Introducing deegree 3 WPS

FOSS4G 2010 Barcelona

Markus Schneider
schneider@lat-lon.de
http://www.lat-lon.de/

Markus Schneider

Involvement with deegree

- Core developer since 2001
- TMC member since 2008

Involvement with lat/lon

- Developer since 2001

Involvement with OSGeo

- Incubation Committee Member

Outline

- What is deegree?
- Web Processing Service 1.0.0
- Features of deegree 3 WPS
- Writing Java-based processes
- What's next?

What is deegree?

- Provides geospatial components (Java)
- LGPL
- Started in 2000 as joint project between:
  - University of Bonn
  - lat-lon GmbH (OGC principal member since 2010)
- Since 2010: Official OSGeo project

```
  _
 _| |                                                    3
/  . |/  ._>/  ._>/  .  || '_>/  ._>/  ._>
\___|\___.\___.\_.  ||_|  \___.\___.
                <___'
```

- Almost 100% rewritten from scratch
- Based on up-to-date base technologies
- Modular, pluggable design
- Major goals
  - Easier configuration
  - Better performance, high scalability

Upcoming deegree 3 releases

Release	Codename	Scheduled Date
3.0	Celsius	November 2010*
3.1	Fahrenheit	May 2011
3.1 + 1	?	November 2011
3.1 + 2	?	May 2012
...	...	...

* targeted for deegree day 2010: 16th - 17th November, Bonn, Germany

Web Processing Service 1.0.0 - Basics

- Official OGC specification
- Goal: Provide geospatial processes via the web
- Based on HTTP and XML
- Operations
  - GetCapabilities: Get service / process info
  - DescribeProcess: Get full process description
  - Execute: Execute process

WPS 1.0.0 – Process Description

- Identifier
- Title (optional)
- Abstract (optional)
- Inputs [0...n]
- Outputs [1...m]


```
−<wps:ProcessDescriptions service="WPS" versi
 /wps/1.0.0/wpsDescribeProcess_response.xsd">
 −<ProcessDescription wps:processVersion="1
   <ows:Identifier>Buffer</ows:Identifier>
   −<ows:Title>
     Process for creating a buffer around a GML g
   </ows:Title>
   −<ows:Abstract>
     The purpose of this process is to create a buf
   </ows:Abstract>
   −<DataInputs>
    −<Input minOccurs="1" maxOccurs="1">
      <ows:Identifier>GMLInput</ows:Identif
      <ows:Title>GMLInput</ows:Title>
     −<ComplexData>
      −<Default>
       −<Format>
         <MimeType>text/xml</MimeType>
         <Encoding>UTF-8</Encoding>
        −<Schema>
          http://schemas.opengis.net/gml/3.1.
```

WPS 1.0.0 – Process parameters

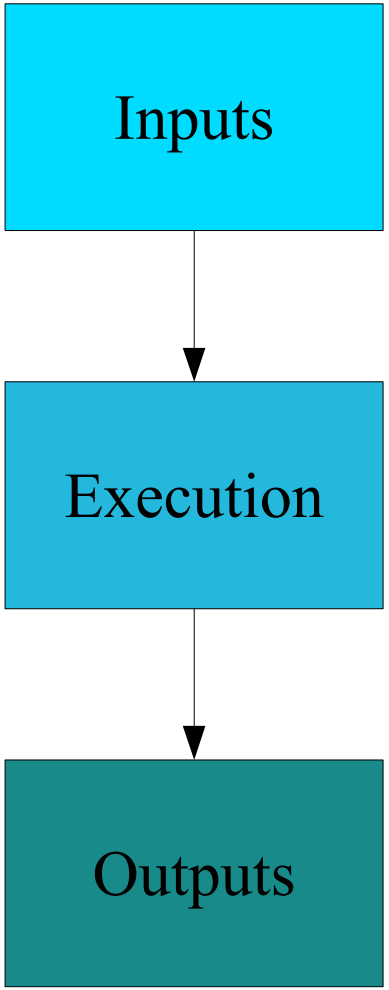
- WPS parameter types
  - Literal: String (with optional type, e.g. double)
  - BoundingBox: Envelope
  - Complex: XML or binary data

WPS 1.0.0 – Complex data type

- Complex is relevant for geospatial data
  - GML geometries / features / feature collections
  - Raster data
  - …
- Defined by three attributes:
  - encoding, mimeType, schema
- Criticism: Very generic, but not well-defined

WPS 1.0.0 – Execute Details

- Request bindings
  - KVP, XML and SOAP
- Parameter passing
  - Inputs inline in request / by reference
  - Outputs inline in response / by ref.
  - Selecting outputs / format



