Démarrer avec OGDI DataLab

Solution Open Data Clé en main

Publication : Septembre 2014

Version: 6.1

Auteur : Philippe Beraud

Contributeurs/Relecteurs : Sébastien Brasseur, Rémi Olivier, Frédéric Romanczuk, Alexandre Fournier, Ludovic Panchaud

Pour les dernières informations sur la solution communautaire Open Data Clé en main, consultez

aka.ms/OpenDataFrance

Sommaire

[Licence d’OGDI DataLab 4](#_Toc399310889)

[Résumé 5](#_Toc399310890)

[Commentaires 5](#_Toc399310891)

[Prérequis 6](#_Toc399310892)

[Connaissances et attendu 6](#_Toc399310893)

[Présentation rapide d’OGDI DataLab 7](#_Toc399310894)

[Contenu d’OGDI DataLab 9](#_Toc399310895)

[Kit de développement logiciel interactif (DataBrowser) 9](#_Toc399310896)

[Service de données (DataService) 10](#_Toc399310897)

[Configurateur de catalogues (DataConfig) 11](#_Toc399310898)

[Utilitaires de chargement de données (DataLoader) 12](#_Toc399310899)

[Configuration requise pour OGDI DataLab 13](#_Toc399310900)

[Configuration système 13](#_Toc399310901)

[Comment installer les composants système nécessaires 14](#_Toc399310902)

[Obtention des sources d’OGDI DataLab 16](#_Toc399310903)

[Organisation du code source d’OGDI DataLab 17](#_Toc399310904)

[Dossier DataBrowser (Kit de développement logiciel interactif) 17](#_Toc399310905)

[Organisation projet 18](#_Toc399310906)

[Eléments de configuration 19](#_Toc399310907)

[A propos des vues proposées et de la localisation 23](#_Toc399310908)

[Dossier DataService (Service de données) 27](#_Toc399310909)

[Organisation projet 27](#_Toc399310910)

[Eléments de configuration 27](#_Toc399310911)

[Dossier DataConfig (Configurateur de catalogues) 28](#_Toc399310912)

[Organisation projet 28](#_Toc399310913)

[Dossier DataLoader (Utilitaires de chargement de données) 28](#_Toc399310914)

[Organisation projet 28](#_Toc399310915)

[Eléments de configuration 29](#_Toc399310916)

[Dossier DAISY 29](#_Toc399310917)

[Dossier Shared 31](#_Toc399310918)

[Organisation projet 31](#_Toc399310919)

[Dépendances 31](#_Toc399310920)

[Résolution des dépendances avec NuGet 32](#_Toc399310921)

[Compilation des éléments d’OGDI DataLab 37](#_Toc399310922)

[Compilation des éléments en ligne de commande 37](#_Toc399310923)

[Compilation des éléments en utilisant Visual Studio 2013 37](#_Toc399310924)

[Compilation des éléments via le script d’automatisation 37](#_Toc399310925)

[Mise en œuvre des éléments d’OGDI DataLab dans Azure 38](#_Toc399310926)

# Licence d’OGDI DataLab



**OGDI** (Open Government Data Initiative) **DataLab**, l’un des accélérateurs de la [solution Open Data Clé en main](http://aka.ms/OpenDataCleEnMain)[[1]](#footnote-1), est publiée sous le contrat de licence de logiciel libre Microsoft Public License (Ms-PL) tel que décrit à l’adresse Internet <http://opensource.org/licenses/ms-pl.html>.

# Résumé

**OGDI** (Open Government Data Initiative) **DataLab** sous licence libre est une initiative menée par les équipes Microsoft en charge du [Secteur Public](http://www.microsoft.com/france/entreprises/secteur-public/solutions-pour-secteur-public.aspx)[[2]](#footnote-2). Il constitue l’un des accélérateurs de la solution Open Data Clé en main.

OGDI DataLab utilise l’environnement d’exécution et d’hébergement [Microsoft Azure](http://azure.microsoft.com/fr-fr/)[[3]](#footnote-3) dans le cloud pour faciliter la publication et l'utilisation d’informations publiques ou données ouvertes (*open data*) variées provenant d'entités publiques (gouvernement, ministères, collectivités locales et territoriales, mairies, services déconcentrés de l'Etat, etc.).

Pour se faire, la plateforme de publication des données ouvertes OGDI DataLab propose une gestion multi-catalogues de (jeux de) données ouvertes et un service de données RESTful [OData (Open Data Protocol)](http://www.odata.org/)[[4]](#footnote-4) pour la consommation des données ainsi mises à disposition. Cet accélérateur de la solution communautaire Open Data clé en main disponible sur la [forge GitHub](https://github.com/openlab/OGDI-DataLab)[[5]](#footnote-5) s’adapte très facilement à la stratégie de l’acteur public notamment en termes de portail usager.

Après une courte présentation de la plateforme, de ses fondations et de son architecture, ce document décrit le contenu à proprement parler de l’accélérateur, l’organisation du code source ainsi mis à disposition et ses modalités de compilation. D’autres documents accompagnant le kit décrivent les modalités de mise en œuvre du kit et de mise à disposition de données dans Azure.

Ce document s’adresse aux architectes, aux développeurs et à tous ceux qui souhaitent exposer et/ou consommer des informations publiques au sens du protocole OData. Les sites Web, services, et utilitaires et le code source afférent fournis dans OGDI DataLab sont conçus pour interagir avec une très large variété de technologies utilisées aujourd’hui sur Internet, telles que ASP.NET, JavaScript, Flash, Flex, PHP, Python, Ruby, Silverlight, etc., ou encore les sites de cartographie comme Géoportail, le portail des territoires et des citoyens, Bing Maps, Google Maps, etc.

# Commentaires

Votre opinion nous intéresse. N’hésitez pas à nous transmettre vos commentaires via la boîte aux lettres Open Data France (<mailto:ogdifrance@live.fr>) afin que nous puissions améliorer la solution communautaire Open Data Clé en main.

Le [blog MSDN Open Data France](http://blogs.msdn.com/b/ogdifrance/)[[6]](#footnote-6) vous tient informé dans le même temps des dernières évolutions de la solution Open Data Clé en main, des différents accélérateurs proposés dans ce contexte, de l’actualité et des nouveautés relatives à ceux-ci.

# Prérequis

La plateforme de publication de données ouvertes OGDI DataLab à destination de l’environnement d’exécution Azure nécessite un ordinateur sous Windows Server (R2) 2008 ou ultérieur.

Le lecteur doit connaître les technologies des services Web RESTful, le langage C# et .NET Framework afin de pouvoir facilement comprendre les éléments mis ainsi à disposition.

# Connaissances et attendu

La mise en œuvre d’OGDI DataLab requiert une expérience préalable :

* De l’environnement d’exécution et d’hébergement Microsoft Azure,
* De l’environnement de développement Microsoft Visual Studio 2013,
* De la programmation avec le Microsoft Framework .NET,
* Du développement et de l’utilisation d’interfaces Web de style REST (Representational State Transfer),
* Des dernières technologies Web comme ASP.NET MVC ou encore jQuery.

Remarque Microsoft met à disposition des formations gratuites téléchargeables sur ces environnements et technologies. Nous vous invitons à considérer le cas échéant le kit de formation [Azure Readiness Content](http://www.microsoft.com/en-us/download/details.aspx?id=8396)[[7]](#footnote-7) couvrant l’environnement Azure et l’environnement de développement Microsoft Visual Studio 2013. Ce dernier comprend des présentations, des vidéos de formation, des démonstrations accompagnées de leur script ainsi que des ateliers didactiques complets.

Remarque Le [blog du groupe Produit Microsoft Azure](http://blogs.msdn.com/b/windowsazure/)[[8]](#footnote-8) propose également de nombreuses ressources en termes de formation sur cet environnement d’exécution et d’hébergement. Celui-ci constitue également un excellent moyen de se tenir informé des évolutions proposées par l’environnement.

# Présentation rapide d’OGDI DataLab

Les informations publiques ou données ouvertes (open data) sont aujourd’hui en termes de potentiel, d’opportunités et d’enjeu au cœur du développement de multiples innovations sociales et économiques grâce à leur mise en ligne par l’administration et les collectivités territoriales.

Portées à la connaissance des citoyens, des chercheurs, des associations, des entreprises, ces informations statistiques, géographiques, urbanistiques, etc. peuvent devenir des mines d’or dès lors qu’elles sont retravaillées, croisées entre elles (mash-up) et mises en scène et ainsi contribuer à la création de nouveaux services tout en favorisant la transparence de l’action publique.

Dans un tel contexte, la publication et la consommation d’informations publiques sur le Web sous une forme facilement accessible par les personnes et les applications constitue une dimension essentielle. Afin justement de faciliter la mise à disposition et l’utilisation d’informations publiques, Microsoft France souhaite contribuer à cette dynamique en marche au travers d’OGDI DataLab qui permet à l’administration et aux collectivités territoriales d’accélérer leur projet de publication de données, de publier ainsi les informations ouvertes de leur choix plus rapidement et efficacement, et de les rendre navigables et interrogeables par les citoyens et les applications.

La technologie ainsi mise à disposition sous forme d’un accélérateur gratuit et sous licence libre dans le cadre de la solution communautaire Open Data Clé en main rend possible la mise à disposition facilitée par l’Administration et les Collectivités Territoriales de services de données performants qu’elles exploitent, les informations publiques étant hébergées dans l’environnement Azure dans le cloud.

Aujourd’hui, la plupart des initiatives se concentrent uniquement sur la publication des données uniquement au travers de fichiers dits « plats », fichiers Excel par exemple, qu’il est nécessaire de télécharger intégralement. L’écriture d'une application signifie alors téléchargement, hébergement et maintenance des données.

La plateforme de publication de données OGDI DataLab procure de nombreux avantages. En premier lieu, il s’agit d’une plate-forme :

* Les personnes peuvent écrire des programmes pour accéder aux données sans téléchargement ni hébergement ; les créateurs d’applications ont seulement besoin d'accéder à des interfaces de programmation (APIs) fondées sur le Web de style REST.

Que vous soyez une Administration, une Collectivité Territoriale, une entreprise en contrat avec une telle entité, une entreprise souhaitant utiliser les informations publiques, ou un développeur « citoyen », ces APIs ouvertes vous permettent de construire des applications innovantes, des représentations graphiques avancées et des agrégations (mash-up) de données dont pourront bénéficier les citoyens et usagers.

* Les Administration et Collectivités Territoriales peuvent conserver les données à jour afin que les créateurs d’applications n'aient pas à s'en inquiéter.
* Les Administration et Collectivités Territoriales n’ont pas la nécessité d'investir et de maintenir des plateformes et logiciels système et ne paieront que pour les ressources qui ont été effectivement consommées.
* Les applications clientes peuvent être écrites à l'aide de toute technologie, sans restriction aucune.

Il offre choix et ouverture. Notamment :

* Les applications clientes et les visualisations de tout type peuvent être écrites à l'aide de toute technologie.
* Les données sont exposées dans une variété de formats ouverts : [AtomPub (Atom Publishing Protocol)](http://tools.ietf.org/html/rfc5023)[[9]](#footnote-9), [KML (Keyhole Markup Language)](http://www.opengeospatial.org/standards/kml/)[[10]](#footnote-10), [JSON (JavaScript Object Notation)](http://tools.ietf.org/html/rfc4627)[[11]](#footnote-11), [JSONP (JSON with padding)](http://www.json-p.org/)[[12]](#footnote-12) et [RDF (Resource Description Framework)](http://www.w3.org/RDF/)[[13]](#footnote-13).

La plateforme de démonstration mise à disposition par Microsoft France à l’adresse <http://ogdifrance.cloudapp.net> est construite en utilisant les composantes d’OGDI DataLab en version 2014-09 et fournit un accès interactif à certains jeux de données accessibles au public avec, à la clé, exemples de code et ressources pour écrire des applications en utilisant le modèle de programmation mis en œuvre par OGDI DataLab.

La version 2014-09 de la plateforme de publication de données de la plateforme OGDI DataLab correspond :

* D’une part, à l’adaptation en français accompagnée d’évolutions significatives du kit de développement logiciel OGDI v2 disponible en téléchargement sous la même licence libre sur la [forge CodePlex](http://ogdi.codeplex.com/)[[14]](#footnote-14).
* Et, d’autre part, à la fusion avec le kit de développement DataLab (OGDI v4) disponible sur la [forge GitHub](https://github.com/openlab/datalab)[[15]](#footnote-15), qui constitue lui-même une évolution du kit de développement logiciel OGDI v2 disponible sur la forge CodePlex.

[image](https://github.com/openlab/DataLab)

**La version 2014-09 d’OGDI DataLab, fusion et évolution de ces deux branches de code, constitue le** [**projet OGDI-DataLab**](https://github.com/openlab/OGDI-DataLab)**[[16]](#footnote-16) sur la forge Github ; ce dernier est ainsi l’unique référence communautaire.**

# Contenu d’OGDI DataLab

L’accélérateur OGDI DataLab se présente sous la forme d’une solution Visual Studio 2013 *ogdi.sln* à destination de l’environnement Azure.

