

Réaliser un système de téléchargement avec barre de progression

par Rodrigue Hunel ([Site perso](#))

Date de publication : 03/03/2008

Suite à mes diverses recherches, je n'ai pas trouvé de tutoriels décrivant la méthode à utiliser pour effectuer ce genre d'application. Par conséquent, ce tutoriel aura pour but de vous présenter une des méthodes possibles pour réaliser une application de téléchargement avec l'affichage de l'état d'avancement avec une barre de progression.

- I - Introduction
- II - La classe WebClient
- III - Création de l'interface
- IV - Moteur de l'application
 - IV-A - Sélection de l'emplacement de sauvegarde
 - IV-B - Utilisation du WebClient
 - IV-B-1 - Tâches à effectuer au clic sur le bouton Télécharger
 - IV-B-2 - Tâches à effectuer pendant le téléchargement
 - IV-B-3 - Tâches à effectuer à la fin du téléchargement
 - IV-B-4 - Tâches à effectuer suite à une demande d'annulation
 - IV-C - Améliorations
- V - Code final
- VI - Conclusion
- VII - Remerciements

I - Introduction

Il existe plusieurs méthodes pour effectuer un téléchargement en C#. L'une des méthodes vous est déjà présentée sur la page **Sources C#** par **Thomas Lebrun (Télécharger un fichier depuis Internet)**.


Ce tutoriel aura pour but de vous présenter cette méthode qui est, selon moi, la plus simple et qui utilise la classe **WebClient** de la *super classe* **System.Net**.

II - La classe WebClient

WebClient est une classe qui offre des méthodes pour recevoir ou envoyer des données depuis n'importe quelle ressource identifiable par une adresse Internet (autrement dit sous la forme `http://site.domaine.fr/fichier/`)

Dans le cas de ce tutoriel, nous nous limiterons qu'aux méthodes de téléchargement :

- **OpenRead** et **OpenReadAsync** : Permettent la récupération de données depuis un flux Internet.
- **DownloadData** et **DownloadDataAsync** : Téléchargent les données et les renvoient sous forme de tableau d'octets.
- **DownloadFile** et **DownloadFileAsync** : Téléchargent des données pour les sauvegarder localement.
- **DownloadString** et **DownloadStringAsync** : Téléchargent et retournent une chaîne de caractères de type **String**.
- **CancelAsync** : Permet l'annulation des téléchargements en cours de traitement.

 *Toutes les méthodes ayant pour suffixe **Async** sont exécutés de la manière asynchrone. C'est-à-dire que ce sont des méthodes non bloquantes et qu'elles permettent d'avoir des applications non figées pendant le téléchargement des données.*


III - Création de l'interface

L'interface, pour ce tutoriel, sera composée de divers composants dont :

- Deux boîtes de texte (**textBox**) :
 - La première servira à saisir l'adresse du fichier à télécharger ;
 - La deuxième affichera l'adresse locale du fichier, choisie par l'utilisateur, après le téléchargement.
- Une boîte de dialogue (**SaveDialogFile**) qui permettra de choisir l'emplacement et le nom de sauvegarde du fichier ;
- Et trois boutons (**button**) :
 - Le premier permettra l'ouverture de la précédente boîte de dialogue ;
 - Le second qui lancera le téléchargement suite à un clic ;
 - Et le dernier offrira la possibilité d'annuler le téléchargement.
- Une barre de progression qui affichera la progression du téléchargement en temps réel.

