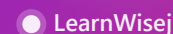


Postback, WebRequest, WebMethod et points de terminaison de données

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



Guide de la leçon

Alimenter des widgets JavaScript qui chargent leurs données en Ajax, tout en gardant les requêtes dans le contexte du bon composant Wisej.NET. Ce que cela signifie pour un développeur qui intègre avec Wisej.NET — et comment aborder ce module.



Objectif production

Alimenter des widgets JavaScript qui chargent leurs données en Ajax, tout en gardant les requêtes dans le contexte du bon composant Wisej.NET.

1. Le problème d'intégration

De nombreux widgets avancés, en particulier les grilles, tableaux croisés, rapports, visionneuses et outils de téléversement, attendent des URL ou des points de terminaison de service plutôt que des données poussées.

2. Concepts clés

Concept	Comment l'utiliser
URL de postback	Router les requêtes du navigateur vers la bonne instance de composant.
WebMethod	Exposer des appels de type RPC pour les callbacks ou les promises côté client.

3. Leçon de production

Certains widgets sont plus faciles à intégrer lorsqu'ils chargent leurs données depuis une URL. C'est courant pour les grilles de données, les tableaux croisés, les tableurs, les visionneuses, les outils de téléversement et les bibliothèques dotées de leur propre abstraction de source de données. Wisej.NET fournit une URL de postback liée à l'instance du composant, si bien que la requête arrive dans la bonne session et dans le contexte du bon contrôle. Un gestionnaire de postback se comporte comme un petit service web. Il reçoit un `HttpRequest` et écrit un `HttpResponse`. La réponse peut être du JSON pour une grille, du contenu binaire pour un fichier, des données d'image pour une visionneuse, ou tout format exigé par la bibliothèque tierce. L'avantage est que le gestionnaire vit avec l'instance du composant : il peut donc utiliser l'état du composant et les services de l'application. `[WebMethod]` a une autre forme. Il est idéal quand le navigateur doit appeler une méthode serveur publique et recevoir une valeur de retour marshalée. Il peut être statique ou rattaché à un conteneur de premier niveau. Il est généralement plus

propre pour les appels de type RPC, comme rechercher un enregistrement, valider une valeur ou renvoyer un petit jeu de résultats, tandis que les gestionnaires de postback restent mieux adaptés aux fichiers, aux images ou aux réponses volumineuses qu'une bibliothèque tierce s'attend à récupérer depuis une URL.

Router une URL vers une instance de composant

De nombreux widgets avancés récupèrent eux-mêmes leurs données. Une grille demande une page de lignes, une visionneuse diffuse un document, un outil de téléversement envoie un fichier. Tous veulent une URL, et la difficulté est que cette URL doit atteindre la bonne session Wisej.NET et, dans celle-ci, la bonne instance de composant. `GetPostBackURL()` côté serveur, ou `getPostbackUrl()` sur le widget client, renvoie précisément une telle URL, encodée avec les identifiants dont le framework a besoin pour livrer la requête à ce composant.

Le wrapper transmet cette URL à la bibliothèque tierce comme adresse de sa source de données, généralement dans `init`, et peut y ajouter ses propres paramètres de requête pour distinguer les actions. Comme l'URL est liée à une instance, deux grilles d'une même page obtiennent deux URL différentes, et chaque requête aboutit au composant qui possède les données.

Considérez l'URL comme liée à la session et de courte durée. Ce n'est pas l'adresse d'une API publique : elle ne doit être ni stockée ni partagée entre utilisateurs.

Traiter la requête dans WebRequest

Quand une requête arrive sur l'URL de postback, l'événement `WebRequest` du composant se déclenche avec les objets bruts `HttpRequest` et `HttpResponse`. Le gestionnaire lit l'action et les paramètres, produit les données, définit le type de contenu et écrit la réponse. Du JSON pour une grille, un flux PDF pour une visionneuse, une image pour une miniature ou un CSV pour un export ne sont que des corps différents avec des types de contenu différents.

```
private void grid_WebRequest(object sender, WebRequestEventArgs e)
{
    var action = e.Request.QueryString["action"];
    if (action != "load") { e.Response.StatusCode = 400; return; }
    var rows = _service.LoadPage(CurrentTenant, ParsePage(e.Request));
    e.Response.ContentType = "application/json";
    e.Response.Write(JsonSerializer.Serialize(rows));
}
```

Le gestionnaire s'exécute dans le contexte du composant : il peut donc lire l'état du composant et la session courante. C'est tout son intérêt : la requête de données sait quel tenant, quels filtres et quelles autorisations s'appliquent, sans que rien de tout cela ne soit transmis par le navigateur.

