Journal de l'OSGeo

Le Journal de la Fondation Open Source Geospatial Volume 1 / Mai 2007

Dans ce volume

OSGeo
Your Open Source Compass

FOSS4G - Ouverture des Inscriptions à la Conférence

Nous sommes heureux de vous annoncer l'ouverture des inscriptions en ligne à la Conférence Free and Open Source Software for Geospatial 2007 (FOSS4G 2007). FOSS4G est l'évènement annuel qui réunit les personnes et les sociétés qui créent, utilisent, et gèrent des logiciels géospatiaux open source. Inscrivez-vous dès maintenant en ligne.[1]

Inscrivez-vous avant la date limite du 27 Juillet, pour économiser sur les frais d'inscription ! Tirez profit de l'opportunité que FOSS4G 2007 vous offre, de construire un réseau avec les autres professionnels des données géospatiales, de renouveler d'anciennes relations, et d'en créer de nouvelles.

Pour les dernières mises à jour, l'inscription et/ou la soumission d'une présentation, visitez le site web de la conférence.[2]

OPPORTUNITES D'EXPOSITION & DE SPONSORING

Concernant les opportunités d'exposition et de sponsoring, lisez la page des partenaires [3] ou contactez Paul Ramsey, Président de la Conférence par email.[4]

SOUMETTRE UNE PRESENTATION

Vous pouvez soumettre une présentation en ligne.[5] La date limite pour les soumissions est le 29 Juin 2007.

Les présentations FOSS4G durent 25 minutes, avec 5 minutes de questions/réponses à la fin. Les présentations concernent l'utilisation ou le développement de logiciels géospatiaux opensource. Tout le monde peut soumettre une proposition de présentation et participer à la conférence comme présentateur. Plus d'informations sont disponibles sur la page des présentations sur le site web.

Nous espérons vous voir à Victoria, au Canada en Septembre !

[1] Inscription en ligne : http://www.foss4g2007.org/register/
[2] Site web de la conférence : http://www.foss4g2007.org/
[3] Page des partenaires : http://foss4g2007.org/sponsors
[4] Email Paul Ramsey : pramsey@foss4g2007.org
[5] Soumettez une présentation sur http://www.foss4g2007.org/presentations/

Étude d'intégration

Traitement géospatial sur des serveurs distants via Internet – PyWPS

PyWPS et Embrio

Jáchym Čepický et Lorenzo Becchi, traduit par Vincent Picavet

Le document 05-007r4[6] *de l'OGC décrit un procédé pour mettre à disposition des opérations géospatiales à travers des réseaux en utilisant des services Web. Cet article présente une implémentation : PyWPS. Même si la cible initiale de PyWPS était de rendre accessible depuis des applications Web les modules du SIG GRASS, en pratique il est possible d'utiliser n'importe quel outil en ligne de commande ou qui possède une interface avec le langage de programmation Python. Avec l'aide de WPS il est possible de réaliser des opérations coûteuses en temps de calcul côté serveur, ou de construire votre propre application de Web-SIG tournant dans un navigateur Web. Décrivons maintenant la façon dont PyWPS fonctionne et déterminons si il peut répondre à vos besoins.*

OGC Web Processing Service

Le standard OGC WPS (Web Processing Service) est relativement jeune et pas encore aussi connu que son cousin le WMS (Web Map Service). Nous commencerons donc par en donner un bref aperçu. L'unité de base du WPS est le *process*, une opération géospatiale, avec des entrées et des sorties de type défini. Le client communique avec le serveur à l'aide de trois types de requêtes. La requête peut être envoyée au serveur par HTTP GET avec des paramètres donnés sous la forme de paires clé-valeur (Key-Value Pairs, KVP) ou par HTTP POST, avec des paramètres données dans un fichier XML. Les trois types de requêtes pouvant être envoyées au serveur sont :

GetCapabilities – Le serveur répond en XML, décrivant le serveur, les coûts, une description générale et donnant une liste de process, prêts à être exécutés.

