Mapfeatureserver как замена ArcGIS Server

Обсудить в форуме Комментариев — 25

Эта страница опубликована в основном списке статей сайта
по адресу http://gis-lab.info/qa/mapfeatureserver.html

Описание веб-сервиса Mapfeatureserver как замены ArcGIS Server

Mapfeatureserver (далее MFS) — это веб-сервис, написанный на Python (WSGI, Flask) и реализующий ArcGIS Server REST API для слоев типа Feature Layer. MFS был задуман как средство, позволяющее избавиться от дорогостоящего ArcGIS Server при работе с веб-картами, использующими ArcGIS API, равно как и получить дополнительные возможности, отсутствующие в ArcGIS Server.

Mapfeatureserver будет полезен разработчикам ГИС решений для веб и интранет, поскольку позволяет получить веб-карты красивые и функциональные как в ArcGIS, но без затрат на приобретение ArcGIS Server. Как продукт с открытым исходным кодом MFS может принести дополнительную пользу Python-разработчикам.

Содержание

Введение в тему

Чтобы было понятнее, что такое MFS, нужно иметь представление о стеке технологий, поверх которых он работает.

Как известно, есть такой подкласс ГИС-решений, как клиент-серверные ГИС. В случае, когда клиент использует для работы с картами веб-браузер или мобильный терминал вроде смартфона или планшета, применяется название «веб-ГИС». Вероятно потому, что для общения клиент-сервер используются веб-технологии. Основные отличающие черты таких решений заключаются в применении некоего картографического сервера и использовании HTTP/HTTPS для передачи данных между клиентом и сервером. Самый распространенный вариант — это отрисовка клиентом растровых картинок, присылаемых с сервера по запросу. Из этих растровых картинок (тайлов), как из мозаики, складывается изображение карты.

На текущий момент известно не так много веб-ГИС решений, которые позволяют не только получать растровые тайлы, но и дают клиенту возможность полноценно работать со слоями данных, используя геометрию объектов (features) в слоях карты, передавая между клиентом и сервером координаты точек, составляющих фигуры, их атрибуты и сопутствующие метаданные. Естественно, передача таких данных, назовем их условно «геометрия», осуществляется согласно определенным протоколам. Имеются в виду форматы передачи данных поверх HTTP/HTTPS. В мире свободного ПО самое большое распространение получили протоколы WMS/WFS, а если точнее, согласно теме данной статьи, WFS (Web Feature Service).

У компании Esri есть свое решение. Продукт под названием ArcGIS for Server содержит подсистему, которая публикует веб-карты, слои данных, из которых сделаны эти карты, и много чего еще, в данном случае не существенного. В процессе публикации карты или слоя, на сервере АркГИС создаются так называемые «веб-сервисы». Нас интересуют опубликованные слои, из которых собираются карты. В терминологии ArcGIS такие слои называются Feature Layer а сервисы, создаваемые АркГИС-ом для них называются Feature Service. Протокол доступа к таким слоям носит название ArcGIS Server REST API и позволяет через HTTP/HTTPS получать геометрию и атрибуты объектов слоя. Сами передаваемые данные упаковываются в текст формата JSON.

Примеры данных будут приведены ниже. Если при создании Feature Service выполнить некоторые условия (например, разместить данные в специальной БД), то при использовании такого сервиса можно будет использовать все операции спектра CRUD, а не только чтение.

Следует упомянуть, что ArcGIS Server умеет создавать разные типы сервисов, не только Feature Sevice. К примеру, вполне можно опубликовать слой геоданных, доступ к которому будет открыт по протоколу WFS. Но в данной статье нас интересует только ArcGIS Server REST API, подраздел Feature Service.

Предпринимаются попытки создать на основе ArcGIS Server REST API всеобщий стандарт на геосервисы. Этот стандарт именуется GeoServices REST API (http://www.opengeospatial.org/standards/requests/89). На данный момент opensource-сообщество очень неоднозначно реагирует на его продвижение. Насколько известно автору, на момент написания статьи не существует ни одной реализации GeoServices REST API.

