Salvora Documentation

Contents

Introduction

Salvora is a tool to manage document collections, content, XForms, Xslt Transformations and XProc processes directly from local folders, without embedding them in a full application server solution.

You can use Salvora to gather data and modify documents using XForms both locally for personal use or from a browser in a VPN protected network for small groups. You can also use Salvora for prototyping document based applications or in order to prepare Xslt and XProc staging solutions. Salvora is not intended primarily for wide production environments, and by default the external access switch is disabled.

Currently, browsers tend to limit some of the functionalities of XForms when run directly from the local file system, and the current Html standard and browsers only support Xslt 1.0. It is easyer to run them in a client-server environment.

Salvora "serverl-less-server" can expose both read only or read/write folders as a REST service, perform Xlst transformations (2.0 and 3.0 also) and trigger XProc III processes. It is also able to print the folder file lists of configured folders.

Use Cases

Salvora is a simple and easy way to run XForms, Xslts and XProcs directly from local folders.

- Salvora primary purpose is to run XForms with simple configurations.
- Salvora can be used to build small document applications, to capture or share XForms based data like team work forms, document collections from some department or group, polls, etc.
- Salvora is useful for management data using semantic techniques as a step up from Excell spreadsheets. Cell based calculation tend to be error prone and dificult to debug on the long term, when manual iterations, data volume and changes accumulate over time.
- Salvora can also be used for testing and developing Forms and browser XSLTs without deployments. Also some processing capability is added using XProc iii triggers on file and transformation calls.

Quick Start

1. Download and paste the "fat" Jar file in the desired folder.
2. Add three subfolders (data, forms and xsltforms) with your data, xforms and the xsltforms distributions
3. Run the command : java -jar salvora-fat-<your-version>.jar
4. Your application should be running and your form should be available on localhost:8080/forms/<your-form-name>.xml

You can also dowload the sample application an run it with the mentioned command.

Command line usage

Salvora can be configured at start-time using several command line options. More complex document collections, transformations and processes can be described using a mapping file.

The usual data, forms and xsltforms folders can be configured with the -d -f and -x options. The output port can be configured with -p (example: -p:8081). By default Salvora runs on the 8080 port. Also you can prevent connections from other computers with the -e:false flag.

Download and run the executable or Jar file in the desired folder. The following examples show you several configuration combinations:

- java -jar salvora-fat-<version>.jar (Params)*
- java -jar salvora-fat-<version>.jar -f:forms -x:xforms -p:8081
- java -jar salvora-fat-<version>.jar -p:8085 -e:true

Using the command line can be configured Salvora manages three folders with different functions:

- xsltforms [Read only Configurable] : Predefined xsltform folder, intended for storing and exposing the xsltforms distributions from www.agencexml.com/xsltforms.
- form [Read only Configurable] : Predefined folder for your form.
- data [Read + Write Configurable] : Predefined writeable data folder.

Once the server is started you can open a browser on http://localhos:8080/*your_path* and start browsing.

Example paths:

- http://localhos:8080/xsltforms/xsltforms.xslt
- http://localhos:8080/form/myform.xml
- http://194.12.45.68:8080/data/mydata.xml

Mapping file configuration

Salvora provides a more powerful way to define http call mappings, transformations and processes. With a mapping file you can define:

- File folders, with or without write access and with or without printing the folder file list.
- Transformations and small transformation chains over local files and file collections.
- Transformations and small transformation chains over local http files and file collections.
- Transformations and small transformation chains over remote http files and file collections (SOON, not yet available).
- Triggers on http calls and transformations as Xproc iii pocesses.

At this moment the file mapping is being developed and documented. Please come back soon for improvements.

The mapping file structure is described with a Xsd Schema provided in the documentation of every Salvora version.

Configuration

Configuration options

Salvora first version started as a command line tool cappable of configuring thre main folders data (writable), form (for your XForms) and xsltforms for the XsltForms distribution. This basic configuration is maintained and you can use it as the easyest way to start with Salvora.

You can also define a configuration xml file where you can define document collections, transformations, processes with XProc triggers, map request parameters and path name portions to collections and transformations.

Command Line configuration.

The following parameters can be passed to the Server. :

Parameters.

- -h: Show this help message
- -f:<path> Form folder path. Default form. This param is not considered if a mapping file (-m) is provided.