DescribeProcess – Le serveur répond en XML, décrivant en détail les types des entrées et des sorties, pour que le client soit capable d'écrire la

[6] OGC Web Processing Service (WPS) : http://www.opengeospatial.org/standards/requests/28

requête *Execute*.

Execute – Le client demande l'exécution d'une opération géospatiale, avec toutes les données nécessaires. Le serveur lance le process et informe le client (l'utilisateur) de l'avancement.

La suite donne quelques exemples pour illustrer l'utilisation de ces requêtes. Disons que nous voulons faire un calcul de ligne de visée à partir de coordonnées x et y sur un fichier raster, disponible à partir d'un serveur distant. Le nom du process sera *visibility*.

Utilisation basique de l'OGC WPS

Tout d'abord nous avons besoin de déterminer quelles opérations le serveur propose (voir ce lien [7]). A partir du XML renvoyé, il apparait clairement que le process *visibility* est disponible sur le serveur, et le résumé nous dit qu'il correspond au service que nous recherchons. Ensuite, nous avons besoin de savoir quel type d'entrée et de sortie le process nécessite et renvoie. (voir ce lien [8]).

- *x* de type *LiteralValue*
- *y* de type *LiteralValue*
- *maxdist* – distance maximale à l'observateur. De type *LiteralValue*, valeur minimale 0, maximale 5000 mètres.
- *observer* – hauteur de l'observateur. De type *LiteralValue*, valeur minimale 0, maximale 50 mètres.
- *dem* – de type *ComplexValue* – carte raster de modèle numérique de terrain, sur laquelle la ligne de visée doit être calculée.

Maintenant nous pouvons écrire la requête d'exécution, l'envoyer au serveur et attendre les résultats de calcul[9]. Le serveur va télécharger le modèle numérique de terrain, faire le calcul de ligne de visée et renvoyer le raster de résultat au client.

Introduction à PyWPS

PyWPS est un projet relativement jeune, commencé en avril 2006. Le but premier du projet était de faire la connexion entre UMN-MapServer et le SIG GRASS de la manière la plus simple possible, afin de pouvoir créer une véritable application WebSIG, capable d'effectuer par exemple de l'interpolation de données raster, ou diverses analyses de modèles numériques de terrain. Avec le temps, il s'est avéré que même si GRASS est un outil puissant, il n'est pas forcément le meilleur ou le seul choix possible pour toutes les tâches. La conception de PyWPS a changé de telle sorte qu'il puisse être utilisé sans GRASS en arrière plan, mais avec n'importe quel autre outil, ou même juste avec Python. PyWPS est une implémentation du standard WPS de l'OGC comme défini dans le document OGC 05-007r4. Actuellement, le support de tout le standard n'est pas complet, mais presque 95 % de la norme est implémentée et utilisable.

Le projet est construit sur un simple script CGI, afin de rendre la vie des développeurs de WebSIG la plus simple possible. Il met à disposition les fonctionnalités pour :

- Analyser toutes les entrées et créer toutes les sorties,
- Réaliser des validations basiques sur les entrées, comme vérifier le type d'entrées LiteralValue ou la taille maximale de fichier pour une entrée de type ComplexValue, etc.
- Créer et supprimer à la volée des fichiers et répertoires temporaires, comme des emplacements et mapsets GRASS, ou d'autres fichiers créés pour pouvoir effectuer le calcul.
- Lancer d'autres opérations utiles.

Le développeur n'a qu'une seule chose à faire : définir les entrées et sorties, ce qui est simplement un script en langage Python. Le process est une classe *Process*, avec une méthode requise nommée *execute*, dans laquelle le calcul est effectué. Les données d'entrée et de sortie sont définies d'une manière similaire, dans une structure de dictionnaire complexe[10] :

```
{
    'Identifier':'maxdist',
    'Title': 'Maximal distance',
    'Abstract':'Maximal distance of visibility',
    'LiteralData': {
        'values':[ [0.,5000] ],
    },
    'dataType': type(0.0),
},
{
```