Существуют и друге протоколы работы с геоданными, несущие модную аббревиатуру REST в своем названии или описании, например MapFish Protocol или GeoServer REST API. Эти решения тоже заслуживают внимания, но не имеют никакого отношения к Mapfeatureserver.

Что касается Mapfeatureserver, то это продукт того же семейства, что Papyrus или FeatureServer, только нацелен на реализацию протокола ArcGIS Server REST API, в отличие от упомянутых.

К созданию MFS автора подтолкнули следующие соображения и обстоятельства. Нашим клиентам очень нравится клиентская часть для веб-ГИС, называемая ArcGIS Viewer for Silverlight, на базе которой мы создали специальную сборку «Картобонус». Разумеется, как детище Esri, такой вьювер лучше всего работает с серверной частью этого же вендора и не очень способствует попыткам подружить его с слоями, загружаемыми через сторонние сервисы, в частности WFS. Это был первый «звоночек» — решение задачи «подключение к Картобонусу слоев из WFS». Можно либо во вьювер добавить код, транслирующий примитивы WFS в обьекты вьювера (как-то так), либо написать некий сервис-прокси для превращения WFS в ArcGIS Feature Service, понимаемый вьювером «из коробки».

Другим пожеланием клиентов была функция «загрузки шейпов» во вьювер. Довольно трудно обьяснить неспециалисту, что парсинг и отрисовка данных из шейпа в браузере — весьма не тривиальная задача. Да и зачем? Ведь можно дать пользователю кнопку «загрузка шейпа», нажав которую, пользователь сможет создать на некоем сервере таблицу в БД и записать в нее копию шейп-файла. После чего подключить к карте во вьювере слой, сформированный из этой таблицы. Второй «звоночек» — нужен сервис, в который мы сможем грузить шейпы и потом выводить их клиентам веб-ГИС.

Кстати, о клиентских вьюверах. Очевидно, лучше всех с геоданными из Feature Service, эмуляцией которого и занимается Mapfeatureserver, умеют работать программы от Esri (упоминать приложения типа ArcMap я тут не буду, только веб).

- ArcGIS Online — яркий пример картографического вьювера, сделанного на JavaScript и продвигаемого как SaaS.
- ArcGIS Viewer for Flex — кроссплатформенное приложение типа RIA на технологии Flash/Flex, расширяемое, с открытым кодом.
- Уже упомянутый ArcGIS Viewer for Silverlight — тоже RIA, но работает только под MS Windows. Зато, используя MS Visual Studio, процесс разработки плагинов и расширений заметно ускоряется и упрощается.

Заметим, что эти вьюверы активно эксплуатируют очень приличный набор API, предоставляемый Esri безвозмездно, то есть даром.

И, конечно, всеми любимый OpenLayers тоже умеет работать со слоями по протоколу ArcGIS Server REST API.

Mapfeatureserver - что это такое?

Как я уже упомянул, MFS — это программа с открытым исходным кодом на Python, которая после запуска создает веб-сервис, отвечающий спецификации ArcGIS Server REST API для картографических слоев типа Feature Layer. Веб модуль MFS написан с использованием фреймворка Flask и отвечает спецификации WSGI,

что позволяет использовать MFS в качестве части более крупных веб-решений.

Геоданные MFS считывает из PostGIS DB, что означает необходимость: а) загрузить данные предполагаемого решения в БД PostGIS; б) обеспечить доступ к этой БД сервису MFS. На текущий момент, кроме PostGIS, другие БД не поддерживаются, но есть планы добавить поддержку MySQL и MongoDB. Также, есть планы по использованию служб WFS в качестве источников данных.

Общая картина использования Mapfeatureserver выглядит примерно так.