WebMethod pour les appels de type RPC

Quand l'appelant est le propre JavaScript de l'intégration et non une source de données tierce, une URL n'est pas la bonne forme. Une méthode marquée `[WebMethod]` peut être appelée depuis le client comme une fonction, avec des arguments typés marshalés depuis JavaScript et une valeur de retour marshalée en sens inverse, et le client peut attendre le résultat. Elle peut être statique, ou être une méthode d'instance d'un conteneur de premier niveau comme la page ou la fenêtre.

Le lab implémente le même jeu de données en lecture seule des deux façons pour rendre le compromis visible. L'URL de postback convient quand la bibliothèque tierce pilote la récupération et attend une adresse qu'elle peut appeler avec ses propres arguments de pagination. La WebMethod convient quand le code du wrapper a besoin d'une valeur, par exemple pour valider un code saisi ou récupérer une liste de référence, et devrait sinon analyser une réponse HTTP brute.

Les deux sont des points d'entrée vers le serveur, et les deux contournent le pipeline d'événements normal : ils exigent donc la même validation et la même autorisation qu'un gestionnaire de bouton.

Paramètres, types de contenu et sécurité

Chaque point de terminaison de données accepte des entrées venant du navigateur, et le navigateur peut être piloté par script par n'importe qui. Le paramètre d'action est comparé à une liste fixe, et non utilisé pour construire un nom de méthode. Les arguments de pagination sont analysés comme des entiers avec une borne supérieure, si bien qu'une demande d'un million de lignes est refusée. Les valeurs de filtre et de tri sont associées à des colonnes connues au lieu d'être injectées dans une requête. Tout ce qui est inattendu renvoie une erreur client et est journalisé avec les identifiants de session et de corrélation.

Le type de contenu est défini explicitement sur chaque réponse, car un corps JSON servi en texte déroute certaines bibliothèques et un corps HTML servi en JSON les déroutent toutes. La réponse de l'auto-vérification est `application/json` pour du JSON. Les réponses de fichier définissent l'en-tête de disposition et le vrai type MIME pour que le navigateur les traite correctement.

- L'utilisateur courant a-t-il le droit de voir ces lignes dans ce tenant, vérifié dans le gestionnaire.
- Les paramètres de pagination et de filtre sont-ils bornés et associés à une liste blanche.
- Une réponse d'erreur ne révèle-t-elle rien du serveur au-delà d'un code de statut.



Réflexe de production

Gardez clairement séparées la propriété de l'état côté Wisej.NET et le comportement du widget client. Documentez le contrat client/serveur avant de vous appuyer sur l'intégration en production.



Erreur courante à éviter

Copier du JavaScript de démonstration dans les écrans sans documenter le cycle de vie, les ressources, les contrats d'événements, le nettoyage ni les frontières de sécurité.



Objectifs d'apprentissage

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

- ✓ Utiliser ``getPostbackUrl`` / ``GetPostbackURL`` pour router les requêtes du navigateur vers une instance de composant.
- ✓ Traiter les requêtes avec ``WebRequest`` et renvoyer du JSON, des fichiers, des images ou toute réponse attendue par le widget.
- ✓ Exposer des méthodes ``[WebMethod]`` pour les appels client qui doivent se comporter comme du RPC plutôt que comme des requêtes HTTP brutes.

- ✓ Entourer les requêtes de données des widgets de paramètres d'action, de validation, de types de contenu et de contrôles de sécurité.

Guide du lab / de l'examen

Ce que vous allez construire dans le lab pratique, la démarche recommandée, les livrables attendus et l'auto-vérification avant la remise.



Lab pratique

Implémentez à la fois un point de terminaison postback et un point de terminaison WebMethod pour le même jeu de données en lecture seule. Connectez un widget simple à chacun et comparez les compromis.

Démarche recommandée

- ✓ Ouvrez le projet existant du cours d'intégration Wisej.NET et exécutez-le en local.
- ✓ Repérez le composant serveur, le widget client, le fichier de ressources ou le point de terminaison concernés par ce module.
- ✓ Faites une petite modification à la fois et testez-la dans le navigateur avant de passer à la suite.
- ✓ Gardez le JavaScript propre à la bibliothèque tierce dans l'adaptateur ou le wrapper au lieu de le disperser dans les pages.
- ✓ Mettez à jour l'interface pour que l'apprenant voie clairement le succès, l'échec, la progression ou les changements de communication.
- ✓ Prenez des captures d'écran ou des notes qui prouvent que l'intégration fonctionne et qui expliquent le choix de conception.

Livrables attendus

- ✓ Gestionnaire WebRequest en postback
- ✓ WebMethod avec arguments et valeur de retour
- ✓ Note comparative : source de données par URL ou callback client

Auto-vérification avant la remise

- ✓ Quand le postback est-il préférable à WebMethod ?
- ✓ Pourquoi faut-il valider les paramètres d'action ?
- ✓ Quel type de contenu les réponses JSON doivent-elles utiliser ?