[7] http://pywps.ominiverdi.org/cgi-bin/wps.py?service=WPS&request=GetCapabilities

[8] http://pywps.ominiverdi.org/cgi-bin/wps.py?service=WPS&request=DescribeProcess&version=0.4.0&identifier=visibility

[9] http://pywps.ominiverdi.org/cgi-bin/wps.py?service=WPS&version=0.4.0&request=Execute&identifier=visibility&datainputs=x,602829.1875,y,4925326.875,maxdist=2000,observer=1.2,dem,http://somewhere/?some&service

[10] Ceci a été remplacé par des appels de méthodes *Add*Input()* dans la version *svn* actuelle

```
    'Identifier': 'dem',
    'Title': 'Digital elevation mode'
    'Abstract': 'Raster map with elevation model',
    'ComplexValueReference': {
        'Formats':["image/tiff"],
    }
},
```

La méthode *execute()* peut utiliser par exemple directement des modules de GRASS :

```
def execute(self):
    # importation de dem
    os.system("r.in.gdal in=%s out=dem" %\
                    (self.datainputs['dem']))
    # reglage du secteur a partir du fichier dem
    file os.system("g.region rast=dem")
    # module de ligne de visee
    os.system("r.los input=dem output=output \
              coordinate=%s,%s max_dist=%f \
              obs_elev=%d" % \
              (self.datainputs['x'],
              self.datainputs['y'],
              self.datainputs['maxdist'],
              self.datainputs['observer']))
    # export du raster de sortie
    os.system("r.out.gdal in=output out=out.tif")
    # reglage de la valeur de retour
    self.dataoutputs['output'] = "out.tif"
    return
```

Comme le code source est la meilleure documentation, environ dix exemples de process sont distribués avec le code source de PyWPS, pour que l'utilisateur puisse avoir une idée générale de la définition d'un process. Il y a aussi une documentation disponible en ligne comme hors ligne, qui tente de décrire le mieux possible le processus d'installation et la mise en place de vos propres process.

Récemment, une nouvelle classe a été définie, qui permet une définition facile des entrées et sorties de process, ainsi qu'un meilleur support pour les modules de GRASS. Toute interface Web sera désormais capable de suivre l'avancement des imports dérivés directement du module `r.in.gdal`. Cette amélioration sera disponible dans la prochaine version de PyWPS.

Avancées du développement

La première version "stable" de PyWPS avec le numéro de version 1.0.0 est sortie en Novembre 2006. Actuellement, l'équipe de développement de PyWPS projette de sortir la version 2.0.0 prochainement, avec de nouvelles fonctionnalités et quelques corrections de bogues. L'effort général de développement est effectué dans trois directions :

- Implémentation du standard OGC WPS complet,
- Sécuriser l'application au maximum afin d'éviter la compromission de serveur,
- Rendre la vie du développeur de process la plus simple possible.

L'équipe de développement de PyWPS voudrait également engager des discussions avec la communauté géospatiale pour définir des métadonnées de process. Le standard OGC WPS définit les types d'entrée d'un point de vue du process, mais n'aborde pas le sujet des types d'entrée (et de sortie) d'un point de vue utilisateur (c'est à dire de l'interface). Par exemple, la coordonnée x est de type *LiteralValue*, mais cela ne définit pas x comme une coordonnée. Il serait cependant utile de définir cette entrée avec un clic de souris sur une carte, plutôt qu'en tapant au clavier dans un champ de texte. Si vous voulez jeter un oeil de plus près à PyWPS, vous pouvez visiter la page du projet[11], où le code source est disponible, ainsi que des liens vers des projets utilisant déjà PyWPS.

Utiliser ka-Map & PyWPS pour créer un WebSIG GRASS

Ominiverdi[12] a joint le développement de PyWPS après la première version. Notre but initial était de créer un client Web libre (FOSS : Free Open Source Software) pour accéder aux fonctions de GRASS. PyWPS n'était pas la première tentative de créer une connexion entre GRASS et le Web, mais aucune autre n'était basée sur des standards ouverts, rendant impossible la création d'une plateforme partagée. Dès le départ nous voulions travailler à deux différents projets : **Embrio** et **Wuiw**.