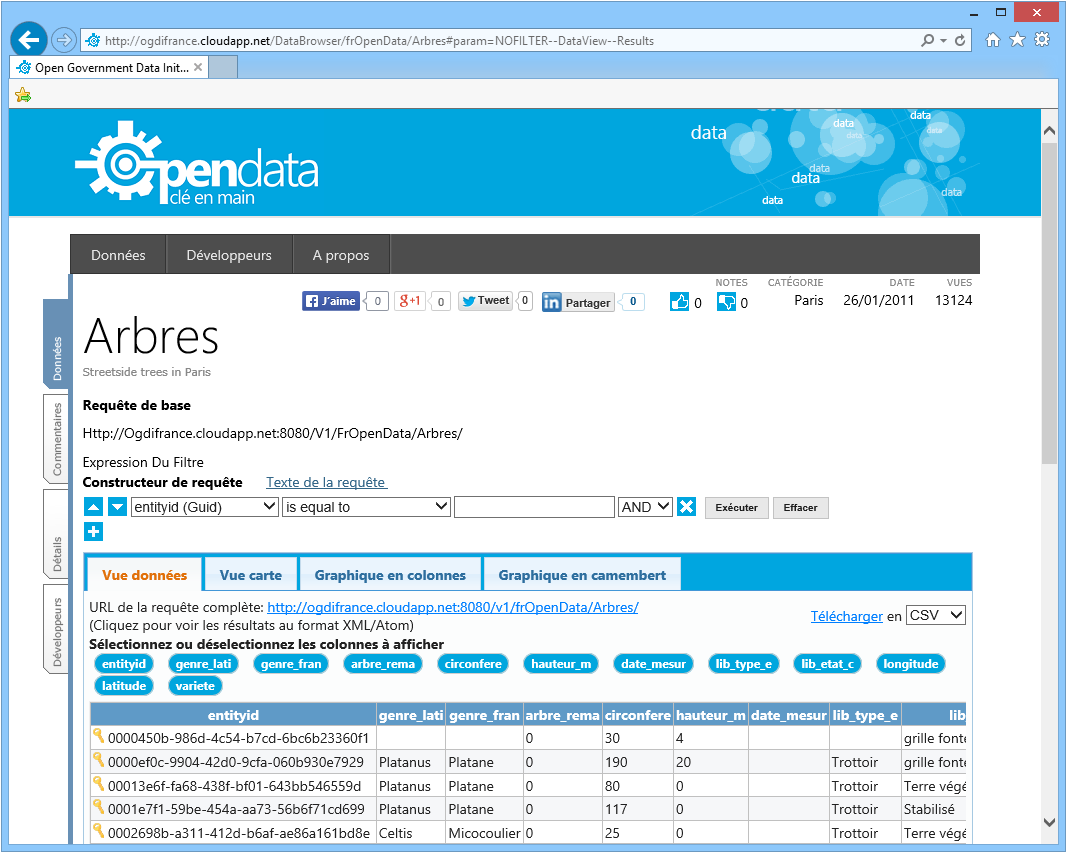
Cette solution s’articule autour de trois composantes de base :

1. Kit de développement logiciel interactif : DataBrowser.
2. Service de données.
3. Utilitaires de chargement de données.

Il convient également de noter enfin qu’un module Drupal est disponible pour intégrer OGDI DataLab avec Drupal, présentant les données ainsi mises à disposition sous forme à la fois de tableaux complètement fonctionnel et de cartes Bing. Ce module OGDI Field est téléchargeable directement depuis le [site Drupal](http://drupal.org/project/ogdi_field)[[17]](#footnote-17).

## Kit de développement logiciel interactif (DataBrowser)

Le Kit de développement logiciel interactif est un site Web à destination des développeurs (et des utilisateurs finaux). Il consomme les informations publiques mises à disposition par l’entrepôt de données en utilisant le Service de données (décrit après).

****

Il s’agit d’une application Web [ASP.NET MVC 4.0](http://www.asp.net/mvc/mvc4)[[18]](#footnote-18) conforme à l’architecture Modèle-Vue-Contrôleur (MVC). L’architecture MVC est une façon d'organiser une interface graphique, mais également et surtout une façon d’architecturer un code source en respectant le principe de responsabilité unique, le principe de séparation des préoccupations et répondre aux problématiques de testabilité du code. Ce paradigme divise l’interface de l’application en un modèle (modèle de données), une vue (présentation, interface utilisateur) et un contrôleur (logique de contrôle, gestion des événements, synchronisation), chacun ayant un rôle précis dans l'interface.

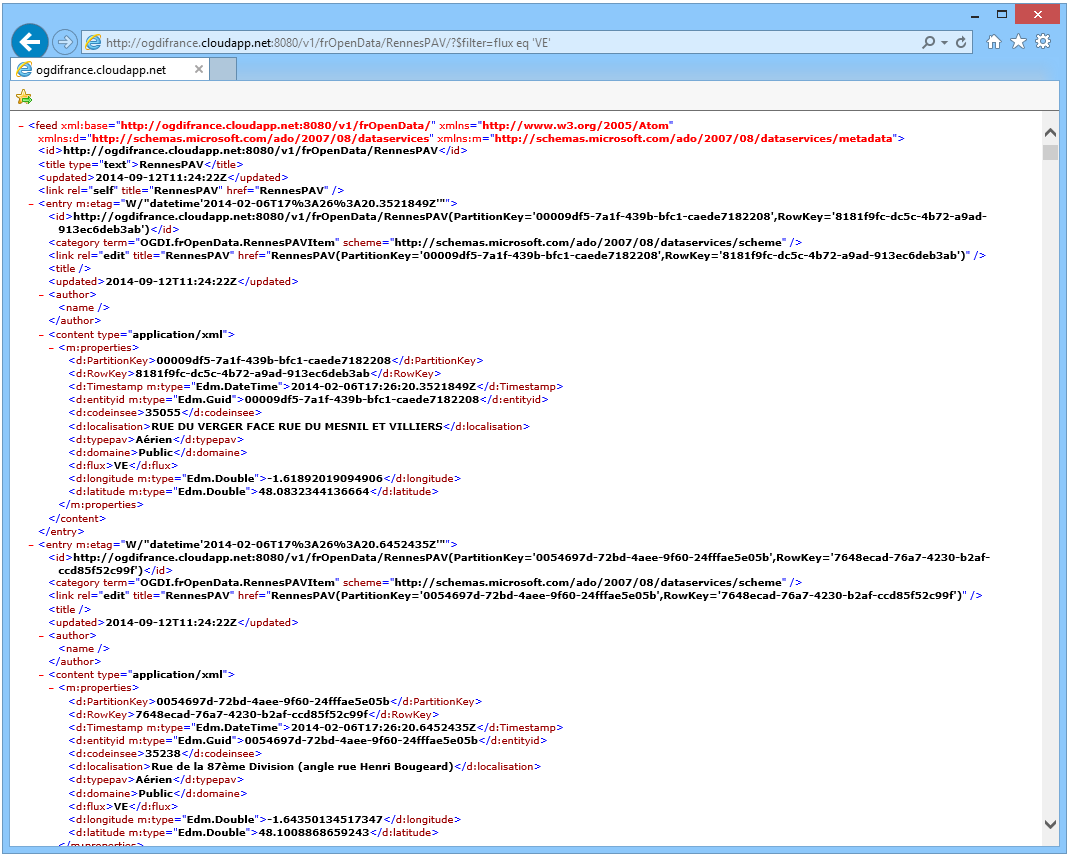
L’application Web utilise [jQuery](http://jquery.com/)[[19]](#footnote-19) et une variété d'autres composants open source et permet aux utilisateurs de parcourir et d'interroger les données publiées. Les données peuvent être commodément visualisées dans des formats largement utilisés et reconnaissables tels que des tableaux, des cartes, des graphiques à barres ou des diagrammes circulaires. Ainsi, au lieu de télécharger un fichier et d’en étudier le contenu, les utilisateurs finaux peuvent interagir directement avec des outils visuels conviviaux qui présentent les données complexes d'une manière plus significative, intuitive et agréable. On peut ainsi citer les extensions de Microsoft Excel dans le cadre de l’offre [Power BI pour Office 365](http://office.microsoft.com/en-us/office365-sharepoint-online-enterprise-help/power-bi-for-office-365-overview-and-learning-HA104103581.aspx)[[20]](#footnote-20) de décisionnel (Business Intelligence) libre-service qui permet de charger ces données ainsi exposées et les mettre à disposition d'objets de visualisation des données en 2D et 3D, comme des tableaux croisés dynamiques, dans une feuille de travail d’un classeur Excel.

Au-delà de la capacité de navigation dynamique dans les données, les développeurs peuvent rapidement apprendre comment utiliser les données publiées exposées par le socle OGDI DataLab depuis leurs propres applications à partir d’exemples de code prêts à l’exécution sur le site. Ces exemples sont disponibles dans une variété de langages et de bibliothèques/Frameworks largement utilisés sur le Web, incluant, entre autres, .NET, Java, JavaScript et Node.js, Flash, Flex, PHP, Python, Ruby, Silverlight, Objective-C (iOS), Android.

## Service de données (DataService)

Le Service de données expose un service Web RESTful OData pour requêter, et consommer les données dans le format du protocole OData, et ainsi pour les exposer pour la programmation.

Le document Utiliser OGDI DataLab décrit le format de base des URLs pour invoquer ce Service de données.



## Configurateur de catalogues (DataConfig)

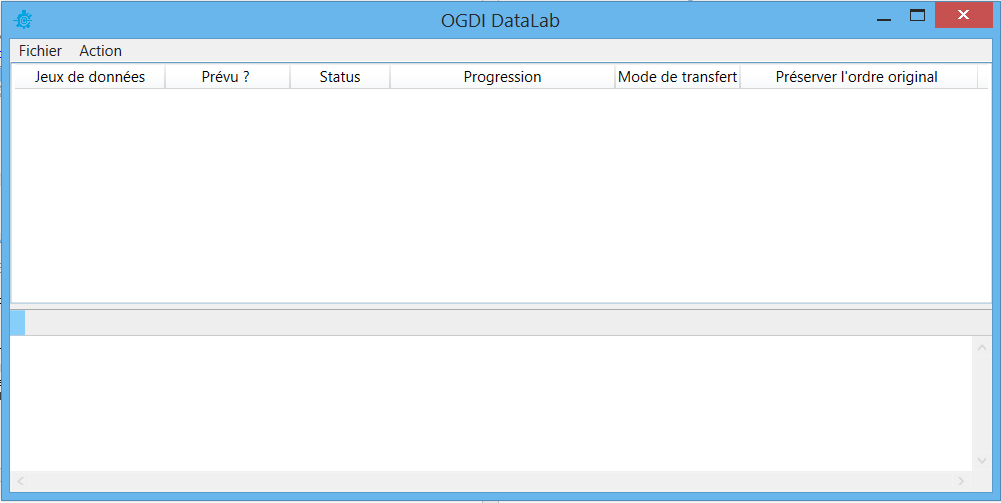
Un utilitaire Web de configuration (initiale) (de catalogues (de jeux)) de données ouvertes permet de définir au niveau d’un compte de stockage Azure la configuration du Service de données. Celle-ci référence en particulier pour l’entrepôt de données du service de données les différents comptes de stockage Azure utilisés pour le stockage (des catalogues de jeux) des données de la plateforme OGDI DataLab.



## Utilitaires de chargement de données (DataLoader)

Les utilitaires de chargement de données OGDI DataLab (graphiques et en mode console) sont destinés à accompagner la mise en œuvre rapide d’OGDI DataLab pour en profiter pleinement.

Un utilitaire en mode console prend des données au format CSV ou KML et les publie au niveau d’une mise en œuvre du service de données. Dans le processus de chargement des données, l'utilitaire peut créer un nouvel ensemble de données, ajouter des données ou mettre à jour les données dans un ensemble de données déjà publié. Étant donné que l'utilitaire est une application console qui est contrôlée par des paramètres en ligne de commande, le chargement de données peut être automatisé à l'aide de scripts et intégrer ainsi les processus ETL (*Extract, Transform, and Load*) existants. Une application graphique permet de réaliser les mêmes opérations au travers d’une interface claire et intuitive pour l’utilisateur.



Le document Mettre à disposition des jeux de données avec OGDI DataLab précise les modalités de chargement de données.

# Configuration requise pour OGDI DataLab

## Configuration système

L’accélérateur OGDI DataLab de la solution Open Data Clé en main a été conçu pour l’environnement Azure. Les prérequis exigés pour cet accélérateur sont donc ceux de cet environnement.

Remarque Ces prérequis avec les correctifs logiciels que cela suppose, peuvent bénéficier d’une installation automatisée de l’ensemble via le programme d’installation de la plateforme Web Microsoft 5.0. Celui-ci simplifie le téléchargement, l'installation et la tenue à jour des composants de la plate-forme Web Microsoft, notamment Internet Information Services (IIS) et ses extensions, SQL Server Express, .NET Framework, Visual Web Developer, etc. Il permet également d'installer des applications Web ASP.NET et PHP open source populaires.