Nous aurons donc l'interface suivante :



 Ici, j'ai rajouté deux propriétés, assez pratiques, au champ de saisie qui servira à saisir l'adresse de téléchargement. Ces propriétés sont :

- **AutoCompleteMode** à **Suggest**
- **AutoCompleteSource** à **AllUrl**

Elles permettront d'ajouter une auto-complétion, lors de la saisie, qui permettra de récupérer une adresse déjà saisie dans votre navigateur. Pour obtenir plus d'informations sur le sujet, n'hésitez pas à consulter le tutoriel de Louis-Guillaume Morand : [L'auto-complétion dans une application .Net](#)

IV - Moteur de l'application

Avant toutes choses, voici le fonctionnement de l'application qui sera réalisé avec la classe **WebClient** :

- 1 On saisi l'adresse du fichier à télécharger ;
- 2 Ensuite on définit l'emplacement de sauvegarde sur le disque ;
- 3 Enfin, on lance le téléchargement suite à un clic sur le bouton *Télécharger*. Au cours de ce téléchargement, la barre de progression affichera son état d'avancement. Un clic sur le bouton *Annuler* annulera le téléchargement et supprimera le fichier temporaire;
- 4 Une fois le téléchargement terminé, nous aurons un fichier enregistré localement et nous pourrons réitérer l'opération si on le souhaite.

IV-A - Sélection de l'emplacement de sauvegarde

Comme je l'ai dit plus haut, nous avons une boîte de dialogue qui nous permettra de le faire suite à un clic sur le bouton *Sélectionner* (nommé ici : selectBouton). Par conséquent, il nous reste plus qu'à définir l'action à effectuer suite à ce fameux clic :


```
private void selectBouton_Click(object sender, EventArgs e)
{
    if (saveFileDialog.ShowDialog() == DialogResult.OK)
    {
        saveTo.Text = saveFileDialog.FileName;
    }
}
```

 *saveTo est le nom du champ texte affichant l'emplacement de sauvegarde choisi.*


IV-B - Utilisation du WebClient

Au cours de ce paragraphe, nous allons définir les tâches à effectuer suite au clic sur le bouton *Télécharger*, pendant le téléchargement et à la fin du téléchargement.

Mais avant toute chose nous devons créer une instance vers un objet de type **WebClient**. Pour ce faire, nous avons deux options : soit le faire à partir de l'éditeur graphique (comme cela a été fait pour la construction de l'interface), soit en la déclarant nous même dans le code. Ici, j'opterais pour la deuxième option.

 *Si vous optez pour la deuxième option, n'oubliez pas d'inclure la super-classe de **WebClient** et de définir l'instance de manière globale afin de pouvoir l'utiliser dans n'importe quelle fonction.*

```
using System.Net;
```

 *Le téléchargement de fichier peut être une action bloquante, il faut donc agir en conséquence soit en utilisant un **Thread** ou encore un **BackgroundWorker** soit faire appel à la méthode asynchrone disponible : **DownloadFileAsync**.*

IV-B-1 - Tâches à effectuer au clic sur le bouton Télécharger

Nous devons ici tout d'abord nous abonner à l'évènement **onclick** du bouton **Télécharger** afin de pouvoir définir toutes les tâches à effectuer. Pour information ce bouton a pour nom : `downloadBouton`. On obtient donc la fonction suivante.

```
private void downloadBouton_Click(object sender, EventArgs e)
{
    //Création d'une nouvelle instance de WebClient
    WebClient client = new WebClient();

    //On lance le téléchargement avec les paramètres saisies.
    //Le premier correspond à l'adresse du fichier à télécharger et le second à l'emplacement de
    sauvegarde
    client.DownloadFileAsync(new Uri(fileToDownload.Text.Trim()), saveTo.Text);

    downloadBouton.Enabled = false; // Rend inaccessible le bouton Télécharger
    downloadBouton.Text = "En cours..."; // Modifie le texte de ce même bouton

    //Rend accessible le bouton Annuler afin de pouvoir arrêter le téléchargement
    cancelButton.Enabled = true;
}
```

IV-B-2 - Tâches à effectuer pendant le téléchargement


On souhaite afficher uniquement l'état d'avancement, il nous faut donc définir une fonction qui sera appelée à chaque état d'avancement du téléchargement. Pour ce faire, on doit abonner notre instance client à l'évènement **DownloadProgressChanged**.