- Геоданные (шейп-файлы, к примеру) загружаем в PostGIS.
- Для каждого слоя данных вписываем сведения в конфигурационные файлы MFS.
- Запускаем веб-сервис.
- В клиентской программе, к примеру Картобонус, добавляем к карте слои точно так же, как обычные FeatureLayer из ArcGIS Server.

Теперь о недостатках и ограничениях MFS.

На текущий момент программа находится в стадии «Proof of Concept», то есть обладает функциональностью, минимально достаточной для демонстрации работоспособности подхода. Из всего многообразия запросов, декларированных в API, наш сервис пока реализует два:

- layer metadata

```
http://<featureservice-url>/<layerId>
```

- layer data query by box

```
http://<featurelayer-url>/query
```

причем запрос данных может быть только одного типа — запрос на выборку по ограничивающему боксу (box). Этого достаточно, чтобы загрузить слой в карту и делать zoom, pan, просмотр атрибутов для features, но и только.

Пример ответа сервера на запрос метаданных слоя


```
"currentVersion": 10.11,
 "id": 0,
 "name": "Ямочный ремонт",
 "type": "Feature Layer",
 "description": "Места установки заплаток на дорогах",
...
 "supportsAdvancedQueries": true,
 "geometryType": "esriGeometryPoint",
 "minScale": 0,
 "maxScale": 0,
 "extent": {
  "xmin": 27.765770083999996,
  "ymin": 52.86936644999997,
  "xmax": 36.71166105100002,
  "ymax": 62.012733638999975,
  "spatialReference": {
   "wkid": 4326,
   "latestWkid": 4326
  }
 },
 "drawingInfo": {
  "renderer": {
   "type": "uniqueValue",
   "field1": "roadcarpet",
   "field2": null,
   "field3": null,
   "fieldDelimiter": ", ",
   "defaultSymbol": null,
   "defaultLabel": null,
   "uniqueValueInfos":
        {
```

```
          "symbol": {
           "type": "esriPMS",
           "url": "http://localhost:5000/static/asfalt.png",
           "imageData": "",
           "contentType": "image/png",
           "width": 24,
           "height": 24,
           "angle": 0,
           "xoffset": 0,
           "yoffset": 0
          },
          "value": "Асфальт",
          "label": "Асфальт",
          "description": ""
         },
         {
          "symbol": {
           "type": "esriPMS",
...
 },
 "hasM": false,
 "hasZ": false,
 "allowGeometryUpdates": true,
...
 "fields":
  {
   "name": "objectid",
   "type": "esriFieldTypeOID",
   "alias": "OBJECTID",
   "domain": null,
   "editable": false,
   "nullable": false
  },
  {
   "name": "ptchlenght",
   "type": "esriFieldTypeSmallInteger",
   "alias": "ДлинаУчастка, м",
   "domain": null,
   "editable": true,
   "nullable": true
  },
  {
   "name": "pthcdeptht",
   "type": "esriFieldTypeSmallInteger",
   "alias": "ГлубинаУчастка, см",
   "domain": null,
   "editable": true,
   "nullable": true
  },
...
 "maxRecordCount": 1000,
 "supportedQueryFormats": "JSON, AMF",
 "capabilities": "Create,Delete,Query,Update,Uploads,Editing"
}
```


Пример ответа сервера на запрос данных слоя


```
{
  "features":
    {
      "attributes": {
        "descr": "werwre",
        "gid": 8,
        "ptchlenght": 500,
        "pthcdeptht": 8,
```

```
        "regdaterec": "2012/08/02",
        "regdaterep": "2012/09/15",
        "roadcarpet": "Асфальт"
      },
      "geometry": {
        "x": 3980475.9450405277,
        "y": 6976079.279805333
      }
    }
  ,
  "fields":
    {
      "alias": "OBJECTID",
      "name": "gid",
      "type": "esriFieldTypeOID"
    },...
    {
      "alias": "REGDATEREP",
      "length": 50,
      "name": "regdaterep",
      "type": "esriFieldTypeString"
    },
    {
      "alias": "ROADCARPET",
      "length": 50,
      "name": "roadcarpet",
      "type": "esriFieldTypeString"
    }
  ,
  "geometryType": "esriGeometryPoint",
  "globalIdFieldName": "",
  "objectIdFieldName": "gid",
  "spatialReference": {
    "latestWkid": 3857,
    "wkid": 102100
  }
}
```


Остальная часть API будет реализована несколько позже. Хорошая новость заключается в том, что проект - open source и любой, кто обладает соответствующими навыками, может ускорить реализацию недостающих функций.