Le programme d’installation de la plateforme Web Microsoft doit être installé préalablement depuis <http://www.microsoft.com/web/downloads/platform.aspx>.

En termes de systèmes d’exploitation, il s’agit en particulier de :

* Microsoft Windows 7 (32 bits ou 64 bits) ou Windows Server 2008 R2 (64 bits) avec le [correctif logiciel cumulatif KB977420 pour Windows Communication Foundation](http://support.microsoft.com/kb/977420)[[21]](#footnote-21).
* Microsoft Windows 8 (32 bits ou 64 bits) ou Windows Server 2012.

Les éléments suivants doivent être installés :

* Microsoft Internet Information Services (IIS) 7.0 ou ultérieur (avec ASP.NET, Activation WCF http, Contenu statique, Console de gestion IIS et éventuellement CGI),
* Microsoft .NET Framework 4.5,

Remarque Azure requiert Microsoft .NET Framework 3.5 (ou ultérieur).

* Microsoft Visual Studio 2013 (ou Microsoft Visual Studio 2013 Express).
* [Kit de développement logiciel Windows Azure 2.4 pour .NET (août 2014)](http://www.microsoft.com/en-us/download/details.aspx?id=43709)[[22]](#footnote-22).

Le Microsoft .NET Framework 4.5 est le modèle de programmation de code managé pour la plateforme Windows. Il combine la puissance du .NET Framework 2.0 avec de nouvelles techniques pour le développement d’applications qui communiquent très facilement au-delà des frontières technologiques, et qui sont capables de prendre en charge un large éventail de processus métiers. Parmi ces innovations se trouve notamment WCF (Windows Communication Foundation).

Auparavant nommé « Indigo », WCF constitue le cadre unifié Microsoft pour le développement d’applications distribuées sécurisées, fiables, transactionnelles et interopérables. Parmi beaucoup d’autres choses, WCF offre une implémentation complète des architectures de type REST au travers de WCF Data Services, technologie auparavant nommé « Astoria » puis ADO.NET Data Services. Il s’agit d’une implémentation du protocole OData par Microsoft.

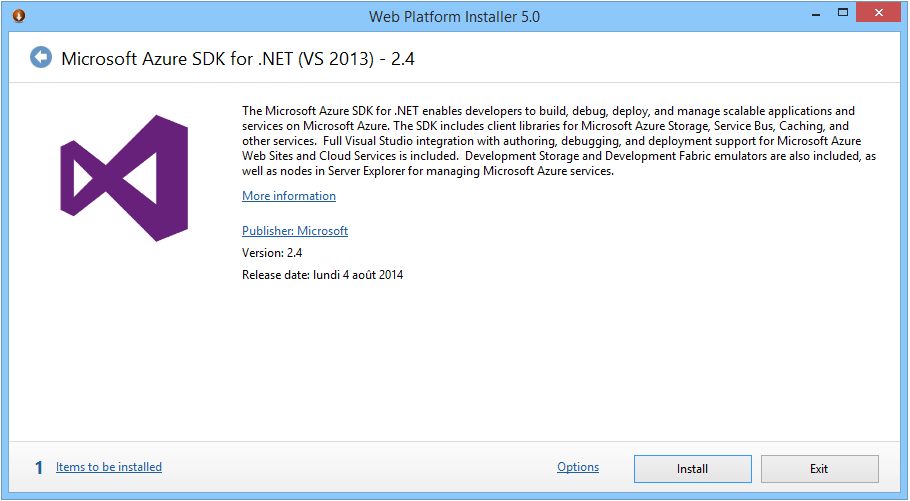
Pour des informations complémentaires, veuillez consulter le Centre de développement MSDN éponyme dédié à l’adresse <http://msdn.microsoft.com/fr-fr/netframework/aa663324.aspx>. Plus spécifiquement, vous pouvez consulter la page dédiée à WCF Data Services à l’adresse <http://msdn.microsoft.com/fr-fr/data/bb931106.aspx> pour la création et la consommation de services OData sur le Web.

OGDI DataLab repose sur la version 4.5 du Microsoft .NET Framework.

Le package redistribuable Microsoft .NET Framework 4.5 est disponible par téléchargement gratuit à l’adresse <http://www.microsoft.com/fr-fr/download/details.aspx?id=30653>.

Vous pouvez télécharger Microsoft Visual Studio Express 2013 for Web gratuitement sur <http://www.visualstudio.com/products/visual-studio-express-vs>.

Le Kit de développement logiciel (SDK) Azure 2.4 pour .NET (aôut 2014) est disponible en téléchargement à l’adresse <http://azure.microsoft.com/fr-fr/downloads/?fb=fr-fr>. cliquez sur le lien [Installation de VS 2013](http://go.microsoft.com/fwlink/p/?linkid=323510&clcid=0x40c) dans la rubrique .NET. Le programme d’installation de la plateforme Web Microsoft 5.0 prend alors en charge installation automatisée du kit de développement avec l’installation des prérequis et correctifs logiciels que cela suppose.



Remarque L’installation manuelle du Kit est disponible à partir de l’adresse <http://www.microsoft.com/en-us/download/details.aspx?id=43709>.

Le Kit de développement logiciel Azure comprend le package Azure Tools pour Microsoft Visual Studio 2013 qui étend Microsoft Visual Studio 2013 pour permettre la création, la configuration, la génération, le débogage, l'exécution, le packaging et le déploiement d'applications et de services Web évolutifs sur Azure.

Remarque Pour plus de détails, vous pouvez consulter la page [Azure pour les développeurs .NET](http://azure.microsoft.com/fr-fr/downloads/dotnet-sdk-24/?WT.mc_id=DOTNET_SDK_24)[[23]](#footnote-23). Cette page liste les paquets additionnels qui peuvent être installés à l’image de la [mise à jour 3](http://go.microsoft.com/fwlink/p/?LinkId=403040)[[24]](#footnote-24) pour Visual Studio 2013, les applets de commandes Azure PowerShell, etc.

## Comment installer les composants système nécessaires

Vous trouverez ci-dessous la procédure d’installation succincte pour la mise en place d’un environnement de développement et de test pour l’accélérateur OGDI DataLab sur une plateforme Windows 7 ou Windows Server 2008 R2 ou ultérieure :

1. Installer Internet Information Services (IIS) 8.0,
2. Installer le Microsoft Framework .NET 4.5,
3. Installer Microsoft Visual Studio 2013,
4. Enfin, utiliser le programme d’installation de la plateforme Web Microsoft 5.0 pour installer Kit de développement logiciel (SDK) Azure 2.4 pour .NET (aôut 2014), avec leurs prérequis et correctifs logiciels associés.

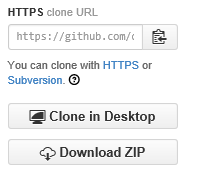
# Obtention des sources d’OGDI DataLab

L’accélérateur OGDI DataLab de la solution communautaire Open Data Clé en main est disponible sur la forge GitHub à l’adresse <https://github.com/openlab/OGDI-DataLab>.

Les solutions communautaires évoluant avec l’investissement de la communauté, il est souhaitable de vous assurer que vous disposez de la dernière version, ne serait-ce que pour bénéficier, le cas échéant, de l’ensemble des correctifs.

[image](https://github.com/openlab/DataLab)

Pour télécharger les sources depuis la forge GitHub, le moyen le plus simple et rapide est de cliquer directement sur **Download ZIP**.



S’il constitue le moyen le plus simple, les autres options se révèlent plus adaptées pour mettre en place une gestion de versions.



A ce titre, nous vous invitons à consulter les billets [OGDI DataLab sur GitHub ! Impliquez-vous dans la communauté](http://blogs.msdn.com/b/ogdifrance/archive/2012/05/14/ogdi-datalab-sur-github-impliquez-vous-dans-la-communaut-233.aspx)[[25]](#footnote-25) et [GitHub for Windows](http://blogs.msdn.com/b/ogdifrance/archive/2012/06/14/github-for-windows.aspx)[[26]](#footnote-26), tous deux présents sur le blog MSDN [Open Data France](http://aka.ms/OpenDataFrance)[[27]](#footnote-27).

Par ailleurs, L’environnement Visual Studio 2013 propose une intégration directe avec GitHub.

Remarque La version précédente 2012 supposait d’installer avec les [outils Visual Studio pour Git](http://visualstudiogallery.msdn.microsoft.com/abafc7d6-dcaa-40f4-8a5e-d6724bdb980c)[[28]](#footnote-28). L’article [Integrating and Using Github in Visual Studio 2012](http://www.codeproject.com/Articles/581907/IntegratingplusandplusUsingplusGithubplusinplusVis)[[29]](#footnote-29) vous propose un guide pas-à-pas dans le contexte de cette version.

Quelle que soit l’approche retenue, et une fois, les sources téléchargées, nous vous invitons à passer à la section suivante.

# Organisation du code source d’OGDI DataLab

L’intégralité du code source des sites, services et composants OGDI DataLab est située dans le sous-répertoire *Source* du répertoire d’extraction d’OGDI DataLab.

Compte tenu de ce qui précède, ce répertoire contient 6 sous-dossiers principaux :

1. Le dossier *DataBrowser* correspondant au Kit de développement logiciel interactif. Il regroupe différents projets qui constituent, au final, l’application Web ASP.NET MVC3 destinée à l’environnement Azure et permettant de visualiser les données publiées;
2. Le dossier *DataService* correspondant au Service de données. Il regroupe différents projets relatifs à la mise en œuvre de ces services Web RESTful ;
3. Le dossier *DataConfig* correspondant à un utilitaire Web de configuration (initiale) (de catalogues (de jeux)) de données ouvertes pour le compte de stockage Azure contenant la configuration OGDI DataLab ;
4. Le dossier *DataLoader* correspondant aux Utilitaires de chargement de données. Il regroupe différents projets relatifs ;
5. Le dossier *DAISY* correspondant au plugin permettant de retourner des données stockées dans Azure au format DAISY (*Digital Access Information SYstem*) ;
6. Le dossier *Shared* correspondant à un dossier partagé contenant 2 projets utilisés par le reste de la solution, à savoir *Ogdi.Azure* et *Ogdi.Config* qui servent essentiellement de point d’entrée au compte de stockage) ainsi que plusieurs bibliothèques utilisées par le reste de la solution.

La suite de cette section décrit les projets/éléments de configuration constituants de ces différents sous-dossiers de façon à mieux cerner la plateforme ainsi proposée par OGDI DataLab et ainsi vous l’accaparer plus facilement en vue par exemple de sa personnalisation (légère ou avancée).

Tous les sites, services et composants afférents sont écrits en C#. Ils sont disponibles pour la version 4.5 du Microsoft Framework .NET. Les fichiers de solution associés et présents dans ces dossiers sont destinés à l’environnement de développement Visual Studio 2013.

## Dossier DataBrowser (Kit de développement logiciel interactif)

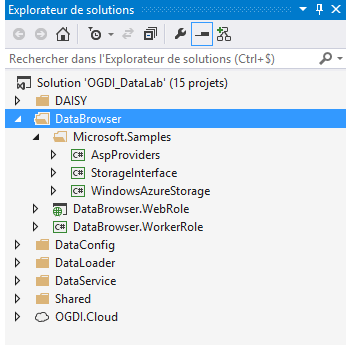
Cette première composante de la solution correspond au kit de développement logiciel interactif et comprend en particulier l’application Web [ASP.NET MVC 4](http://www.asp.net/mvc/mvc4)[[30]](#footnote-30) destinée à l’environnement Azure.

Le document Déployer OGDI DataLab dans Azure précise les modalités de publication du kit de développement logiciel interactif dans Azure.

Remarque importante La publication requiert un compte et une souscription active avec Azure. Vous pouvez accéder à Azure avec un paiement à l’usage, sans engagement, ou bien des forfaits comme décrit à l’adresse <http://azure.microsoft.com/fr-fr/pricing/purchase-options/>.

Remarque Il est possible de déployer le kit de développement logiciel interactif localement en entreprise moyennent une adaptation légère de la configuration de la solution.