WPS 1.0.0 – Execute Details

- Response variants
  - RawDataOutput
  - ResponseDocument
- Asynchronous execution
  - Storing of response document
  - Polling of execution status

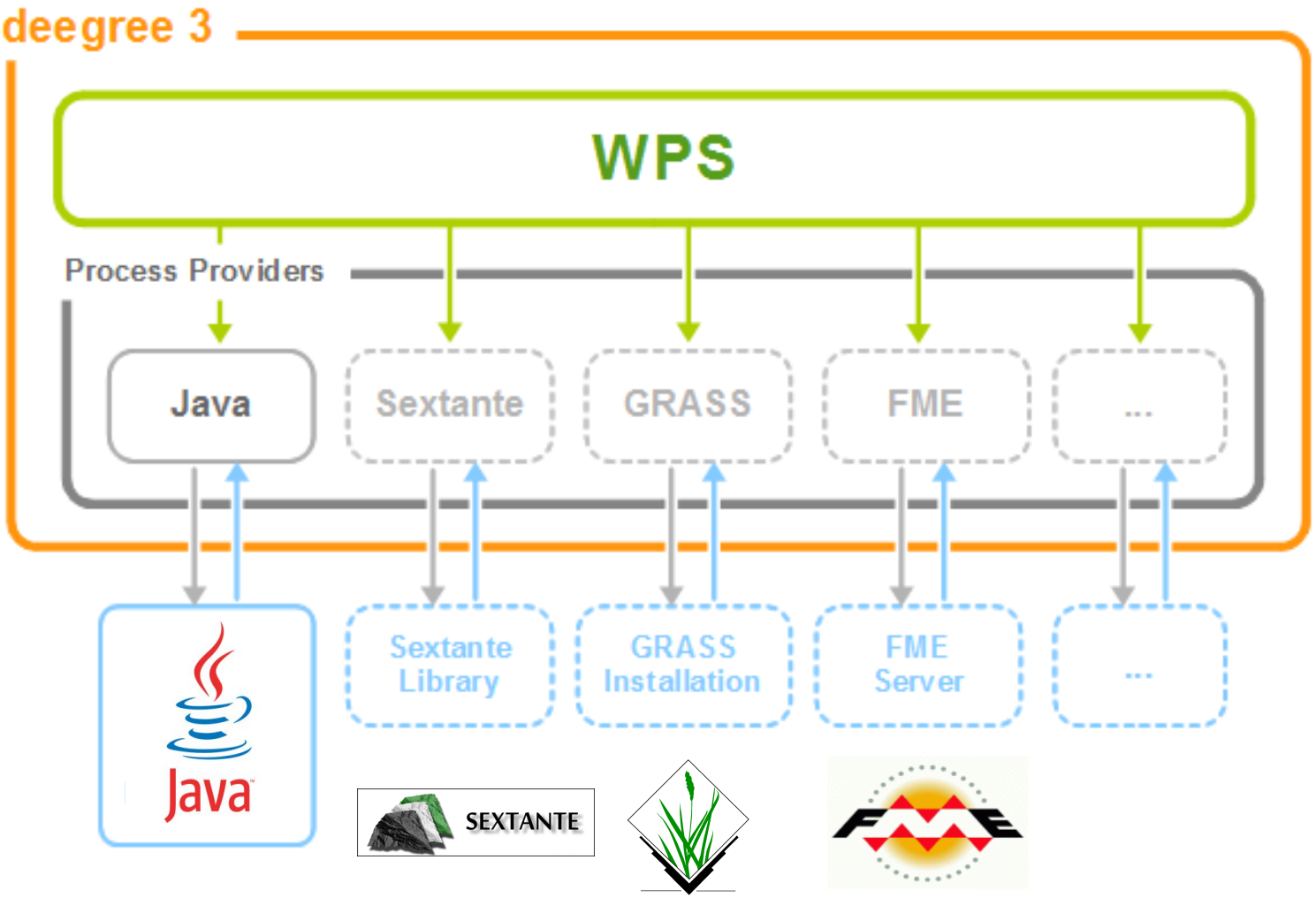
Get deegree 3 WPS up and running in 5 minutes

- System requirements
  - Java 6 (JDK)
  - Webcontainer (e.g. Tomcat 6)
- Download options
  - Now: WAR (Web Archive)
  - Soon: zip/tar.gz (with Tomcat)
- http://wiki.deegree.org → WPS

Features of the deegree 3 WPS

- Full implementation of the WPS 1.0.0 spec.
- Generic container for deploying processes via WPS
- Process API abstracts from protocol details
- Scales to gigabytes of data
  - Streaming access to complex inputs / outputs
- Integration of different process sources

WPS: Process providers



Options for integrating processes

- Now: Java-based process + XML process desc.
- Now: Write a process provider
- Soon: Use an available process provider
  - Sextante (work in progress)
  - GRASS (planned)
  - FME (planned)
  - ...