Embrio

Embrio est la première implémentation que nous avions à l'esprit : utiliser PyWPS pour faire interagir UMN MAPSERVER et l'API MapScript avec GRASS.

Un autre but important était un client web riche, et c'est pourquoi nous avons décidé de baser nos efforts sur ka-Map. Ka-Map utilise Mapscript pour accéder à l'ensemble de puissantes fonctionnalités de

[11] Page du projet PyWPS : http://pywps.wald.intevation.org
[12] Site d'Ominiverdi : http://www.ominiverdi.org

UMN Mapserver et offrir une expérience de navigation à la Google Maps. La sortie de PyWPS, une fois que le process retourne une carte, peut être en GeoTiff, pour une sortie raster, ou en GML pour une sortie vecteur. Les deux formats sont accessibles nativement par UMN MapServer et peuvent être facilement couplés avec d'autres environnements cartographiques. Ainsi, une requête au module *Visibility* peut retourner un GeoTiff, qui est gardé par ka-Map et inséré dans le vrai système de coordonnées de la carte. Ensuite un style, qui peut être un SLD, est appliqué aux valeurs du raster, et un cache temporaire est créé à la volée pendant que les tuiles sont renvoyées au client. Le cache temporaire est lié au *sessionId*.

Ce système permet à différents modules de s'exécuter en parallèle et de comparer leurs sorties en utilisant l'interface de ka-Map. Le contrôle de la transparence et de la position du layer peuvent être utiles pour comprendre le résultat final. La fonctionnalité de requête est également synchronisée si la sortie est requêtable. Le module *Visibility* peut produire l'angle incident avec le point de vue, et la fonction de requête peut retourner la valeur de chaque pixel cliqué.

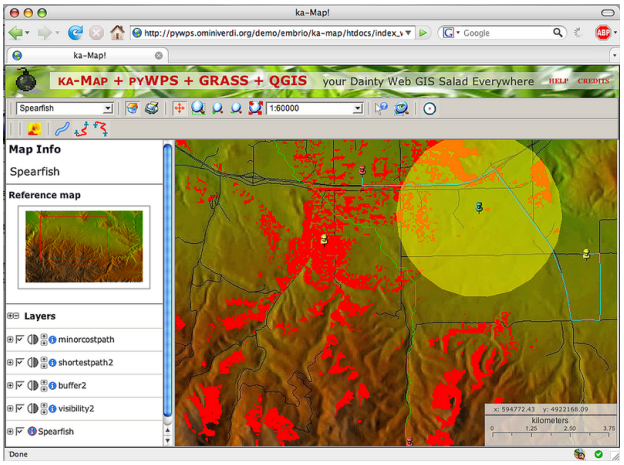

FIG. 1 – Copie d'écran de l'interface Embrio la plus récente.

Wuiw

Wuiw est encore aujourd'hui plus un concept qu'une réelle application. Il a pour objectif de fournir une API Javascript qui connecte le service WPS à des sources de données en OWS, et de faire un rendu de la sortie indépendamment d'une quelconque application de serveur cartographique.

Nous sommes encore en train d'évaluer les limites actuelles du brouillon WPS, qui ne relient pas encore les définitions des entrées avec la définition d'un type complexe. La première idée était d'utiliser des informations en métadonnées pour créer ce type de relation, mais le risque existe de créer un dialecte parallèle qui viendrait amoindrir les avantages de l'intéropérabilité par les standards. Nous espérons que le chemin de WPS vers une version 1.0 apportera une solution adaptée à cette limitation.

Développements futurs

Il est clair que WPS, PyWPS, Embrio et Wuiw ont tous une histoire courte. Même si la plupart des choses sont encore à faire, le début est prometteur. Concernant Embrio, nous prévoyons d'améliorer l'interaction avec les retours des scripts de process WPS afin d'afficher une barre de progression. Beaucoup d'autres modules de GRASS peuvent être développés, et nous espérons recevoir de l'aide de la communauté GRASS pour les réaliser.

