ZOO Project

オープンソース WEB プロセシングサービスエンジンの展望

社会事業部　企画室
林　　博文
帝塚山学院大学　リベラルアーツ学部
吉田　大介

1. はじめに

地理空間情報システム（GIS）の業界では近年、2005 年の Google のマップサービスの開始を機に地理空間情報の自由化と相互利用が始まり、現在日本では、国土交通省や国土地理院をはじめとする多くの行政機関や団体で GIS データが公開され、自由に利用ができるようになった。国際的にも、「Gov2.0」というキーワードと共に各国が公共データの民間活用に積極的に取組んでいる。

GIS データが自由に活用できることは良いことであるが、一般の人々にとって、GIS の生データをそのまま活用するというのは技術的にもコスト的にも敷居が高く、もっと簡単にデータが扱え、自分たちの目的の実現のために利用したいと考えている。これを実現するための基盤技術が「ウェブプロセシングサービス（Web Processing Service：WPS）」と呼ばれるものである。最近話題のクラウドコンピューティングの基盤的な技術である WPS は、現在 GIS 業界においてもっとも注目されている技術で、様々な分野で応用が可能である。

1.1　WPS と ZOO

WPS は利用者が必要とするデータや処理をサーバー側に URL の引数として要求して、XML で結果を返答するサービスである。例えば Google API のように、処理の詳細を知らなくても、「住所」を入力して「緯度経度」を得ることがとても簡単にできるようになる。WPS を介して GIS データやその他のサービスを利用することで、データ交換や処理サービスの共通化を図ることができ、巨大なデータを取り扱う上でのコスト削減、さらに利用者のアプリケーションが様々なデータを利用することが可能になる。

WPS はネットワークを介して GIS データ等を交換する手順やデータの共通仕様に基づいて、機能仕様が決定されている[1]。標準を定めているのは Open Geospatial Consortium（OGC [2]）で、規格を通じた地理空間情報の共有に取り組む国際標準化団体である。OGC 規格に準拠した WPS エンジンをオープンソースで開発しているプロジェクトは世界に4つあり、「52°　North　WPS」、「deegree WPS」、「pyWPS」、そして「ZOO」である。

WPS はサーバー側でプログラム言語を使用してサービスを記述することで、クライアントからのリクエストにより GIS データの解析を行い、処理結果を利用者に返す機能を実現している。プログラム言語はプロダクトにより様々で、中でも最後発の「ZOO」は、多言語に対応したインタフェース（表1）と簡単な記述によるサービスの実装と、チェイン機能（サービスとサービスの連携）を実現している。

表 1 対応言語

（◎は ZOO Kernel ネイティブサポート、○はオプション調整が必要）

言語名	サポート	コンパイラ/インタプリタ
C / C++	◎	GCC4
Python	○	Python interpreter
Fortran	○	g90
PHP	○	PHP embedded
Java	○	Java SDK
Javascript	○	SpiderMonkey
Perl	○	Perl interpreter

「ZOO」という名称の由来は、この多言語対応によって、オープンソースの利用者や開発者が既に所有している技術（プログラム）を、ウェブサービス化するというアイディアが、オープンソースの各プロダクトが持つ、象やペンギン、牛などのマスコットキャラクタを集めた動物園（ZOO）に見立てていることから来ている。

ZOO プロジェクトは FOSS4G 2008 カンファレンスの間に、Gerald FENOY 氏、 Nicolas BOZON 博士、 Venkatesh RAGAHAVAN 教授によって構想され、大阪市立大学、帝塚山学院大学、応用技術株式会社等幾つかのパートナーが積極的なプロダクトの支援を行っている。

ZOO は「ZOO プロジェクト[3]」コミュニティにより開発サポートが行われており、MIT/X11 ライセンスで提供され、ソースコードとバイナリは商用として自由に使用することが出来る。オープンソース地理空間情報の国際団体、OSGeo 財団が主催する「FOSS4G 2010 カンファレンス」で配布された「OSGeoLiveDVD[4]」に収録されており、汎用 PC を使用して DVD や USB から起動することで、ZOO を簡単に学習することが出来る。

2. ZOO の仕組み

ZOO はサーバー側でユーザーからのリクエストを受け、タスク管理を行う「ZOO Kernel」、処理サービスを記載するメタデータファイル（.zcfg）によりウェブサービス化されたサービスである「ZOO Service」、サービスチェインを実現するスクリプト「ZOO API」から構成される（図 1）。