Write a Java-based process - steps

- Create a ProcessDefinition XML file
  - Identifier, Abstract, Title
  - Input, output definitions
  - XML Schema available for validation
- Implement deegree's Processlet interface
  - #execute (…)
  - #init (…), #destroy (...)

Write a Java-based process

- Working example: Simple buffering process
- Two inputs
  - GMLInput: Input GML geometry
  - BufferDistance: Buffer distance
- Single output
  - GMLOutput: Output GML data

XML process definition: Basics

```
<ProcessDefinition configVersion="0.5.0" processVersion="1.0.0"
  storeSupported="true" statusSupported="false">

  <Identifier>Buffer</Identifier>

  <JavaClass>org.deegree.wps.jts.BufferProcesslet</JavaClass>

  <Title>Process for creating a buffer around a GML geometry.</Title>

  <Abstract>The purpose of this process is to...</Abstract>

  <InputParameters>
    ...
  </InputParameters>

  <OutputParameters>
    ...
  </OutputParameters>

</ProcessDefinition>
```

XML process definition: Inputs

```
...
<InputParameters>

  <ComplexInput>
    <Identifier>GMLInput</Identifier>
    <DefaultFormat
      mimeType="text/xml"
schema="http://schemas.opengis/[..]/geometryComplexes.xsd" />
  </ComplexInput>

  <LiteralInput>
    <Identifier>BufferDistance</Identifier>
    <DataType
reference="http://www.w3.org/TR/xmlschema-2/#double">double</DataType>
    <DefaultUOM>unity</DefaultUOM>
  </LiteralInput>

</InputParameters>
...
```

XML process definition: Outputs

```
...
<OutputParameters>

  <ComplexOutput>
    <Identifier>BufferedGeometry</Identifier>
    <DefaultFormat
       mimeType="text/xml"
schema="http://schemas.opengis.net/[..]/geometryComplexes.xsd" />
  </ComplexOutput>

</OutputParameters>
...
```