Инструкция по использованию Mapfeatureserver

Изложенная здесь информация может устареть к тому времени, как вы читаете этот текст. Наиболее свежую информацию о проекте вы всегда можете найти на странице проекта в GitHub https://github.com/vasnake/mapfeatureserver

Чтобы запустить сервис MFS, вам понадобится выполнить следующие шаги (в планах есть пункт: сделать нормальный Python distribution):

- Скачать MFS с GitHub.
- По настройки в файле
  `mapfeatureserver\wsgi\default_settings.py`
- Установить Python 2.7 и необходимые библиотки, к примеру, для MS Windows

```
set path=%path%;c:\d\Python27;c:\d\Python27\Scripts
pip install Flask flask-login blinker psycopg2 simplejson
```

psycopg2 for Windows

- Запустить приложение Flask

```
pushd mapfeatureserver\wsgi
python mapfs_controller.py
```

URL веб-службы будет таким http://localhost:5000/ Если вы откроете эту страницу в браузере, вы увидите служебную страницу со ссылками на тестовые слои. Эти ссылки работать не будут, так как у вас нет таких слоев. Удалить лишнее и добавить свое вы можете, поправив файл

```
mapfeatureserver\wsgi\templates\servlets.html
```

Чтобы создать новый слой, вам нужно выполнить следующие шаги:

- Получить доступ к БД PostGIS, к примеру, установив БД на свой хост.
- Загрузить шейп-файл с нужными данными в БД, к примеру так (описания опций и ключей к командам доступны во встроенной в команды справке):

```
set path=%path%;c:\Program Files\PostgreSQL\9.0\bin
pushd c:\t\shpdir
shp2pgsql.exe -d -I -s 4326 -W cp1251 flyzone.shp mfsdata.flyzone > flyzone.dump.sql
psql -f flyzone.dump.sql postgisdb mfs
```

Здесь shpdir это папка, где расположены шейп-файлы; flyzone.shp - загружаемый шейп-файл; mfsdata.flyzone - схема данных и таблица, в которую будет загружен шейп; postgisdb - имя БД; mfs - учетная запись в БД.

- Записать сведения о слое в конфигурационный файл

```
mapfeatureserver\config\layers.config.ini
```

Идентификатор слоя (любое положительное целое число) надо вписать в список (в примере указаны три слоя)

```
layer.ID.list: 0,1,2
```

Вписать данные подключения к БД PostGIS, пример:

```
PG.DSN: host=vags101 port=5432 dbname=postgisdb user=mfs password=12345678
connect_timeout=10 client_encoding=utf8
```

Создать секцию, именованную по идентификатору слоя, скажем, «2»

```
2
layer.table = flyzone
layer.geomfield = geom
layer.oidfield = gid
layer.name = Зоны полетов с ограничениями
```

Находящиеся в конфиге примеры и комментарии помогут не ошибиться.

- Создать файл метаданных для слоя. Это самая трудная часть.

```
mapfeatureserver\config\layer.<layer id>.config.json
```

Чтобы было легче, можно скопировать метаданные из аналогичного существующего слоя Feature Layer ArcGIS и внести в него правки.

Метаданные слоя из ArcGIS Server доступны по URL типа

```
http://testags/arcgis/rest/services/flyzone/FeatureServer/2?f=pjson
```

если на сервере опубликована служба «flyzone» соответствующего типа.