Un autre but objectif important est d'ajouter plus d'interaction avec les applications en ligne de commande (CLI : Command Line Interface), comme R, GDAL/OGR, etc. pour réaliser des géostatistiques, des conversions de format, et beaucoup, beaucoup d'autres fonctionnalités.

Nous espérons aussi ajouter une ligne de commande en AJAX pour intéragir avec un système protégé utilisant WPS.

Licenses

Il est important de noter que ce projet utilise beaucoup de logiciels avec au moins deux licences différentes :

- GNU/GPL : GRASS, PyWPS, R
- MIT/BSD : UMN MapServer, MapScript, ka-Map, Embrio

Références

- OGC Web Processing Service [13]

[13] http://www.opengeospatial.org/standards/requests/28
[14] PyWPS Home page : http://pywps.wald.intevation.org
[15] Démonstration : http://pywps.ominiverdi.org/demo/embrio/ka-map/htdocs/index_wps_qgis.html

- Site web de PyWPS : [14]
- Dernière démonstration d'embrio : [15]
- Site web d'Embrio[16]

Jáchym Čepický
http://les-ejk.cz
jachym AT les-ejk cz

Lorenzo Becchi
http://ominiverdi.org
lorenzo AT ominiverdi com

[16] http://pywps.ominiverdi.org/

FOSS4G - Ouverture des Inscriptions à la Conférence

Nous sommes heureux de vous annoncer l'ouverture des inscriptions en ligne à la Conférence Free and Open Source Software for Geospatial 2007 (FOSS4G 2007). FOSS4G est l'évènement annuel qui réunit les personnes et les sociétés qui créent, utilisent, et gèrent des logiciels géospatiaux open source. Inscrivez-vous dès maintenant en ligne.[17]

Inscrivez-vous avant la date limite du 27 Juillet, pour économiser sur les frais d'inscription ! Tirez profit de l'opportunité que FOSS4G 2007 vous offre, de construire un réseau avec les autres professionnels des données géospatiales, de renouveler d'anciennes relations, et d'en créer de nouvelles.

Pour les dernières mises à jour, l'inscription et/ou la soumission d'une présentation, visitez le site web de la conférence.[18]

OPPORTUNITES D'EXPOSITION & DE SPONSORING

Concernant les opportunités d'exposition et de sponsoring, lisez la page des partenaires [19] ou contactez Paul Ramsey, Président de la Conférence par email.[20]

SOUMETTRE UNE PRESENTATION

Vous pouvez soumettre une présentation en ligne.[21] La date limite pour les soumissions est le 29 Juin 2007.

Les présentations FOSS4G durent 25 minutes, avec 5 minutes de questions/réponses à la fin. Les présentations concernent l'utilisation ou le développement de logiciels géospatiaux opensource. Tout le monde peut soumettre une proposition de présentation et participer à la conférence comme présentateur. Plus d'informations sont disponibles sur la page des présentations sur le site web.

Nous espérons vous voir à Victoria, au Canada en Septembre !

[17] Inscription en ligne : http://www.foss4g2007.org/register/
[18] Site web de la conférence : http://www.foss4g2007.org/
[19] Page des partenaires : http://foss4g2007.org/sponsors
[20] Email Paul Ramsey : pramsey@foss4g2007.org
[21] Soumettez une présentation sur http://www.foss4g2007.org/presentations/

Le *journal de l'OSGeo* est une publication de la *Fondation OSGeo*. La base de ce journal, les sources du style LaTeX 2ε ont été généreusement fournit par l'équipe éditoriale de l'actualité de GRASS et R.

Journal en ligne : http://www.osgeo.org/journal

Site de l'OSGeo : http://www.osgeo.org

Contact postal pour l'OSGeo, PO Box 4844, Williams Lake, British Columbia, Canada, V2G 2V8

ISSN 1994-1897