Tout d'abord, on définit notre nouvelle fonction qui mettra à jour notre barre de progression.

```
private void client_DownloadProgressChanged(object sender, DownloadProgressChangedEventArgs e)
{
    // Met à jour la position de la barre de progression à partir
    // de l'état d'avancement contenu dans l'attribut ProgressPercentage
    progressBar.Value = e.ProgressPercentage;
}
```

Puis on rajoute une nouvelle ligne à la fonction **downloadBouton_Click** :

```
//On abonne notre instance client à l'évènement de progression du téléchargement
client.DownloadProgressChanged += new
DownloadProgressChangedEventHandler(client_DownloadProgressChanged);
```

 Avec les précédentes fonctions définies, nous avons déjà une application qui permet d'effectuer un téléchargement avec affichage de son état d'avancement. Il nous reste plus qu'à rajouter les tâches à effectuer à la fin du téléchargement.

IV-B-3 - Tâches à effectuer à la fin du téléchargement

A la fin du téléchargement survient l'évènement **DownloadFileCompleted** et c'est grâce à lui que l'on saura qu'il est terminé.

Tout d'abord, on définit les tâches à effectuer dans une nouvelle fonction.

```
private void client_DownloadFileCompleted(object sender, AsyncCompletedEventArgs e)
{
    //Affichage du message de confirmation
    MessageBox.Show("Téléchargement terminé.");

    //Réinitialisation des champs texte
    fileToDownload.ResetText();
    saveTo.ResetText();

    //Réinitialisation de la boîte de dialogue et de ses données
    saveFileDialog.Reset();

    //Rend accessible le bouton Télécharger
    downloadBouton.Enabled = true;
    downloadBouton.Text = "Télécharger";

    //Rend inaccessible le bouton Annuler
    cancelButton.Enabled = false;

    //Réinitialisation de la barre de progression
    progressBar.Value = 0;
}
```

Une fois que c'est fait, il nous reste plus qu'à lui dire qu'il doit effectuer ces tâches suite à la capture de cet évènement en s'y abonnant lors de la création de notre instance (c'est-à-dire dans la fonction **downloadBouton_Click**).

```
//On abonne notre instance client à l'évènement de fin du téléchargement
client.DownloadFileCompleted += new AsyncCompletedEventHandler(client_DownloadFileCompleted);
```

Maintenant, il nous reste plus qu'à définir les actions à effectuer dans le cas d'une demande d'annulation.

IV-B-4 - Tâches à effectuer suite à une demande d'annulation

Ici, nous ferons appel à la méthode **CancelAsync** qui, une fois appelée, mais fin au téléchargement. Donc nous aurons une unique instruction dans notre fonction.

```
private void cancelButton_Click(object sender, EventArgs e)
{
    client.CancelAsync(); //Annulation du téléchargement en cours
}
```

Etant donné que le fait d'annuler le téléchargement émet un évènement de fin de téléchargement (vu précédemment), il faut modifier le code défini plus haut afin de différencier l'annulation et un téléchargement terminé avec succès.

On peut faire une modification de ce genre :

```
if (!e.Cancelled)
{
    //Affichage du message de confirmation
    MessageBox.Show("Téléchargement terminé.");
}
else
{
    //Affichage du message d'annulation
    MessageBox.Show("Téléchargement annulé.");
}
```

Une fois toutes ces étapes réalisées, nous avons enfin notre logiciel de téléchargement qui est entièrement fonctionnel. Toutefois, nous pouvons l'améliorer afin d'effectuer quelques petites vérifications comme celle de la chaîne saisie par l'utilisateur, par exemple.

IV-C - Améliorations

Comme je le disais plus haut, une des améliorations possibles est la vérification de l'adresse saisie. Pour ce faire, il existe des fonctions définies à cet effet.