### Organisation projet

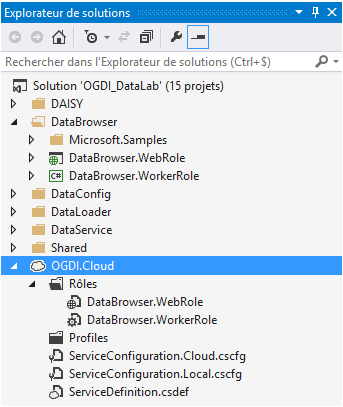
* 1. Le kit de développement logiciel interactif est composé des 6 projets suivants :
  2. 

| **Projet** | **Description** |
| --- | --- |
| AspProviders | Ce projet contient les différents fournisseurs (providers) ASP.NET en matière de sécurité, à savoir fournisseurs d’authentification (membership provider), d’appartenances (role provider), de profil (profile provider) et d’état de session *(*sessionstate provider), adaptés à un contexte Azure (solution architecturée avec les tables d’Azure pour stocker les différentes données).  La technologie ASP.NET 2.0 et ultérieure propose, en effet, une architecture modulaire extensible en termes de mécanisme de sécurité notamment au travers de la définition de fournisseurs. A ce titre, nous vous conseillons également la lecture de l’article [ASP.NET 2.0 Provider Model: Introduction to the Provider Model](http://msdn.microsoft.com/en-us/library/aa479030.aspx)[[31]](#footnote-31) sur le modèle de fournisseurs. Le billet [ASP.NET 2.0 Membership, Roles, Forms Authentication, and Security Resources](http://weblogs.asp.net/scottgu/archive/2006/02/24/ASP.NET-2.0-Membership_2C00_-Roles_2C00_-Forms-Authentication_2C00_-and-Security-Resources-.aspx)[[32]](#footnote-32) regroupe également un ensemble de ressources qui vous permet de disposer rapidement d’une bonne vue d’ensemble des possibilités ainsi offertes. |
| StorageInterface | Ce projet fournit la classe abstraite *IsdkStorageProviderInterface* qui définit des méthodes pour consommer en format XML, DAISY ou CSV les données ou métadonnées stockées dans les tables Azure. |
| WindowsAzureStorage | Ce projet fournit la classe *IsdkWindowsAzureStorageProvider* qui est une implémentation de la classe abstraite *IsdkStorageproviderInterface* du projet *StorageInterface* (Cf. ci-dessus). |
| DataBrowser.WebRole | Ce projet contient le site web frontal en ASP.NET MVC permettant de visualiser les données stockées dans les tables Azure sous forme de tableaux, cartes ou graphiques. |
| DataBrowser.Cloud | Ce projet standard Azure est configuré pour supporter les deux rôles *DataBrowser.WebRole* et *DataBrowser.WorkerRole* (Cf. ci-dessus et ci-après). Ce projet contient la configuration de ces deux rôles qui composent l’application Cloud à déployer dans Azure.  Il inclut le fichier de définition de service *ServiceDefinition.csdef* qui contient les métadonnées nécessaires à Azure pour comprendre les exigences de l’application, tels que les deux rôles utilisés, leur niveau de confiance, les éventuels points de terminaison exposés par chaque rôle, les exigences de stockage local et les éventuels certificats utilisés par les rôles. La définition de service établit également les paramètres de configuration spécifiques à l'application.  Les fichiers de configuration de service *ServiceConfiguration.Cloud.cscfg et ServiceConfiguration.Local.cscfg* spécifient le nombre d'instances à exécuter pour le ou les rôles configurés dans le projet (ici deux) et définit la valeur des paramètres de configuration définis dans le fichier de définition de service. Cette séparation entre la définition et la configuration de service vous permet de mettre à jour les paramètres d'une application en cours d'exécution en téléchargeant un nouveau fichier de configuration de service.  Le nœud **Roles** dans le projet de service Cloud vous permet de configurer quels rôles le service inclut (web, travailleur (worker) ou les deux) ainsi que les projets à associer à ces rôles. L’ajout et la configuration des rôles au travers de ce nœud **Roles** mettra à jour les fichiers *ServiceDefinition.csdef,* ServiceConfiguration.Cloud.cscfg et ServiceConfiguration.Local*.cscfg* précédents.  Ce projet sert directement pour publier dans Azure le paquet complet *DataBrowser*.  Dans un souci de réduction des coûts dans Azure ce projet exposera aussi l’application *DataService.WebRole* permettant de fournir les données (Cf. ci-après). Cette application sera exposée, par défaut, sur le port 8080 du même rôle que *DataBrowser.WebRole.*  L’ensemble de la configuration est définie dans le fichier ServiceDefinition.csdef notamment dans les parties <Sites> et <Endpoints>. |
| DataBrowser.WorkerRole | Ce projet de type Service est utilisé en interne dans Azure pour convertir de façon asynchrone les données stockées dans les tables Azure en fichiers au format DAISY et CSV qui seront ensuite stockés dans un conteneur Blob (toujours dans Azure). Ces fichiers pourront ensuite être téléchargés en toute transparence à travers une simple URL. |

### Eléments de configuration

L’ensemble des paramètres de configuration du kit de développement logiciel interactif est regroupé au niveau :

* Des fichiers de configuration de service *ServiceConfiguration.Cloud.cscfg* et *ServiceConfiguration.Local.cscfg* du projet Azure *OGDI.Cloud* relatif aux rôles *DataBrowser.WebRole* et *DataBrowser.WorkerRole* ;
* Du fichier *WebConfig* du projet *DataBrowser.WebRole*. Ce fichier est situé à la racine du projet.



Remarque Pour accéder à la valeur d’un paramètre depuis une vue utilisant le moteur de vues Razor pour ASP.NET MVC, il suffit d’utiliser par exemple la syntaxe suivante :

@Ogdi.Azure.Configuration.OgdiConfiguration.GetValue("<*parameter*>")

où <*parameter*> est le nom du paramètre considéré. La classe *Ogdi.Azure.Configuration.OgdiConfiguration* fait partie du projet *Ogdi.Azure* situé dans le dossier *Shared*.

Remarque L’approche adoptée ici pour être étendue à d’autres paramètres qu’une personnalisation de l’accélérateur OGDI DataLab rendrait souhaitable.

Remarque importante La plupart des paramètres des fichiers de configuration *ServiceConfiguration.Cloud.cscfg* et *ServiceConfiguration.Local.cscfg* du projet Azure DataBrowser.Cloud sont dupliqués au niveau des fichiers *Web.Config* et *Web.Release.Config* du projet *DataBrowser.WebRole*, de façon à faciliter le cas échéant un fonctionnement du site Web frontal en ASP.NET MVC indépendamment de l’environnement d’exécution de l’environnement Azure.

La méthode statique *GetValue* de la classe *Ogdi.Azure.Configuration.OgdiConfiguration* permet de faire la distinction entre les situations d’exécution.

public sealed class OgdiConfiguration

{

/// <summary>

/// This method finds the value for the passed key from the AppSettings in XML or Azure role environment.

/// </summary>

/// <param name="key">The value name.</param>

/// <returns>The value.</returns>

public static string GetValue(string key)

{

return

(RoleEnvironment.IsAvailable

? RoleEnvironment.GetConfigurationSettingValue(key)

: null)

?? System.Configuration.ConfigurationManager.AppSettings[key]

?? string.Empty;

}

}

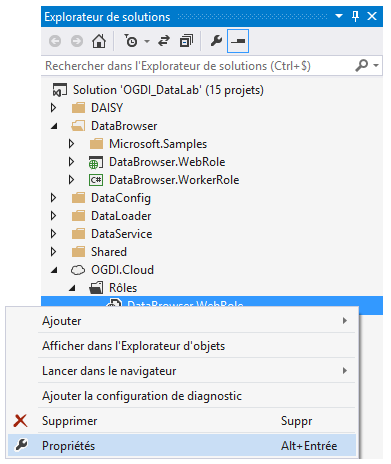
#### Paramètres des fichiers ServiceConfiguration.Cloud.cscfg et ServiceConfiguration.Local.cscfg

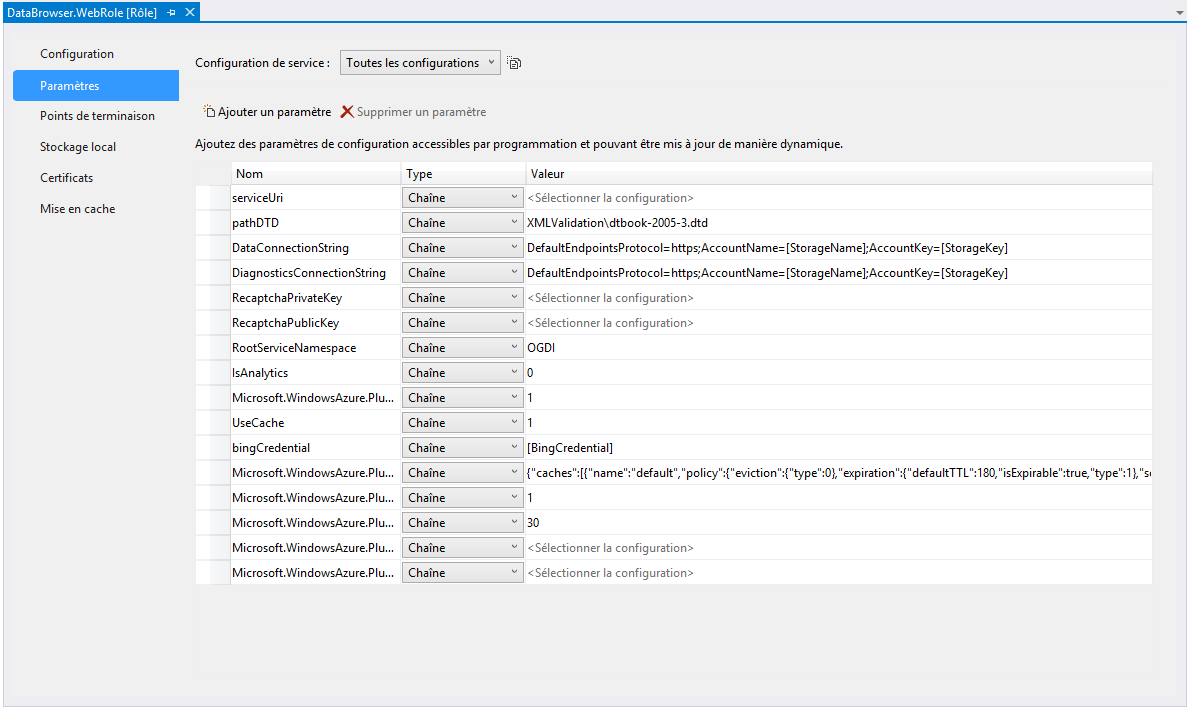
Les fichiers *ServiceConfiguration.Cloud.cscfg* et *ServiceConfiguration.Local.cscfg* contiennent les paramètres suivants :

| **Paramètre** | **Description** |
| --- | --- |
| DiagnosticsConnectionString | Chaine de connexion pointant vers le compte de stockage Azure gérant la configuration OGDI DataLab.  Ce paramètre apparaît deux fois dans les fichiers *ServiceConfiguration.\*.cscfg*,une fois pour chacun des deux rôles *DataBrowser.WebRole* et *DataBrowser.WorkerRole* du projet Azure.  La valeur par défaut est la suivante :  DefaultEndpointsProtocol=https;AccountName=**[StorageName]**; AccountKey=**[StorageKey]** |
| DataConnectionString | Chaine de connexion pointant vers le compte de stockage Azure gérant la configuration OGDI DataLab.  Ce paramètre apparaît deux fois dans les fichiers *ServiceConfiguration.\*.cscfg*,une fois pour chacun des deux rôles *DataBrowser.WebRole* et *DataBrowser.WorkerRole* du projet Azure.  La valeur par défaut est la suivante :  DefaultEndpointsProtocol=https;AccountName=**[StorageName]**; AccountKey=**[StorageKey]** |
| serviceUri | URL du point de terminaison d’un service de données permettant de consommer les informations publiques.  Ce paramètre apparaît deux fois dans les fichiers *ServiceConfiguration.\*.cscfg*,une fois pour chacun des deux rôles *DataBrowser.WebRole* et *DataBrowser.WorkerRole* du projet Azure.  La valeur par défaut est la suivante : [*http://data.govdata.eu:8080/v1/*](http://data.govdata.eu:8080/v1/) |
| SmtpSettings | 1. Paramétrage SMTP (*Simple Mail Transfer Protocol*) pour l’envoi de message. Le paramétrage est structuré comme suit : 2. <hôte>,[<port>],[<ssl>],<de>,[<utilisateur>],[<mot de passe>] 3. où :  * <*hôte*> est le nom FQDN DNS du relai/serveur STMP, * <*port*> le port utilisé, * <*ssl*> est la chaîne de caractère « *ssl* » si la communication doit reposer sur un canal SSL/TLS chiffré et une chaîne de caractères vide dans le cas contraire, * <*de*> l’adresse de l’émetteur du mèl, * et <*utilisateur*>/<*mot de passe*> les crédentités pour l’authentification SMTP.  1. Les paramètres entre cochets ([]) sont optionnels ; les virgules (,) utilisées comme séparateur ne le sont pas. |
| pathDTD | 1. Chemin relatif vers le fichier *Digital Talking Book Document Type Definition*. Ce fichier permet de rendre compatible un texte avec le format DAISY. 2. Ce paramètre apparaît deux fois dans les fichiers *ServiceConfiguration.\*.cscfg*,une fois pour chacun des deux rôles *DataBrowser.WebRole* et *DataBrowser.WorkerRole* du projet Azure. 3. La valeur par défaut est la suivante : *XMLValidation\dtbook-2005-3.dtd* 4. Ce paramètre ne doit pas être modifié. |
| IsAnalytics | Permet d’activer ou de désactiver la partie informations analytiques du site web. Il est à noter que l’activation de la partie informations analytiques peut engendrer des ralentissements lorsque le nombre de visites est important.  0 : Désactivé. Valeur par défaut   1. 1 : Activé |
| RecaptchaPrivateKey | 1. Clé privée pour le captcha. Le jeu de clés publique/privée doit être obtenu via le service reCaptcha à l’adresse <http://recaptcha.net/whyrecaptcha.html>. |
| RecaptchaPupblicKey | 1. Clé publique pour le captcha. Le jeu de clés publique/privée doit être obtenu via le service reCaptcha à l’adresse <http://recaptcha.net/whyrecaptcha.html>. |
| bingCredential | 1. Clé unique à générer pour l'API de cartographie Bing Map. Ceci suppose l'enregistrement du kit de développement logiciel interactif auprès de Bing à l’adresse <https://www.bingmapsportal.com>. |
| UseCache | 1. Permet d’activer ou de désactiver la fonction de Caching sur Azure.   0 : Désactivé. Valeur par défaut   1. 1 : Activé |
| Microsoft.WindowsAzure.Plugins.Caching.CacheSizePercentage | 1. Taille du cache. C’est la quantité maximale de mémoire qui peut être utilisée par le cluster de cache sur les instances de l’ordinateur virtuel. 2. La valeur par défaut est 30. |
| Microsoft.WindowsAzure.Plugins.Diagnostics.ConnectionString | Chaine de connexion pointant vers un compte de stockage Azure. Généralement celui gérant la configuration OGDI DataLab.  La valeur par défaut est la suivante :  DefaultEndpointsProtocol=https;AccountName=**[StorageName]**; AccountKey=**[StorageKey]**  En configuration Local, la valeur doit être :  UseDevelopmentStorage=true |

Les paramètres des rôles *Databrowser.WebRole* et *DataBrowser.WorkerRole* peuvent être modifiés directement depuis le projet *DataBrowser.Cloud*.

