exprimées. Rouvrir l'éditeur modal à chaque rotation parce que le gestionnaire n'est pas idempotent. Masquer un panneau sur téléphone sans alternative mobile pour ce qu'il contenait.



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 :

- ✓ Expliquer comment ClientProfiles.json est lu et évalué et comment Application.ActiveProfile est sélectionné.
- ✓ Utiliser les propriétés responsives dans le designer pour des propriétés comme Visible, Display, Size, Location, Dock, Anchor, Font et les valeurs liées à la mise en page lorsqu'elles sont disponibles.
- ✓ Gérer ResponsiveProfileChanged sur Application et sur les conteneurs clés lorsqu'une logique propre au profil est nécessaire.
- ✓ Concevoir des variantes téléphone, tablette, petit bureau et bureau qui changent le comportement de l'interface, pas seulement des dimensions CSS.

Guide du lab / de l'examen

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



Lab pratique

Ajoutez à Adaptive Operations Console un `ClientProfiles.json` avec les profils Phone, Phone Landscape, Tablet, Tablet Landscape, Small Desktop et Desktop, ordonnés du plus étroit au plus large. Dans le designer, utilisez les propriétés responsives pour masquer le rail de navigation et le panneau de détails sur téléphone, passer les boutons de la barre d'outils en Display icône seule et ré-ancrer la région de détails sous la grille sur tablette. Gérez `ResponsiveProfileChanged` pour afficher le nom du profil actif dans le label d'état, journaliser chaque changement et ouvrir l'éditeur de détails sous forme de formulaire modal sur téléphone au lieu d'un panneau latéral.

Point de départ

Ouvrez Adaptive Operations Console telle que l'a laissée le Module 5 : le shell est composé de conteneurs imbriqués avec une zone de cartes de métriques en Flow, un éditeur de ticket en `TableLayoutPanel` dans le UserControl de détails et un espace de travail en Flex, le thème `AdaptiveOps` et la feuille de style ciblée sont chargés, et le label d'état du Module 1 affiche toujours la largeur du navigateur. Il n'existe pas encore de `ClientProfiles.json`, donc `Application.ActiveProfile` vaut null pour tous les clients et chaque largeur affiche le shell de bureau.

Approche suggérée

- ✓ Écrivez `ClientProfiles.json` à la racine de l'application avec six entrées dans l'ordre de première correspondance (Phone, Phone Landscape, Tablet, Tablet Landscape, Small Desktop, Desktop) : les deux entrées téléphone utilisent le `device` Mobile et diffèrent par `landscape`, la paire tablette fait de même avec Tablet, Small Desktop est un appareil Desktop avec `maxWidth` 1024 et Desktop est la règle générale, en dernier. Exécutez une fois et vérifiez que les noms de profils apparaissent dans le designer.
- ✓ Dans le designer, donnez au UserControl du rail de navigation et au panneau de détails une valeur `Visible` à false pour les deux profils Phone, et vérifiez que la grille de l'espace de travail a toujours son `MinimumSize` pour que la mise en page téléphone ne s'effondre pas à zéro.
- ✓ Toujours dans le designer, réglez le `Display` des boutons de la barre d'outils sur icône seule pour les profils Phone et Tablet, et changez le `Dock` de la région de détails de Right à Bottom pour les deux profils Tablet afin qu'elle se place sous la grille des tickets ; ne touchez pas aux valeurs bureau.
- ✓ Dans `MainPage`, abonnez-vous une seule fois à `Application.ResponsiveProfileChanged`, écrivez une méthode `ApplyProfile(string profileName)` qui définit `lblProfile.Text` et écrit une ligne de journal par changement, appelez-la à la construction avec `Application.ActiveProfile?.Name` et depuis le gestionnaire avec `e.CurrentProfile?.Name`, et désabonnez-vous dans `Dispose`.
- ✓ Pour les profils Phone, faites ouvrir par `ApplyProfile` l'éditeur de ticket dans un `Form` modal via `ShowDialog`, en réutilisant le formulaire s'il est déjà ouvert ; testez ensuite en faisant glisser

le navigateur et avec l'émulation d'appareil, en rafraîchissant après chaque changement d'appareil, et capturez le label de profil à chaque taille.

Livrables attendus

- ✓ ClientProfiles.json avec les profils Phone, Phone Landscape, Tablet, Tablet Landscape, Small Desktop et Desktop dans l'ordre de première correspondance
- ✓ Valeurs de propriétés responsives attribuées dans le designer pour masquer le rail de navigation et le panneau de détails sur téléphone
- ✓ Display icône seule de la barre d'outils et région de détails ré-ancrée pour la tablette, définis via les propriétés responsives
- ✓ Gestionnaire ResponsiveProfileChanged qui affiche et journalise le nom du profil actif
- ✓ Éditeur de détails ouvert sous forme de formulaire modal sur téléphone via le gestionnaire de profil

Questions de révision

- ✓ Si Small Desktop était déplacé en tête de ClientProfiles.json, lesquels des cinq autres profils pourraient encore être retenus, et que verrait une tablette en paysage ?
- ✓ Le ré-ancrage tablette vit dans le designer et l'éditeur modal téléphone dans le code. Qu'est-ce qui décide quel mécanisme porte chaque changement, et qu'est-ce qui tournerait mal si vous les échangiez ?
- ✓ Un testeur a fait pivoter un téléphone quatre fois et votre gestionnaire s'est exécuté quatre fois. Quel état doit être identique après la quatrième exécution et après la première, et comment votre code le garantit-il ?



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.