Ainsi, on peut modifier la fonction.

```
private void downloadBouton_Click(object sender, EventArgs e)
{
    if (Uri.IsWellFormedUriString(fileToDownload.Text.Trim(), UriKind.Absolute))
    {
        //Création d'une nouvelle instance de WebClient
        client = new WebClient();

        //On abonne notre instance client à l'évènement de progression du téléchargement
        client.DownloadProgressChanged += new
        DownloadProgressChangedEventArgs(client_DownloadProgressChanged);

        //On abonne notre instance client à l'évènement de fin du téléchargement
        client.DownloadFileCompleted += new
        AsyncCompletedEventHandler(client_DownloadFileCompleted);

        //On lance le téléchargement avec les paramètres saisies.
        //Le premier correspond à l'adresse du fichier à télécharger et le second à l'emplacement
        de sauvegarde
        client.DownloadFileAsync(new Uri(fileToDownload.Text.Trim()), saveTo.Text);

        downloadBouton.Enabled = false; // Rend inaccessible le bouton Télécharger
        downloadBouton.Text = "En cours..."; // Modifie le texte de ce même bouton

        //Rend accessible le bouton Annuler afin de pouvoir arrêter le téléchargement
        cancelButton.Enabled = true;
    }
    else
    {
        MessageBox.Show("Vous devez saisir une adresse correcte");
    }
}
```

V - Code final

```
WebClient client = null;

private void selectBouton_Click(object sender, EventArgs e)
{
    if (saveFileDialog.ShowDialog() == DialogResult.OK)
    {
        saveTo.Text = saveFileDialog.FileName;
    }
}

private void downloadBouton_Click(object sender, EventArgs e)
{
    if (Uri.IsWellFormedUriString(fileToDownload.Text.Trim(), UriKind.Absolute))
    {
        //Création d'une nouvelle instance de WebClient
        client = new WebClient();

        //On abonne notre instance client à l'évènement de progression du téléchargement
        client.DownloadProgressChanged += new
DownloadProgressChangedEventArgs(client_DownloadProgressChanged);

        //On abonne notre instance client à l'évènement de fin du téléchargement
        client.DownloadFileCompleted += new
AsyncCompletedEventHandler(client_DownloadFileCompleted);

        //On lance le téléchargement avec les paramètres saisies.
        //Le premier correspond à l'adresse du fichier à télécharger et le second à l'emplacement
de sauvegarde
        Uri url = new Uri(fileToDownload.Text.Trim());
        client.DownloadFileAsync(url, saveTo.Text);

        downloadBouton.Enabled = false; // Rend inaccessible le bouton Télécharger
        downloadBouton.Text = "En cours..."; // Modifie le texte de ce même bouton

        //Rend accessible le bouton Annuler afin de pouvoir arrêter le téléchargement
        cancelButton.Enabled = true;
    }
    else
    {
        MessageBox.Show("Vous devez saisir une adresse correcte");
    }
}

private void client_DownloadProgressChanged(object sender, DownloadProgressChangedEventArgs e)
{
    // Met à jour la position de la barre de progression à partir
// de l'état d'avancement contenu dans l'attribut ProgressPercentage
    progressBar.Value = e.ProgressPercentage;
}

private void reinitialisation()
{
    //Réinitialisation des champs texte
    fileToDownload.ResetText();
    saveTo.ResetText();

    //Réinitialisation de la boîte de dialogue et de ses données
    saveFileDialog.Reset();

    //Rend accessible le bouton Télécharger
    downloadBouton.Enabled = true;
    downloadBouton.Text = "Télécharger";
}
```

```
//Rend inaccessible le bouton Annuler
cancelButton.Enabled = false;

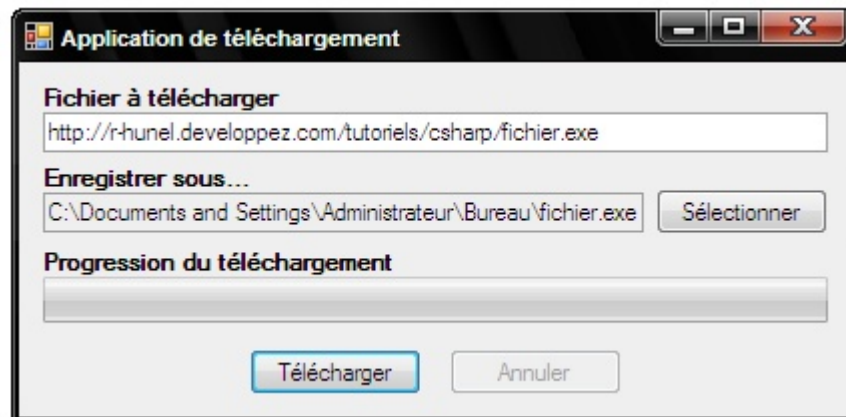
//Réinitialisation de la barre de progression
progressBar.Value = 0;
}

private void client_DownloadFileCompleted(object sender, AsyncCompletedEventArgs e)
{
    if (!e.Cancelled)
    {
        //Affichage du message de confirmation
        MessageBox.Show("Téléchargement terminé.");
    }
    else
    {
        //Affichage du message d'annulation
        MessageBox.Show("Téléchargement annulé.");
    }

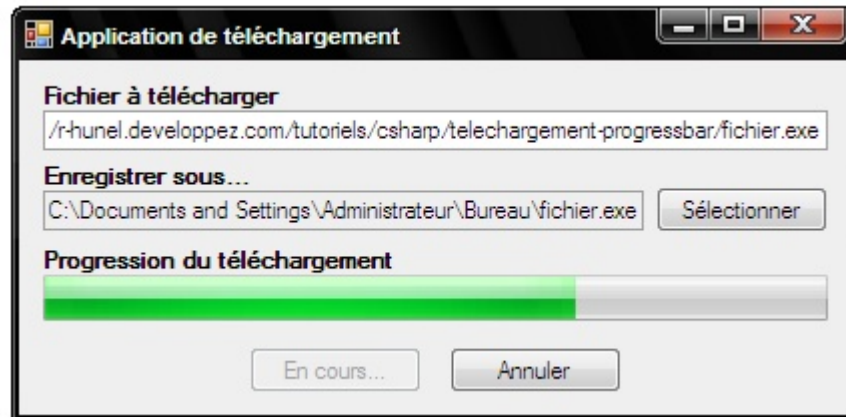
    reinitialisation();
}

private void cancelButton_Click(object sender, EventArgs e)
{
    client.CancelAsync();
}
```