Pour ce faire, procédez comme suit :

1. Développez le dossier **Roles** du projet *DataBrowser.Cloud* dans l'Explorateur de solutions, faites un clic droit sur le projet *DataBrowser.WebRole* (ou le projet *DataBrowser.WorkerRole*) et puis sélectionnez **Propriétés**.
   * 1. 
2. Sélectionnez l’onglet **Paramètres.**



1. Modifiez le ou les paramètres souhaités.
2. Appuyez sur les touches CTRL+S pour sauvegarder la ou les modifications de la configuration.

#### Paramètres du fichier Web.Config du projet DataBrowser.WebRole

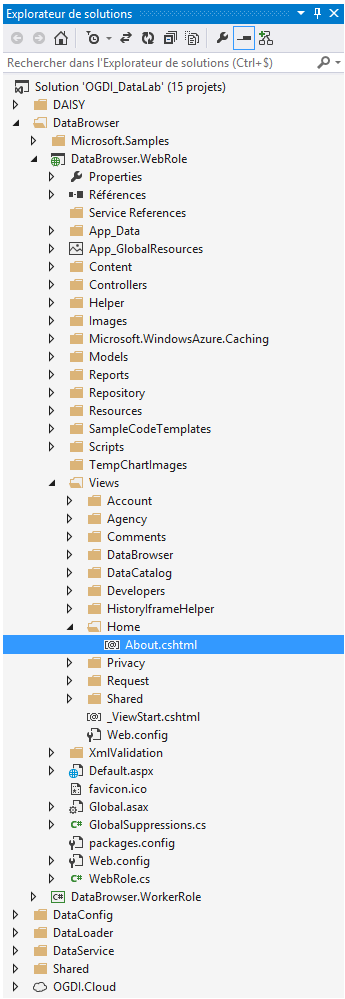
Les fichiers *Web.Config* et *Web.Release.Config* contiennent les paramètres suivants :

| **Paramètre** | **Description** |
| --- | --- |
| serviceUri | URL du point de terminaison d’un service de données permettant de consommer les informations publiques.  Ce paramètre apparaît deux fois dans les fichiers *ServiceConfiguration.\*.cscfg*,une fois pour chacun des deux rôles *DataBrowser.WebRole* et DataBrowser*.WorkerRole* du projet Azure.  La valeur par défaut est la suivante : <http://ogdifrance.cloudapp.net:8080/v1/> |
| pathDTD | 1. Chemin relatif vers le fichier *Digital Talking Book Document Type Definition*. Ce fichier permet de rendre compatible un texte avec le format DAISY. 2. Ce paramètre apparaît deux fois dans les fichiers *ServiceConfiguration.\*.cscfg*,une fois pour chacun des deux rôles *DataBrowser.WebRole* et *DataBrowser.WorkerRole* du projet Azure. 3. La valeur par défaut est la suivante : *XMLValidation\dtbook-2005-3.dtd* 4. Ce paramètre ne doit pas être modifié. |
| DataConnectionString | Chaine de connexion pointant vers le compte de stockage Azure gérant la configuration OGDI DataLab.  La valeur par défaut est la suivante :  DefaultEndpointsProtocol=https;AccountName=**[StorageName]**; AccountKey=**[StorageKey]** |
| DiagnosticsConnectionString | Chaine de connexion pointant vers le compte de stockage Azure gérant la configuration OGDI DataLab.  La valeur par défaut est la suivante :  DefaultEndpointsProtocol=https;AccountName=**[StorageName]**; AccountKey=**[StorageKey]** |
| RecaptchaPrivateKey | 1. Clé privée pour le captcha. Le jeu de clés publique/privée doit être obtenu via le service reCaptcha à l’adresse <http://recaptcha.net/whyrecaptcha.html>. |
| RecaptchaPupblicKey | 1. Clé publique pour le captcha. Le jeu de clés publique/privée doit être obtenu via le service reCaptcha à l’adresse <http://recaptcha.net/whyrecaptcha.html>. |
| IsAnalytics | Permet d’activer ou de désactiver la partie informations analytiques du site web. Il est à noter que l’activation de la partie informations analytiques peut engendrer des ralentissements lorsque le nombre de visites est important.  0 : Désactivé. Valeur par défaut  1 : Activé |
| UseCache | 1. Permet d’activer ou de désactiver la fonction de Caching sur Azure.   0 : Désactivé. Valeur par défaut   1. 1 : Activé |
| bingCredential | 1. Clé unique à générer pour l'API de cartographie Bing Map. Ceci suppose l'enregistrement du kit de développement logiciel interactif auprès de Bing à l’adresse <https://www.bingmapsportal.com>. |

### A propos des vues proposées et de la localisation

Le site Web frontal du kit de développement logiciel interactif ASP.NET MVC est entièrement localisé de façon à s’adapter dynamiquement à la culture de l’internaute.

Ceci s’applique à l’ensemble des vues proposées par le projet *DataBrowser.WebRole*. Prenons à titre d’illustration la page d’accueil dont la vue associée correspondant au fichier *About.cshtml* situé dans le sous dossier *Views\About*.



On peut voir au début du fichier le code suivant :

@Html.Raw(HomeResources.About.AboutPageHeading)

@Html.Raw(HomeResources.About.AboutPageContent)

Ce code tire parti de la localisation en ASP.NET qui est basée sur des fichiers de ressources .*resx*.

En observant l’arborescence du projet *DataBrowser.WebRole*, il convient de noter la présence d’un dossier *Resources*contenant une hiérarchie de sous-dossiers. Cette hiérarchie reprend la hiérarchie de premier niveau du projet. Ainsi, le sous-dossier *Resources\Views\About* vient en correspondance du dossier précédent *Views\About*.

Ce sous-dossier contient les fichiers de ressources relatives à la page/vue considérée.

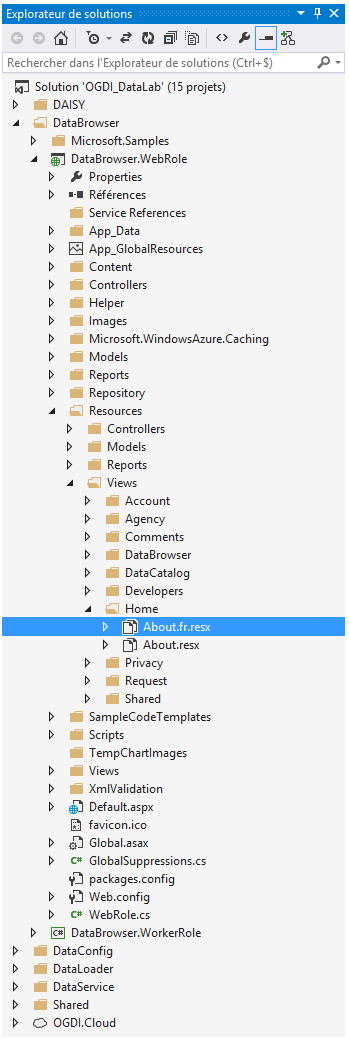
Ces fichiers de ressources respectent, pour pouvoir obtenir au final des contenus localisés, une convention de nommage, à savoir <*nom de fichier*>.<*culture*>.resx, où <*nom de fichier*> est le nom du fichier de la page/vue considérée et <*culture*> sert à indiquer la culture du fichier de ressource.

Il existe deux types de culture :

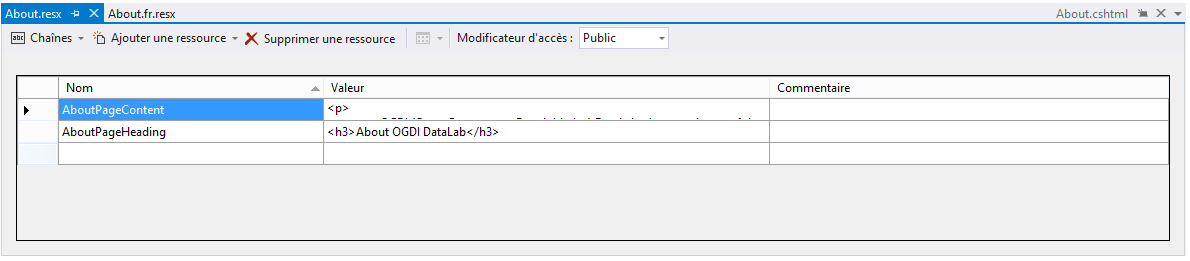
1. Les cultures dites neutres (aussi appelées invariantes). Elles contiennent seulement un code identifiant un langage, par exemple *fr*, *en*, etc.
2. Les cultures dites concrètes qui contient à la fois un code identifiant le langage et aussi un code identifiant la région, par exemple *fr-FR*, ou *fr-CA*, *en-UK* ou *en-US*.

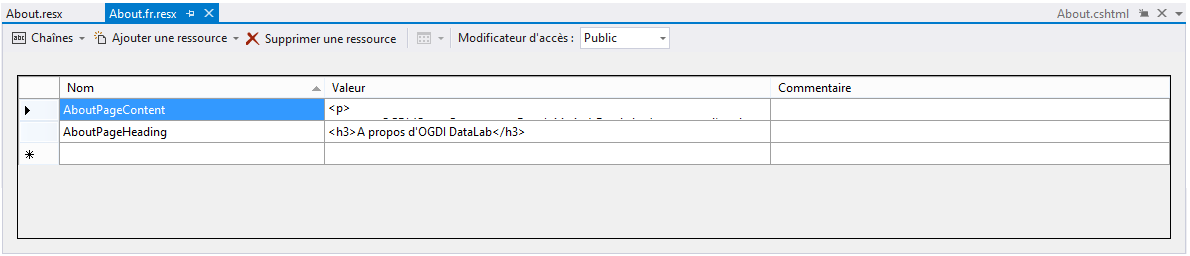
Dans le cas présent, OGDI DataLab propose systématiquement :

1. Un fichier de ressource par défaut, par exemple *About.resx* dans notre illustration,
2. Un fichier de ressource pour la langue française, par exemple *About.fr.resx* dans notre illustration.



Si l’on considère à présent le contenu de ces fichiers de ressources, on peut constater que ces derniers contiennent des paramètres sous forme clé/valeur. On retrouve les mêmes clés à l’identique dans les deux fichiers, seules les valeurs sont traduites.





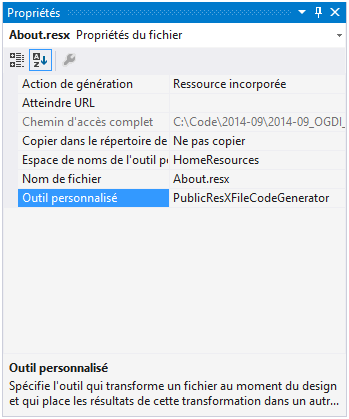
Remarque Pour pouvoir accéder à ces fichiers de ressources, la valeur de la liste déroulante Modificateur d’accès doit être Public. Si la valeur est Internal, le fichier de ressource considéré sera alors invisible du reste du code. La raison en est que les fichiers de ressources sont compilés dans des assemblages satellites à l’assemblage principal qui contient le site Web frontal. Dès lors, un modificateur d’accès de type Internal pour les classes d’accès aux ressources générées automatiquement par Visual Studio n’est donc pas adapté.