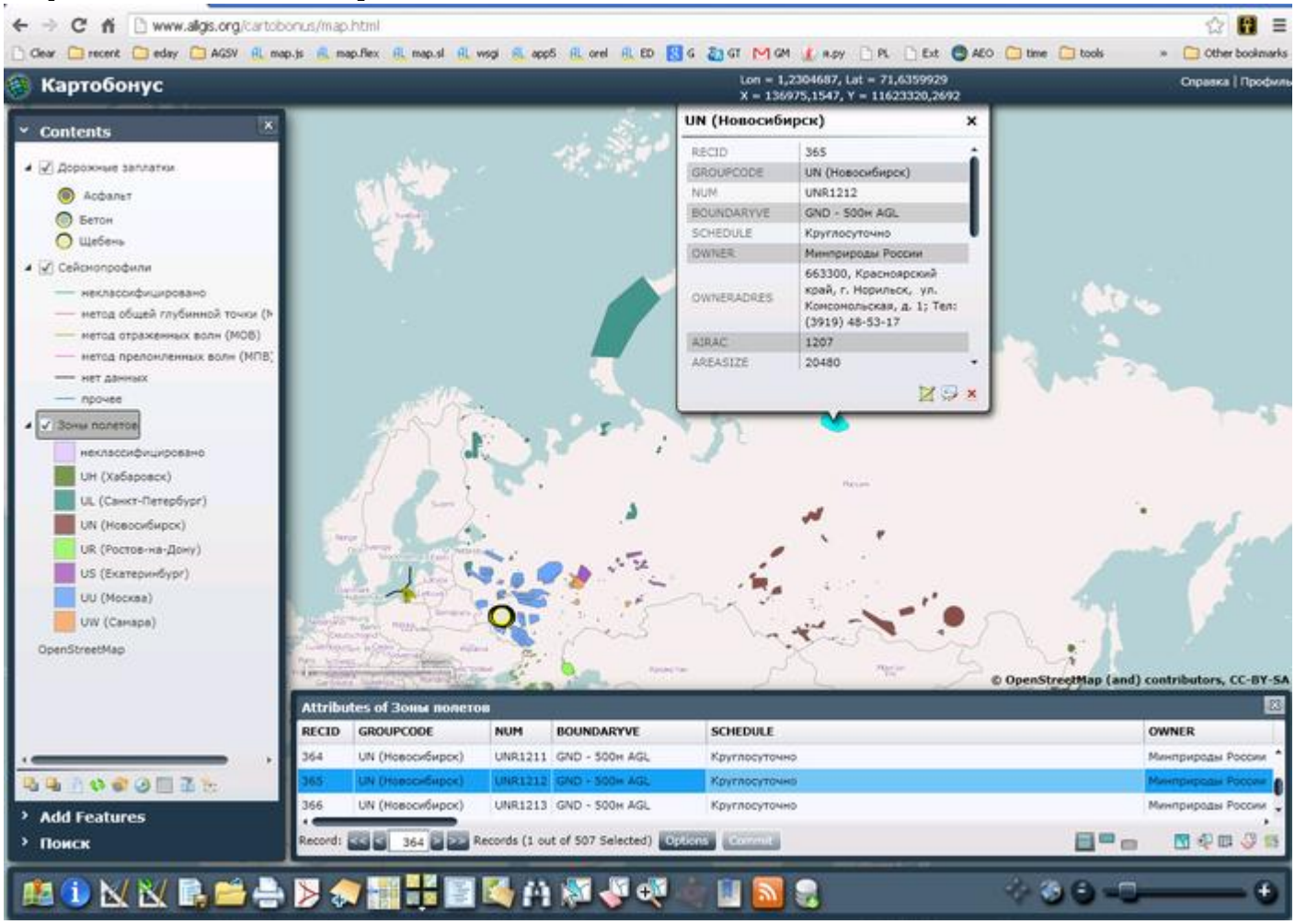
Также, в MFS есть специальные страницы типа

```
http://localhost:5000/admin/dsn/flyzone?oidfield=gid&geomfield=geom
```

для помощи в составлении файла метаданных.