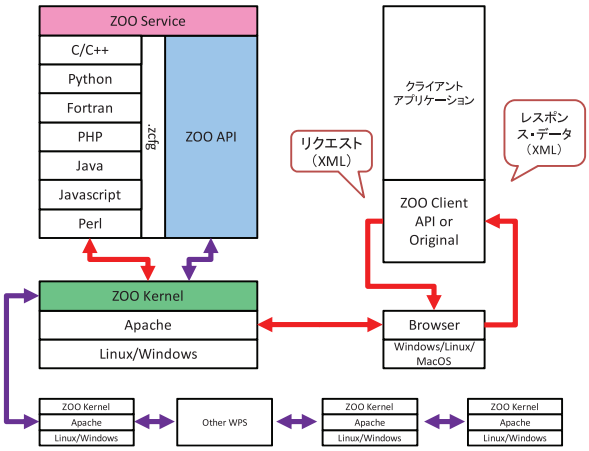

図 1 ZOO のシステム構成

2. 1 ZOO Kernel

ZOO Kernel について開発者がコードを追加する必要はない。サポート言語を追加する場合は必要となるが、コミュニティが開発者のサポートを行っている。[3][6][7]

ZOO Kernel は Apache ウェブサーバーから呼び出された際に、受け取ったリクエストを解析し、必要な処理を呼び出すタスクマネジメントの役割を実行する。ユーザーが送信するリクエストは次のような URL である。

```
http://localhost/cgi-bin/zoo_loader.cgi?
Request=GetCapabilities&Service=WPS
```

GetCapabilities リクエストでは、WPS の実行可能なサービスを下記のように返答する。

```
−<wps:Capabilities xsi:schemaLocation="http://www.opengis.net/wps,
 /wps/1.0.0/wpsGetCapabilities_response.xsd" service="WPS" version="
 −<ows:ServiceIdentification>
    <ows:Title>The Zoo WPS Server</ows:Title>
    <ows:Abstract>FOSS4G 2010 - WorkShop ZooWPS.</ows:Abstrac
    <ows:Fees>None</ows:Fees>
    <ows:AccessConstraints>none</ows:AccessConstraints>
   −<ows:Keywords>
      <ows:Keyword>WPS</ows:Keyword>
      <ows:Keyword>GIS</ows:Keyword>
      <ows:Keyword>buffer</ows:Keyword>
    </ows:Keywords>
    <ows:ServiceType>WPS</ows:ServiceType>
    <ows:ServiceTypeVersion>1.0.0</ows:ServiceTypeVersion>
  </ows:ServiceIdentification>
```

2. 2 サービスの実装

サービスは ZOO Kernel がプログラムの必要な機能を呼び出すことで生成される。必要な機能については、「.zcfg」の拡張子を持つ ZOO メタファイルの記述により、サーバー側にサービスとして実装される。「.zcfg」形式での記載例は次の通りである。

```
[Boundary]
    Title = Compute boundary.
    Abstract = A new geometry object is created and returned
containing the boundary of the geometry on which the method is
invoked.
    processVersion = 1
    storeSupported = true
    statusSupported = true
    serviceProvider = ogr_service.zo
    serviceType = C
    <MetaData>
      title = Demo
    </MetaData>
    <DataInputs>
     [InputPolygon]
     Title = Polygon to compute boundary
     Abstract = URI to a set of GML that describes the polygon.
      minOccurs = 1
      maxOccurs = 1
      <MetaData lang="en">
       title = Mon test
      </MetaData>
      <ComplexData>
       <Default>
        mimeType = text/xml
        encoding = UTF-8
        schema = http://fooa/gml/3.1.0/polygon.xsd
       </Default>
       <Supported>
        mimeType = text/xml
        encoding = base64
        schema = http://fooa/gml/3.1.0/polygon.xsd
       </Supported>
      </ComplexData>
    </DataInputs>
    <DataOutputs>
     [Result]
      Title = The geometry created
      Abstract = The geometry containing the boundary of the
geometry on which the method is invoked.
      <MetaData lang="en">
       title = Mon test
      </MetaData>
      <ComplexData>
       <Default>
        mimeType = application/json
        encoding = UTF-8
        extension = js
        asReference = true
       </Default>
       <Supported>
        mimeType = text/xml
        encoding = UTF-8
        schema = http://fooa/gml/3.1.0/polygon.xsd
        extension = xml
       </Supported>
       <Supported>
        mimeType = text/xml
        encoding = base64
        schema = http://fooa/gml/3.1.0/polygon.xsd
        extension = xml
       </Supported>
      </ComplexData>
  </DataOutputs>
```