Remarque Il convient également de s’assurer, au niveau des propriétés du fichier de ressources que :

. L’attribut Outil personnalisé soit égal à *PublicResxFileCodeGenerator* ; ce qui permet de générer des classes de ressources fortement typées publiques en accord avec le modificateur d’accès précédent.

. L’attribut Espace de noms de l’outil personnalisé comporte un nom « user-friendly » afin de bénéficier d’un espace de noms facilement identifiable quand nous irons pointer vers les ressources dans notre code C#.

Ainsi, on bénéficie directement de la syntaxe <*Espace de noms de l’outil personnalisé*>.<*nom de fichier*>.<*clé*> pour aller chercher une ressource particulière, par exemple *HomeResources.About.AboutPageHeading* dans notre illustration



Compte tenu de ces éléments, se pose maintenant la question « Comment le site Web va-t-il identifier la langue du client et donc se servir du fichier de ressource adéquat pour rendre le contenu approprié ? »

La réponse se trouve dans le fichier *global.asax* du site Web frontal. Ce fichier unique contient une classe héritant de la classe **HttpApplication** de l’espace nom **System.Web**. Cette classe permet de définir toutes les propriétés et les évènements qui sont communs à l’ensemble de l’application ASP.NET MVC.

Dans le cas présent, l’évènement qui nous intéresse est **AcquireRequestState**. Cet évènement se déclenche quand ASP.NET acquiert l’état courant associé à la requête courante, par exemple l’état de la session. Cet évènement s’avère donc particulièrement bien approprié pour aller chercher un contenu localisé.

Dans la classe héritant de **HttpApplication** dans le fichier *global.asax*, on peut donc définir une méthode ***Application\_AcquireRequestState*** qui va s’abonner à l’évènement **AcquireRequestState** et analyser la requête http du navigateur client afin d’en retirer, au niveau de ses en-têtes, la culture attendue en retour. Une fois la culture récupérée sous forme de chaine de caractères, il ne reste plus qu’à créer un objet de type **CultureInfo** et de faire pointer les propriétés **CurrentCulture** et **CurrentUICulture** du thread courant vers cet objet. Et le tour est joué !

protected void Application\_AcquireRequestState(object sender, EventArgs e)

{

if (HttpContext.Current.Request.UserLanguages != null)

{

var culture = HttpContext.Current.Request.UserLanguages[0];

if (!string.IsNullOrEmpty(culture))

{

CultureInfo ci = new CultureInfo(culture);

Thread.CurrentThread.CurrentCulture = ci;

Thread.CurrentThread.CurrentUICulture = ci;

}

}

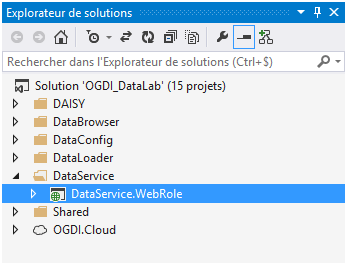
}

Nous vous invitons vivement à reprendre cette approche ainsi décrite pour toute personnalisation/extension d’OGDI DataLab.

## Dossier DataService (Service de données)

Cette seconde composante correspond au Service de données et aux différents services Web RESTful ainsi proposés par le socle. Le document Déployer OGDI DataLab dans Azure précise les modalités de publication du Service de données dans Azure, le document Utiliser OGDI DataLab les modalités d’utilisation avec le protocole OData.

### Organisation projet

* 1. Le Service de données se compose du projet suivant :
  2. 

| **Projet** | **Description** |
| --- | --- |
| WebRole | Ce projet expose les données stockées dans les tables Azure. Les données sont exposées via le protocole OData. |

### Eléments de configuration

L’ensemble des éléments de configuration du Service de données est regroupé dans le projet *DataBrowser.Cloud* au niveau des fichiers de configuration de service *ServiceConfiguration.\*.cscfg* du projet Azure *Services* comme expliqué précédemment.

Cependant il est possible de déployer séparément l’application DataService.WebRole en rajoutant le projet Azure DataService.Cloud dans le dossier DataService et en supprimant la configuration de celui-ci dans le fichier *ServiceDefinition.csdef* du projet *DataBrowser.Cloud*.

Le fichier *ServiceConfiguration.cscfg* du projet ajouté contient les paramètres suivants :

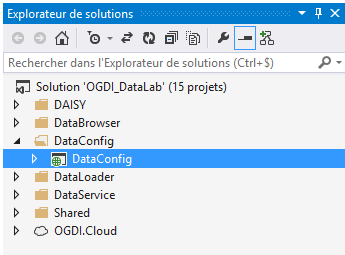
| **Paramètre** | **Description** |
| --- | --- |
| DataConnectionString | Chaine de connexion pointant vers le compte de stockage Azure gérant la configuration OGDI DataLab.  La valeur par défaut est la suivante :  DefaultEndpointsProtocol=https;AccountName=**[StorageName]**; AccountKey=**[StorageKey]** |
| DiagnosticsConnectionString | Chaine de connexion pointant vers le compte de stockage Azure gérant la configuration OGDI DataLab.  La valeur par défaut est la suivante :  DefaultEndpointsProtocol=https;AccountName=**[StorageName]**; AccountKey=**[StorageKey]** |

## Dossier DataConfig (Configurateur de catalogues)

Cette troisième composante correspond à l’utilitaire Web de configuration (initiale) (de catalogues (de jeux)) de données ouvertes pour le compte de stockage Azure contenant la configuration OGDI DataLab ;

### Organisation projet

* 1. Cette composante est composée du projet éponyme suivant :

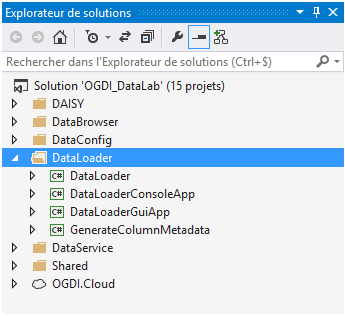


| **Projet** | **Description** |
| --- | --- |
| DataConfig | Ce projet de type site Web met à disposition un simple outil de configuration pour le compte de stockage Azure contenant la configuration OGDI DataLab. |

## Dossier DataLoader (Utilitaires de chargement de données)

Cette quatrième composante correspond aux utilitaires de chargement d’informations publiques. Le document Mettre à disposition des Jeux de données avec OGDI DataLab précise les modalités d’utilisation de ces utilitaires.

### Organisation projet

* 1. Cette dernière composante est composée des 4 projets suivants :
  2. 

| **Projet** | **Description** |
| --- | --- |
| DataLoader | Ce projet de type bibliothèque contient toutes les classes permettant de télécharger un fichier de données texte au format CSV ou KML vers les tables Azure. |
| DataLoaderConsoleApp | Ce projet de type application en ligne de commande s’appuie sur la bibliothèque *DataLoader* pour télécharger un fichier de données texte au format CSV ou KML. Comme il s’agit d’une application en ligne de commande, il est possible de s’en servir pour créer un script automatisé. |
| DataLoaderGuiApp | Ce projet de type application graphique permet d’effectuer les mêmes opérations qu’avec l’application en ligne de commande précédente *DataLoaderConsoleApp* mais, cette fois-ci, avec des interactions avec l’utilisateur. |
| GenerateColumnMetadata | Ce projet de type application en ligne de commande peut être utilisé pour mettre à jour la génération RDF des jeux de données précédemment stockés dans les tables de stockage. |

### Eléments de configuration

Les utilitaires de chargement de données qu’il s’agisse de la version graphique ou en ligne de commande reposent sur la configuration regroupée au niveau du fichier de configuration *App.config* des projets *DataLoaderGuiApp* (version graphique) et *DataLoaderConsoleApp* (version en ligne de commande).

Le fichier *App.config* contient les paramètres suivants :

| **Paramètre** | **Description** |
| --- | --- |
| DataConnectionString | Chaine de connexion pointant vers le compte de stockage Azure gérant les données de la plateforme OGDI DataLab.  La valeur par défaut est la suivante :  DefaultEndpointsProtocol=https;AccountName=**[StorageName]**;AccountKey=**[Storagekey]** |

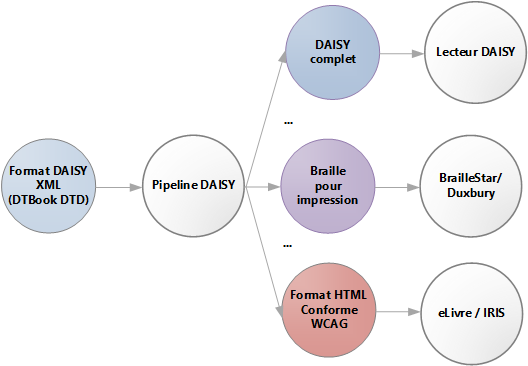
En ce qui concerne l’utilitaire de mise à jour de la génération RDF (*GenerateColumnMetadata*), ce dernier dispose lui aussi d’un fichier *App.config* avec les paramètres suivants :

| **Paramètre** | **Description** |
| --- | --- |
| DataConnectionString | Chaine de connexion pointant vers le compte de stockage Azure gérant la configuration OGDI DataLab.  La valeur par défaut est la suivante :  DefaultEndpointsProtocol=https;AccountName=**[StorageName]**;AccountKey=**[StorageKey]** |
| DefaultDescription | Champ représentant la description par défaut qui sera associée avec les métadonnées des colonnes. Sa valeur peut être changée, mais les deux paramètres {0} et {1} sont obligatoires. {0} représente le nom de la colonne et {1} le nom de l’ensemble de données.  La valeur par défaut est la suivante :  It represents the {0} of a/an {1}. |

## Dossier DAISY

Le format DAISY (*Digital Access Information SYstem*) XML est issu du [consortium DAISY](http://www.daisy.org)[[33]](#footnote-33) qui regroupe 70 associations (notamment l’Association BrailleNet en France) dont le but est, depuis 1996, de rendre tous les documents publiés accessibles aux personnes souffrant de déficiences visuelles, de dyslexie, ou encore de la maladie de Parkinson. Autant de handicaps qui rendent difficiles la lecture, la saisie sur un clavier et même la simple manipulation d'un livre.

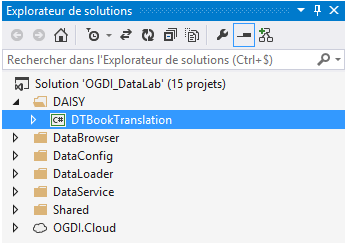
Le format Daisy XML (DTBook DTD) est la base fondatrice du standard DAISY 3, à savoir les [spécifications ANSI/NISO Z39.86 (R2012)](http://www.daisy.org/z3986/2005/Z3986-2005.html?q=z3986/2005/z3986-2005.html)[[34]](#footnote-34) pour les livres audio numériques, utilisée dans le monde entier pour lire et publier des contenus multimédias et constituer des livres numériques audio. Ce format pivot peut ensuite être converti dans de multiples formats au travers notamment du [Pipeline DAISY](http://www.daisy.org/project/pipeline)[[35]](#footnote-35), une suite gratuite sous licence libre proposée par le consortium DAISY.



Les fichiers résultants sont ensuite non seulement lus par des synthétiseurs vocaux pour créer une forme de « braille électronique » mais, grâce au standard DAISY, les utilisateurs peuvent naviguer rapidement par en-tête ou numéro de page, utiliser des index et références, le tout avec des enregistrements audio et du texte classés et synchronisés.

Pour de plus amples informations, nous vous invitons à visionner/écouter le [film DAISY rend la lecture plus facile](http://www.microsoft.com/france/accessibilite/video/20080513-daisy-makes-reading-easier.asx)[[36]](#footnote-36).

Ce dossier contient un sous projet éponyme :

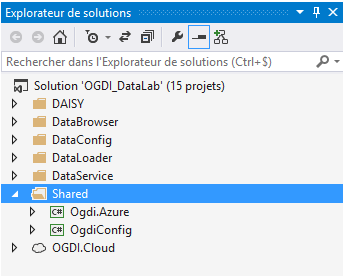


| **Projet** | **Description** |
| --- | --- |
| DAISY | Ce projet de type bibliothèque permet de retourner un ensemble de données stockées dans Azure au format DAISY et ce par le biais de l’application d’une transformation XSL (*Extensible Stylesheet Language*).  La transformation repose sur les fichiers *ConvertToDTBook.xslt* et *Shematron.xsl* situés dans le sous-répertoire *Resources* du projet.  L’assemblage résultant est référencé dans le projet *DataBrowser.WebRole*  du kit de développement logiciel interactif. |

## Dossier Shared

### Organisation projet

Ce dossier contient deux sous-projets d’OGDI DataLab, à savoir *Ogdi.Azure* et *Ogdi.Config* utilisés dans les solutions précédentes :



| **Projet** | **Description** |
| --- | --- |
| Ogdi.Azure | Ce projet de type bibliothèque sert de point d’entrée vers le compte de stockage Azure contenant les données de configuration OGDI DataLab. Il permet notamment d’accéder et d’interagir avec les tables intitulées *AnalyticInfo* et *ViewEntries* qui servent à collecter des statistiques sur les données exposées. |
| Ogdi.Config | Ce projet de type bibliothèque sert également de point d’entrée vers le compte de stockage Azure contenant les données de configuration OGDI DataLab mais permet d’accéder seulement à la table intitulée *AvailableEndpoints* ; celle-ci permet de lister les URL « racines » relatives aux différents jeux de données définies dans le compte de stockage des données de la plateforme OGDI DataLab. |

### Dépendances

Par ailleurs, ce dossier *Shared* est destiné à recevoir également les diverses dépendances du Kit de façon à pouvoir compiler la solution dans son ensemble.