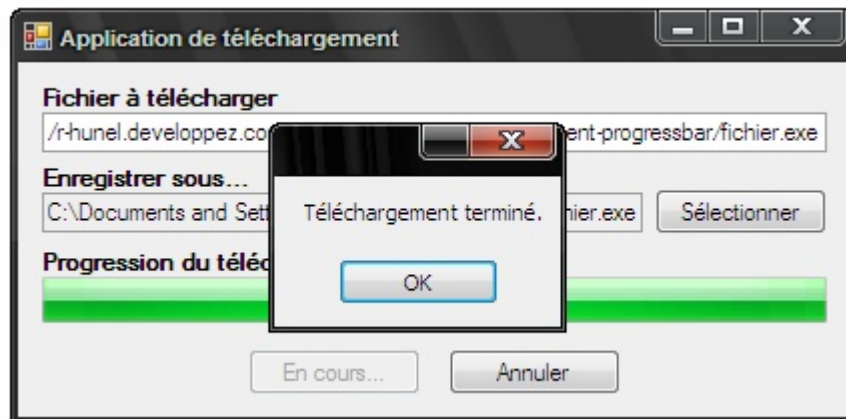
Voici quelques captures d'écran de l'application effectuant un téléchargement.




Après sélection du fichier et de l'emplacement de sauvegarde



En cours de téléchargement



A la fin du téléchargement

 Je vous propose de télécharger le code source afin de pouvoir l'utiliser sans avoir à le retaper :

- [Source](#) Compatible Visual Studio 2008
- [Source](#) Compatible Visual Studio 2005

VI - Conclusion

Ce tutoriel nous a permis de découvrir l'une des méthodes de téléchargement de fichier et d'afficher son état d'avancement. Ce n'est pas une méthode universelle mais elle a le mérite d'être simple à mettre en oeuvre.

VII - Remerciements

Tous mes remerciements à **Cardi** et **Aspic** pour leurs précieux conseils, ainsi qu'à **ArHacKnIdE** pour sa relecture.