После выполнения этих шагов можно использовать слои MFS как обычные слои ArcGIS FeatureLayer в веб-картах, построенных на ArcGIS web API. К примеру, есть вьювер Картобонус, построенный на ArcGIS API for Silverlight, именно он использовался для тестирования MFS. Чтобы добавить слой в карту, используйте URL вида

`http://hostname:5000/<layer id>`



Как видите, за исключением файла метаданных, всё достаточно просто. Теперь у нас есть свободный и бесплатный сервер Feature Layer-ов для обеспечения работы любого картографического софта, использующего спецификации ArcGIS REST API.

Оно не работает!

Поскольку, как уже говорилось, проект находится в стадии разработки, вполне вероятно, что вы не сможете заставить его работать с вашими данными. Безошибочным признаком того, что MFS не работает как должно, считается наличие Python traceback в окне с приложением Flask (модуль mapfs_controller.py). К примеру, это может выглядеть так:

```
Traceback (most recent call last):
  File "mapfs_controller.py", line 166, in layerOperations
    resp = layerdata.layerData(lyrconf, ds, op)
  File "C:\d\code\git\mapfeatureserver\wsgi\layerdata.py", line 65, in layerData
    res = layerDataInBox(datasource.cursor, lyrconf, outSR, inpBox)
  File "C:\d\code\git\mapfeatureserver\wsgi\layerdata.py", line 109, in layerDataInBox
    box = esri.AGGeometryBox(inpBox)
  File "C:\d\code\git\mapfeatureserver\wsgi\esri.py", line 47, in __init__
    box = simplejson.loads(jsontext)
  File "c:\d\Python27\lib\site-packages\simplejson\__init__.py", line 461, in loads
    return _default_decoder.decode(s)
  File "c:\d\Python27\lib\site-packages\simplejson\decoder.py", line 374, in decode
    obj, end = self.raw_decode(s)
  File "c:\d\Python27\lib\site-packages\simplejson\decoder.py", line 393, in raw_decode
    return self.scan_once(s, idx=_w(s, idx).end())
  File "c:\d\Python27\lib\site-packages\simplejson\scanner.py", line 119, in scan_once
    return _scan_once(string, idx)
  File "c:\d\Python27\lib\site-packages\simplejson\scanner.py", line 84, in _scan_once
    raise JSONDecodeError(errmsg, string, idx)
```

```
JSONDecodeError: Expecting value: line 1 column 1 (char 0)
```

Если вы столкнетесь с чем-то подобным, не пожалейте времени, скопируйте текст traceback-а и отправьте автору по почте. Или, еще лучше, откройте issue на GitHub.

Дополнительные сведения по настройке MFS.

- Минимальный набор привилегий пользователя в PostgreSQL, пример:

```
CREATE USER guest WITH password 'guest';
ALTER USER guest SET search_path TO mfsdata,public;
GRANT USAGE ON schema mfsdata TO guest;
GRANT SELECT ON table mfsdata.patching TO guest;
GRANT SELECT ON geometry_columns TO guest;
GRANT SELECT ON geography_columns TO guest;
GRANT SELECT ON spatial_ref_sys TO guest;
```

Подразумевается, что пользователь 'guest', от имени которого должен работать Mapfeatureserver, должен иметь доступ на чтение к слою, хранимому в таблице 'mfsdata.patching'.

Ссылки

- Спецификации REST API или, более подробно, здесь
- Статья в блоге автора
- MFS на GitHub
- web map viewer Cartobonus

E-mail: vasnake AT gmail DOT com

Обсудить в форуме Комментариев — 25

Последнее обновление: 2014-05-15 01:47

Дата создания: 23.05.2013
Автор(ы): Валентин Федулов