A ce titre, le tableau ci-dessous précise l’ensemble des composants tiers utilisés dans la solution OGDI DataLab. Sauf indication contraire, ces composants doivent être copiés dans le dossier *Shared*. Copiez uniquement les fichiers spécifiés ci-dessous. Veuillez-vous assurer, par ailleurs, de revoir préalablement et d'accepter les termes de licence de chaque composant.

|  |  |  |
| --- | --- | --- |
| **Assembly/Composant** | **Emplacement de destination** | **Remarques** |
| [Newtonsoft.Json.dll](http://json.codeplex.com/releases/view/25040) | %INSTALL\_ROOT%\Source\Shared\ |  |
| [CSharpFormat.dll](http://www.manoli.net/csharpformat/CSharpFormat.zip) | %INSTALL\_ROOT%\Source\Shared\ | Nécessite de reconstruire l’assemblage |
| [System.Threading.dll](http://go.microsoft.com/fwlink/?LinkId=182996) | %INSTALL\_ROOT%\Source\Shared\ | Après installation du paquet, l’assemblage à copier sous le dossier *Reactive Extensions\v1.0.2563.0\Net35folder* du répertoire racine d'installation par défaut. Ce dernier est *%PROGRAM FILES%\Microsoft Cloud Programmability*. |
| [System.Web.DataVisualization et System.Web.DataVisualizationDesign](http://www.microsoft.com/downloads/details.aspx?FamilyId=130F7986-BF49-4FE5-9CA8-910AE6EA442C&displaylang=en) | Référencé à partir de l'emplacement d'installation par défaut | Suppose d’installer le paquet .*msi* |
| [ASP.NET MVC 4.0](http://go.microsoft.com/fwlink/?LinkID=215693) | Référencé à partir de l'emplacement d'installation par défaut | Suppose d’installer le paquet .*msi* |
| [jquery.clipboard.min.js et jquery.clipboard.swf](http://bradleysepos.com/projects/jquery/clipboard/release/jquery.clipboard-2.0.1.zip) | %INSTALL\_ROOT%\Source\DataBrowser\  DataBrowser.WebRole\Scripts |  |
| [LumenWorks.Framework.IO.dll](http://www.codeproject.com/KB/database/CsvReader.aspx) | %INSTALL\_ROOT%\Source\Shared\ | Nécessite de suivre les instructions pour télécharger l’assemblage |

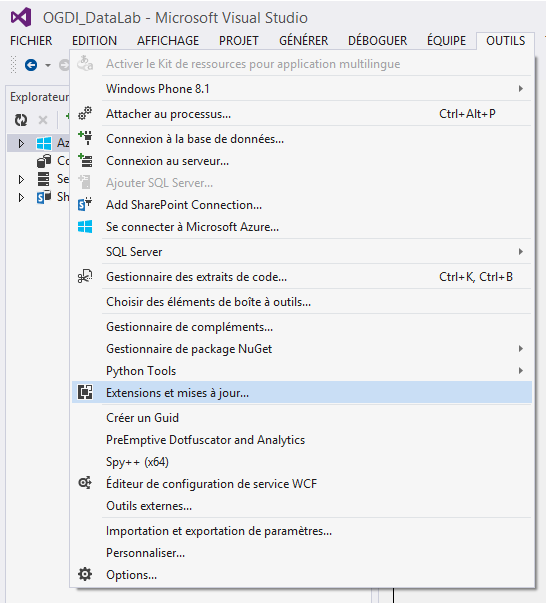
### Résolution des dépendances avec NuGet

Les dépendances précédentes peuvent être résolues avec NuGet. NuGet est une extension qui vient s’ajouter à l’environnement Visual Studio 2013 et qui permet d’intégrer en quelques clics des bibliothèques et des outils sous licence libre à vos projets sous forme de paquets (package) installables.

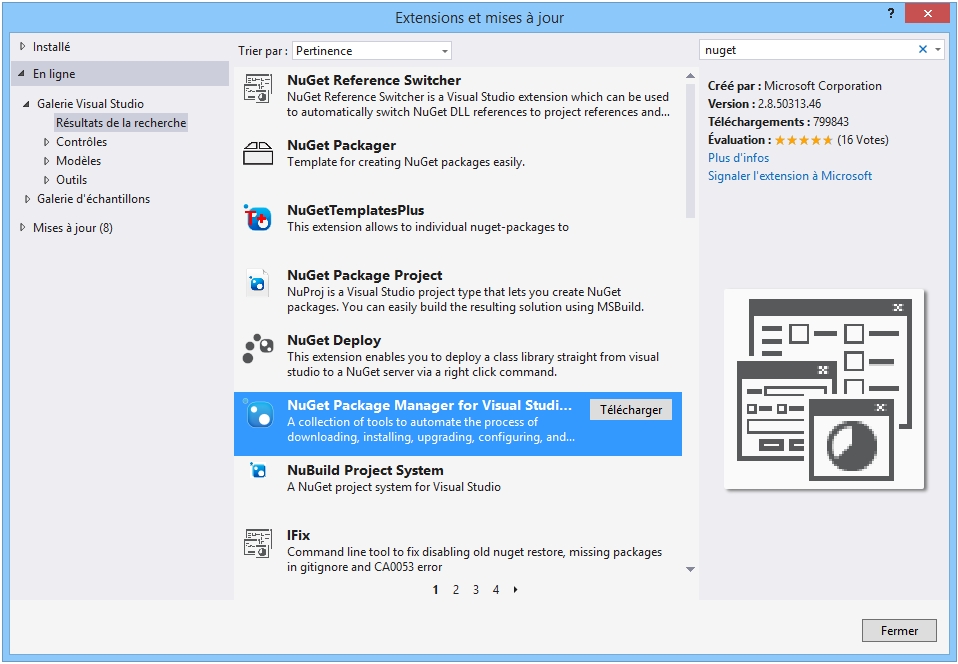
Les dépendances des projets de la solution sont disponibles sous forme de paquets depuis la [galerie en ligne](http://nuget.org/List/Packages)[[37]](#footnote-37) (online gallery) de NuGet.

Pour installer l’extension NuGet, procédez comme suit :

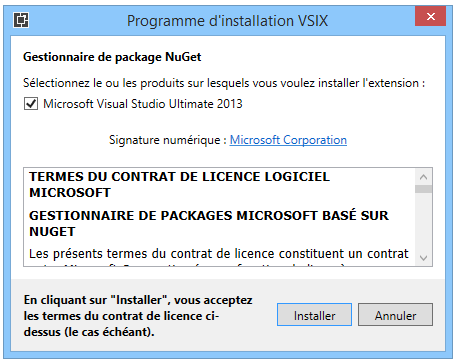
1. Lancez l’environnement de développement Visual Studio 2013.
2. Depuis la barre d’outils, cliquez sur **Outils**, puis **Extensions et mises à jour..**.



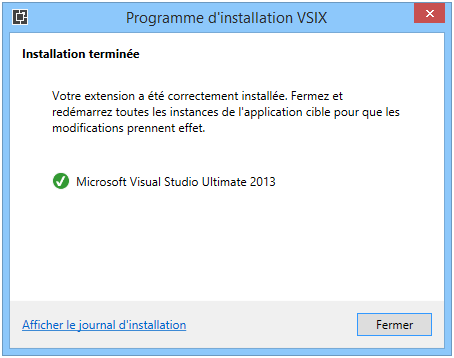
1. Une fois la fenêtre **Extensions et mises à jour**ouverte, sélectionnez l’onglet **En ligne**, puis, dans la barre de recherche, tapez simplement « *nuget* ».



1. Cliquez ensuite sur **NuGet Package Manager for Visual Studio 2013** puis sur le bouton **Télécharger**. Une fois le paquet télécharge, l’installation se lance alors.

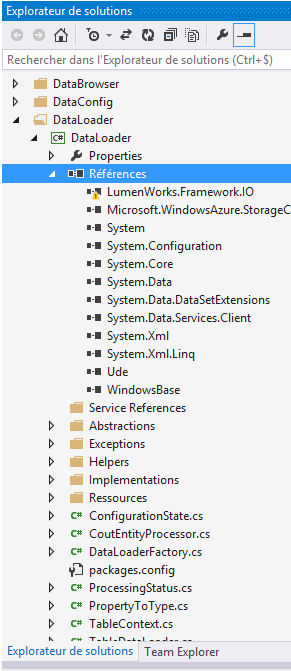


1. Cliquer sur **Installer**. A l’issue de l’installation, un redémarrage de l’environnement est nécessaire.

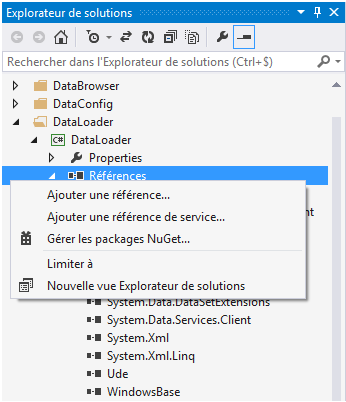


Pour résoudre les (éventuels problèmes de) dépendances précédentes des projets de la solution *Ogdi.sln*, procédez comme suit :

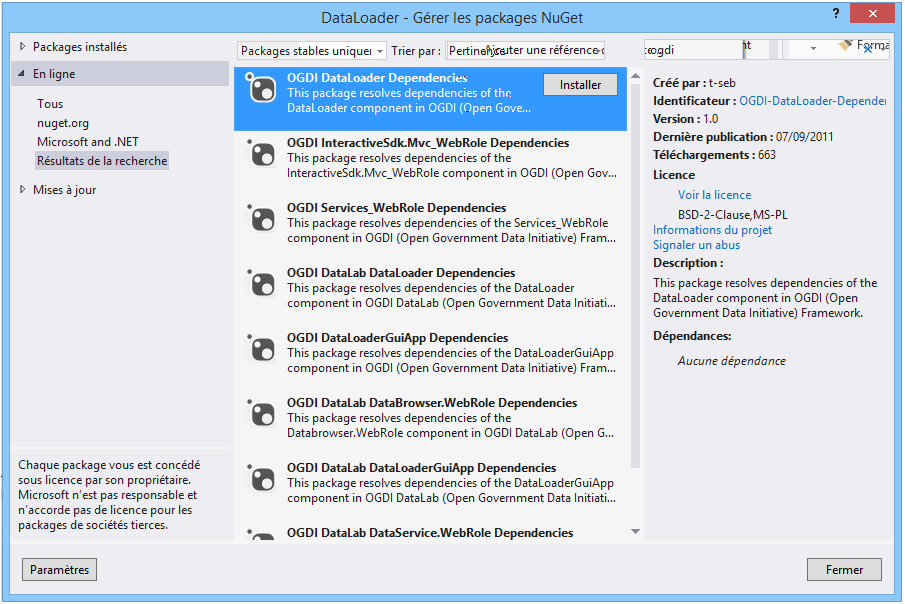
1. Ouvrez Windows Explorer avec des privilèges administratifs et placez-vous dans le sous-répertoire *Source* du répertoire d’installation d’OGDI DataLab.
2. Double-cliquez sur l’icône du fichier *Ogdi.sln* situé à la racine du sous-répertoire *Source* pour ouvrir le fichier dans l’environnement Visual Studio 2013.
3. Depuis le volet **Explorateur de solutions**, sélectionnez par exemple le projet **DataLoader** et dépliez l’onglet **Références**. Dans l’exemple suivant, vous pouvez constater qu’il y a une dépendance non-résolue vers la bibliothèque *LumenWorks.Framework.IO*.



1. Maintenant, faites un clic-droit sur **Références**, et cliquez sur **Gérer les packages NuGet…**



1. Dans la fenêtre **Gérer les packages NuGet** qui s’ouvre, sélectionnez l’onglet **En ligne** puis tapez ensuite « *ogdi* » dans la barre de recherche.

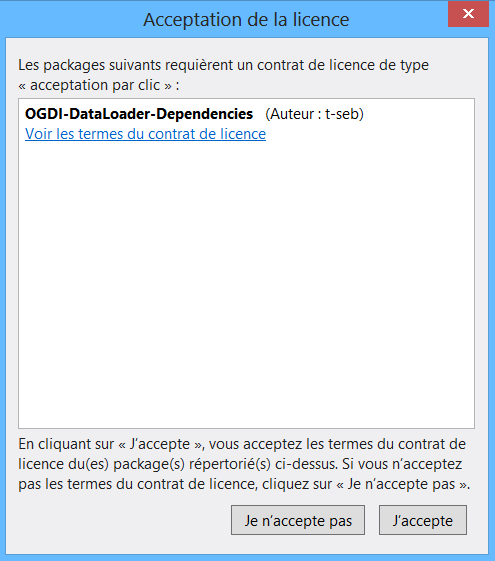


1. Comme vous pouvez le constater, 4 paquets sont disponibles et préfixés par OGDI DataLab, chacun installant les dépendances des projets de la solution OGDI DataLab dont ils portent le nom. Par exemple, dans le cas présent, nous cherchons à résoudre les dépendances du projet **DataLoader*.*** Sélectionnez donc le paquet **OGDI DataLab DataLoader Dependencies**, cliquez sur **Installer**.
2. Dans la boîte de dialogue Acceptation de la licence, cliquez sur le lien Voir les termes du contrat de licence.