Implementing the Processlet code

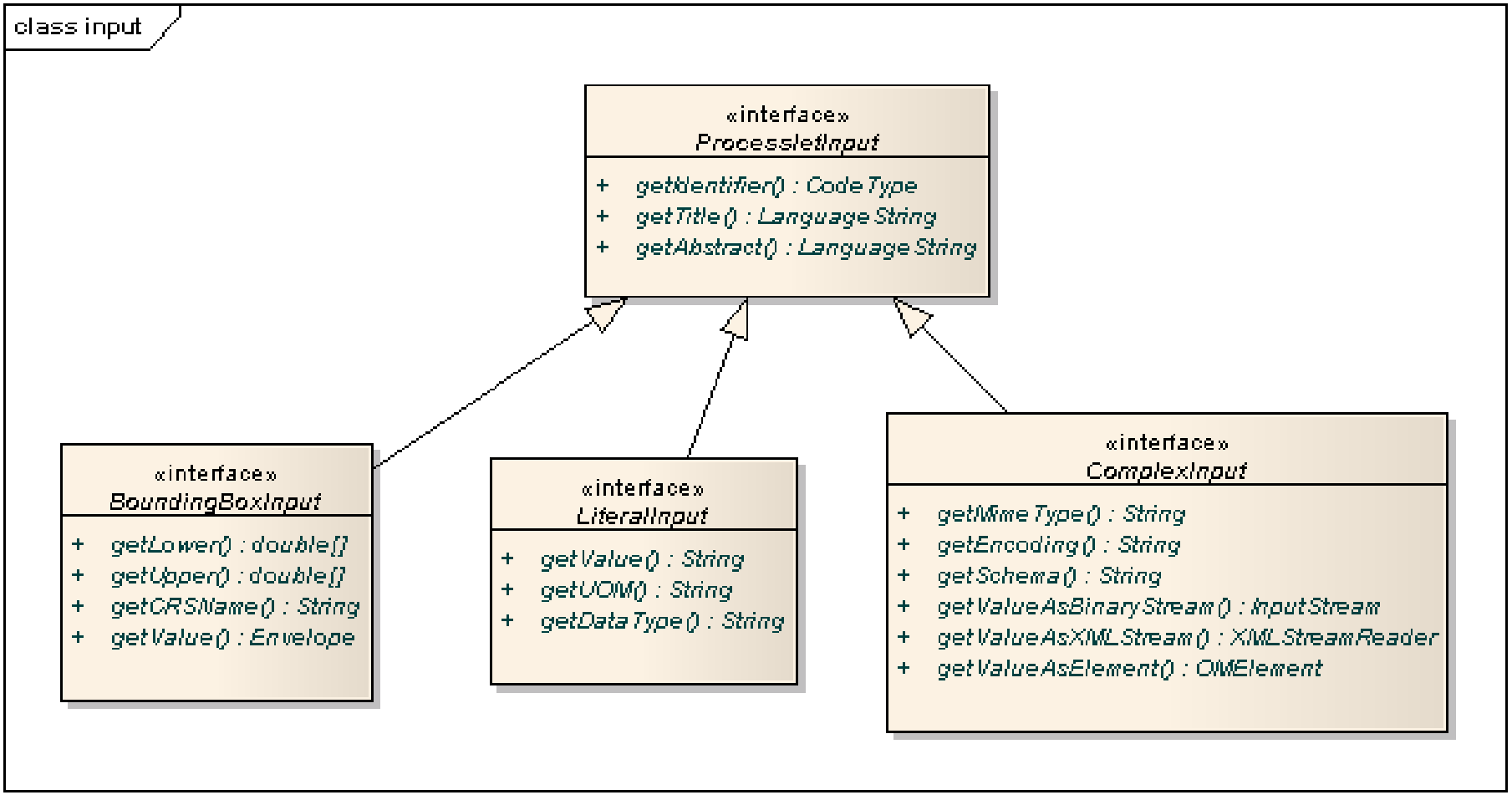
```
public class BufferProcesslet implements Processlet {

    @Override
    public void process( ProcessletInputs in, ProcessletOutputs out, ProcessletExecutionInfo info )
                            throws ProcessletException {

        // get buffer distance
        LiteralInput distanceInput = (LiteralInput) in.getParameter( "BufferDistance" );
        double bufferDistance = Double.parseDouble( distanceInput.getValue() );

        // get input GML parameter stream
        ComplexInput gmlInputGeometry = (ComplexInput) in.getParameter( "GMLInput" );
        XMLStreamReader xmlReader = gmlInputGeometry.getValueAsXMLStream();

        // get output GML parameter sink
        ComplexOutput gmlOutputGeometry = (ComplexOutput) out.getParameter( "BufferedGeometry" );
        XMLStreamWriter xmlWriter = gmlOutputGeometry.getXMLStreamWriter();

        // do the actual processing (parse GML input, buffer geometries, write GML output)
        ...
    }

    @Override
    public void destroy() {}

    @Override
    public void init() {}
}
```

Processlet#process(...)

```
public void process( ProcessletInputs in,
                     ProcessletOutputs out,
                     ProcessletExecutionInfo info )
                     throws ProcessletException;
```

- in: Access input parameters
- out: Access output parameter sinks
- info: Provide status information

ProcessletInput hierarchy



Processlet: Accessing inputs

```
...
// get input distance
LiteralInput distanceInput = (LiteralInput)
  in.getParameter( "BufferDistance" );

double bufferDistance =
  Double.parseDouble( distanceInput.getValue() );


// get input GML parameter stream
ComplexInput gmlInputGeometry = (ComplexInput)
  in.getParameter( "GMLInput" );

XMLStreamReader xmlReader = gmlInputGeometry.getValueAsXMLStream();
...
```

Processlet: Accessing outputs

```
...
ComplexOutput gmlOutputGeometry = (ComplexOutput)
  out.getParameter( "BufferedGeometry" );

XMLStreamWriter xmlWriter = gmlOutputGeometry.getXMLStreamWriter();
...
```

Processlet lifecycle: init() and destroy()

```
public void init();

public void destroy();
```

- Mimics Java Servlet's lifecycle concept
  - Only one Processlet instance is created
  - #init(): set up needed resources
  - #destroy(): clean up resources
- Well-tested for multi-threaded execution

Process code implementation

- Use the geospatial API you want:
  - deegree
  - GeoTools
  - JTS
  - …
- The deegree 3 WPS just handles the protocol
  - Access to inputs and outputs

What's next?

- Soon: Sextante integration finished
- Soon: More user-accessible web client
- Planned: FME integration
- Planned: Annotations for process description
- Mid-November (deegree day): deegree 3.0 final
- Post 3.0: Integrate processes into WMS / WFS
- Post 3.0: Support WPS 2.0.0 spec.

Thank you for your attention!

- Visit http://www.deegree.org
- Follow us on Twitter: @deegree_org
- Come to the deegree day 2010 (it's free)
  - November 16th -17th 2010
  - Bonn, Germany
  - http://deegreeday.deegree.org

Markus Schneider
schneider@lat-lon.de
http://www.lat-lon.de