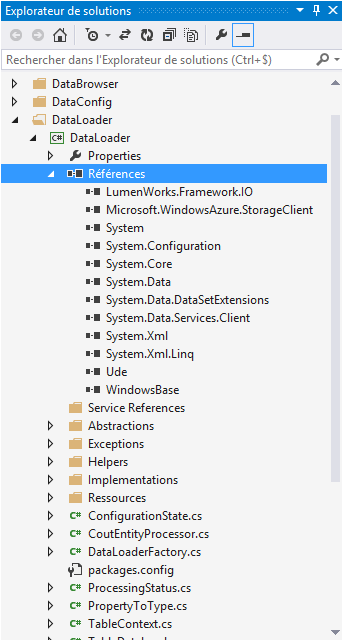
Remarque importante Les composants logiciels contenus dans le paquet sont publiés sous leur différent contrat de licence respectif. Vous devez donc impérativement lire les termes de chaque contrat de licence qui s'applique et accepter les termes de chaque contrat avant d'accepter l'installation de ce paquet. Les liens afférents sont précisés pour faciliter cette opération.

Remarque importante Il apparaît au total 8 paquets : 4 préfixés par *OGDI DataLab* et 4 autres seulement préfixés par *OGDI* ces derniers étant les paquets de l’ancienne version du kit de démarrage OGDI v2a. Il conviendra donc d’utiliser les paquets préfixés par OGDI DataLab dans votre solution

1. Si vous acceptez les termes de chaque contrat qui s’applique, cliquez ensuite sur **J’accepte**afin matérialiser cette acceptation.



1. Cliquez maintenant sur **Fermer** dans la fenêtre **Gérer les packages NuGet**. Au niveau de la liste des références du projet, vous pouvez observer que la précédemment manquante bibliothèque *LumenWorks.Framework.IO*a été téléchargée et que la dépendance a donc été résolue.



La même opération peut être réalisée pour les projets *DataLoaderGuiApp*, *DataBrowser.WebRole* et Data*Service.WebRole* pour résoudre toutes les dépendances vers les bibliothèques sous licence libre utilisée dans le cadre d’OGDI DataLab.

# Compilation des éléments d’OGDI DataLab

Les différentes composantes d’OGDI DataLab peuvent être compilées en utilisant Visual Studio 2013, la commande *msbuild* en ligne de commande ou le script d’automatisation de compilation inclut. Les trois procédures sont décrites dans cette section.

## Compilation des éléments en ligne de commande

Pour compiler une solution particulière via la ligne de commande, veuillez suivre ces étapes :

1. Ouvrez l’invite de commande du SDK avec des privilèges administratifs et placez-vous dans le sous-répertoire *Source* du répertoire d’installation d’OGDI DataLab.
2. Saisissez *msbuild Ogdi.sln* sur la ligne de commande. Les éléments relatifs à la solution considérée sont compilés dans le répertoire */bin/Debug* relatif au répertoire de chaque projet composant la solution.

## Compilation des éléments en utilisant Visual Studio 2013

Pour compiler une solution particulière via Visual Studio 2013, veuillez suivre ces étapes :

1. Ouvrez Windows Explorer avec des privilèges administratifs et placez-vous dans le sous-répertoire *Source* du répertoire d’installation d’OGDI DataLab.
2. Double-cliquez sur l’icône du fichier *Ogdi.sln* situé à la racine du sous-répertoire *Source* pour ouvrir le fichier dans l’environnement Visual Studio 2013.
3. Dans le menu **Générer**, choisissez **Régénérer la solution**. Tous les éléments relatifs à la solution considérée sont compilés dans les répertoires */bin/Debug* relatifs aux répertoires de chaque projet composant la solution.

## Compilation des éléments via le script d’automatisation

Pour compiler l’ensemble de la solution via le script, veuillez suivre ces étapes :

1. Ouvrez Windows Explorer et placez-vous dans le sous répertoire Source du répertoire d’installation d’OGDI DataLab.
2. Double-cliquez sur le fichier *Build.cmd* situé à la racine du sous-répertoire *Source* pour lancer la compilation. Les éléments relatifs à la solution sont compilés dans le répertoire *Deployment* créé lors de la compilation à la racine du dossier *Source*.

# Mise en œuvre des éléments d’OGDI DataLab dans Azure

OGDI DataLab se présente sous la forme de composants logiciels qui permettent à toute entité publique :

1. De récupérer, de s’approprier et de mettre en œuvre très rapidement et à moindre coût une solution ouverte de bout-en-bout de publication d’informations publiques via Azure,
2. Et de publier et d’exposer sur ce socle les jeux de données que l’entité souhaite mettre à disposition des usagers et citoyens.

La figure suivante illustre les éléments d’OGDI DataLab tels que décrits précédemment dans un contexte de mise en œuvre complète et opérationnelle dans Azure :



Dans ce cadre, la mise en œuvre des éléments d’OGDI DataLab hors personnalisation particulière suppose les étapes suivantes :

1. Classification du ou des jeux de données à rendre public en termes de type souhaité de réutilisation :

* Navigation  au sein des données brutes par un utilisateur final ;
* Analyse statistique sur des données brutes ;
* Type d'applications où les informations publiques pourraient être réutilisées.

1. Activation d’une souscription sur Azure. Vous pouvez accéder à Azure avec un paiement à l’usage (sans engagement) et des offres d’abonnement qui permettent de bénéficier d’une réduction par rapport au tarif standard comme décrit à l’adresse <http://azure.microsoft.com/fr-fr/pricing/purchase-options/>. Pour permettre aux clients d’anticiper les coûts d’exploitation, Microsoft met à disposition un outil de calcul en ligne accessible ici : <http://azure.microsoft.com/fr-fr/pricing/calculator/>.
2. Déploiement de la solution dans Azure conformément au document Déployer OGDI DataLab dans Azure. Ce dernier propose une procédure guidée étape par étape pour la mise en œuvre opérationnelle du socle technologique proposé par le Kit de démarrage OGDI DataLab.
3. Publication du ou des jeux de données en suivant les instructions du document Mettre à disposition des jeux de données avec OGDI DataLab pour notamment la préparation des informations publiques de type texte dans un format prêt pour le téléchargement.

L’association d’informations en format binaire y est également abordée.

1. Prise de contact avec les créateurs d’applications potentiellement intéressés par la prise en compte des informations publiques ainsi disponibles dans leurs applications ou dans le développement de nouvelles applications qui utilisent ces informations publiques. Le document Utiliser OGDI DataLab précise comment consommer ces informations publiques.

L’éventail des outils de développement disponibles pour l’accès aux informations publiques et à leur consommation facilite l’implication des acteurs quels qu’ils soient.

Cette étape peut, par ailleurs, conduire à la mise en place d’un concours d’idées avec « la vertu de l’exemple pour amorcer la pompe ». L’accélérateur OGDI DataLab de la solution Open Data Clé en main peut accompagner une telle dynamique en offrant, par exemple, un socle d’accueil dédié aux applications pour Smartphone ainsi proposées dans ce contexte.

1. Solution Open Data clé en main : http://aka.ms/OpenDataCleEnMain [↑](#footnote-ref-1)
2. Site portail Web Microsoft Secteur Public : http://www.microsoft.com/france/entreprises/secteur-public/solutions-pour-secteur-public.aspx [↑](#footnote-ref-2)
3. Microsoft Azure : http://azure.microsoft.com/fr-fr/ [↑](#footnote-ref-3)
4. Protocole OData : http://www.odata.org [↑](#footnote-ref-4)
5. OGDI DataLab sur GitHub : https://github.com/openlab/OGDI-DataLab [↑](#footnote-ref-5)
6. Blog MSDN Open Data France : http://aka.ms/OpenDataFrance/ [↑](#footnote-ref-6)
7. Azure Readiness Content : http://www.microsoft.com/en-us/download/details.aspx?id=8396 [↑](#footnote-ref-7)
8. Microsoft Azure Blog : http://azure.microsoft.com/blog/ [↑](#footnote-ref-8)
9. RFC 5023 The Atom Publishing Protocol : http://tools.ietf.org/html/rfc5023 [↑](#footnote-ref-9)
10. OGC KML : http://www.opengeospatial.org/standards/kml/ [↑](#footnote-ref-10)
11. RFC 4627 The application/json Media Type for JavaScript Object Notation (JSON) : http://tools.ietf.org/html/rfc4627 [↑](#footnote-ref-11)
12. Site Defining Safer JSON-P : http://www.json-p.org/ [↑](#footnote-ref-12)
13. RDF : http://www.w3.org/RDF/ [↑](#footnote-ref-13)
14. OGDI : http://ogdi.codeplex.com [↑](#footnote-ref-14)
15. DataLab : https://github.com/openlab/datalab [↑](#footnote-ref-15)
16. OGDI-DataLab : https://github.com/openlab/OGDI-DataLab [↑](#footnote-ref-16)
17. Module OGDI Field : http://drupal.org/project/ogdi\_field [↑](#footnote-ref-17)
18. ASP.NET MVC 4.0 : http://www.asp.net/mvc/mvc4 [↑](#footnote-ref-18)
19. jQuery : http://jquery.com/ [↑](#footnote-ref-19)
20. Power BI pour Office 365: http://office.microsoft.com/en-us/office365-sharepoint-online-enterprise-help/power-bi-for-office-365-overview-and-learning-HA104103581.aspx [↑](#footnote-ref-20)
21. Correctif logiciel cumulatif KB977420 pour Windows Communication Foundation : http://support.microsoft.com/kb/977420 [↑](#footnote-ref-21)
22. Kit de développement logiciel Windows Azure 2.4 pour .NET (août 2014) : http://www.microsoft.com/en-us/download/details.aspx?id=43709 [↑](#footnote-ref-22)
23. Azure pour les développeurs .NET : http://azure.microsoft.com/fr-fr/downloads/dotnet-sdk-24/?WT.mc\_id=DOTNET\_SDK\_24 [↑](#footnote-ref-23)
24. Mise à jour 3 pour Visual Studio 2013 : http://go.microsoft.com/fwlink/p/?LinkId=403040 [↑](#footnote-ref-24)
25. OGDI DataLab sur GitHub ! Impliquez-vous dans la communauté : http://blogs.msdn.com/b/ogdifrance/archive/2012/05/14/ogdi-datalab-sur-github-impliquez-vous-dans-la-communaut-233.aspx [↑](#footnote-ref-25)
26. GitHub for Windows: http://blogs.msdn.com/b/ogdifrance/archive/2012/06/14/github-for-windows.aspx [↑](#footnote-ref-26)
27. Blog MSDN Open Data France: http://aka.ms/OpenDataFrance/ [↑](#footnote-ref-27)
28. Visual Studio Tools for Git : http://visualstudiogallery.msdn.microsoft.com/abafc7d6-dcaa-40f4-8a5e-d6724bdb980c [↑](#footnote-ref-28)
29. Integrating and Using Github in Visual Studio 2012 : http://www.codeproject.com/Articles/581907/IntegratingplusandplusUsingplusGithubplusinplusVis [↑](#footnote-ref-29)
30. ASP.NET MVC 4 : http://www.asp.net/mvc/mvc4 [↑](#footnote-ref-30)
31. Article ASP.NET 2.0 Provider Model: Introduction to the Provider Model : http://msdn.microsoft.com/en-us/library/aa479030.aspx [↑](#footnote-ref-31)
32. Billet ASP.NET 2.0 Membership, Roles, Forms Authentication, and Security Resources : http://weblogs.asp.net/scottgu/archive/2006/02/24/ASP.NET-2.0-Membership\_2C00\_-Roles\_2C00\_-Forms-Authentication\_2C00\_-and-Security-Resources-.aspx [↑](#footnote-ref-32)
33. Consortium DAISY : http://www.daisy.org [↑](#footnote-ref-33)
34. Standard DAISY/NISO : http://www.daisy.org/z3986/2005/Z3986-2005.html?q=z3986/2005/z3986-2005.html [↑](#footnote-ref-34)
35. DAISY Pipeline : http://www.daisy.org/project/pipeline [↑](#footnote-ref-35)
36. Film DAISY rend la lecture plus facile : http://www.microsoft.com/france/accessibilite/video/20080513-daisy-makes-reading-easier.asx [↑](#footnote-ref-36)
37. Galerie en ligne NuGet : http://nuget.org/List/Packages [↑](#footnote-ref-37)