- -x:<path> XSltForms folder path. Default xsltforms. This param is not considered if a mapping file (-m) is provided.
- -f:<path> Data folder path. Default data. This param is not considered if a mapping file (-m) is provided.
- -p:<number> Port number (integer). Default 8080
- -e:<boolean> true/false Enabled or disabled external access. Default false
- -m:<file path/name> The name of the mapping file. If the mapping file is provided, the command line defined folders are not taken in consideration. The access port number and external access param is considered even if a mapping file is provided.

Examples.

- java -jar salvora-<version>.jar
- java -cp salvora-<version>.jar net.vionta.salvora.server.Salvora -f:forms -x:xforms -p:8081
- java -jar salvora-<version>.jar -e:false
- java -jar salvora-<version>.jar -m:mapping-file.xml
- java -jar salvora-<version>.jar -m:mapping-file.xml -e:true -p:8082

Remember that if you use a mapping file, the command line defined folders, or even the default ones, are not considered.

Mapping Samples

The following element shows a listing sample of a request/mapping file, with folders configuration, transformations, etc.

The elements that maps requests have path (external exposed http paths) and urls (internal consumed internal-paths of the resources). We use two variants, BASE paths that expose an initial key of a set of files or resources and single paths that map to a single internal-paths or resource. When you use the "base-" paths it is meant for sets of paths and "path"/"rul" it is meant for single resources. For every base-path or single path there should be a corresponding base-internal-path or internal-path.

```
<?xml version="1.0" encoding="UTF-8"?>
<sal:salvora-application
    xmlns:sal="net:vionta:schemas:salvora:mapping:v1.0"
    >
  <collection
      name="configuration" base-path="configuration"
      write-allowed="false" file-list="false" >
    <description>Configuration path</description>
  </collection>
  <collection
      name="form" base-path="form"
      write-allowed="false"
      >
  </collection>
  <collection
      name="xsltforms" base-path="xsltforms"
      write-allowed="false" />
  <collection
      name="data" base-path="data"
      file-list="true"
      write-allowed="true" />
    <collection
      name="config" base-path="config"
      file-list="false"
      write-allowed="true" />
```

```
    <collection name="primera"
      internal-path="transformations/archivos-sample.xml"
      path="issues"
      type="local_file"
      >

 <step name="primera"
     source="transformations/lista-archivos.xsl" />

    </collection>


    <collection
 name="issues-index" base-path="index"  write-allowed="false" >
         <!-- a process trigger to be called when the file is requested.
        The (xproc iii) trigger can be called before or after the
  call is served -->
   <trigger name="file-indek" before="true"
     source="trigger/issue-list.xpl" />
    </collection>


</sal:salvora-application>
```

Before and After

Both transformations and triggers can be executed before or after the REST request is frocessed. By default both are performed before the content processed but you can set the "before" attribute to false so they are fired after the document request is processed.

This can be used to customize the file before is sent, save a copy or generate a index of certain files after it is stored, for example.

```
    <collection
 name="issues-index" base-path="index"   >
        <trigger name="file-indek" before="true"
     source="trigger/issue-list.xpl" />
    </collection>
```

Parameters

We have three kind of parameters that can be passed to the transformations and processes.

1. Regular parameters:

   literal parameters that can be passed with a fixed value.
2. Request parameters:

   The HTTP parameters that are sent as part of the request query string.
3. Path parameters:

   Path parameters can be inferred from the request path as name patterns.

Both Request and Path Parameters can be mapped with a different name in the transformation call from the one used in the request or the path definition

```
     <transformation
```

```
      name="xslt_transformation_test" type="local_file"
      path="options.xml" internal-path="conf/options.xml"  >
      <step name="instancecode" type="xslt" source="trigger/options-
view.xsl" >

 <request-parameter
     key="tuno" transformation-param-name="query-param" />
 <parameter
     key="param-test" value="'setted'" />
     </step>

  </transformation>


  <transformation name="form-config"
 type="local_file" path="form/:entity/:code.xml"  internal-
path="form/:entity.xml"  >

    <step name="instancecode" type="string" >
      <path-parameter
   key="entity" transformation-param-name="ENTITY" />
      <path-parameter
   key="code" transformation-param-name="INSTANCE_CODE" />
   <parameter key="TEMPLATE_LOCATION" value="collections" />
   </step>
  </transformation>
```

In the case of XProcs parameters are mapped by default to options by default but they can be declared to be used as input ports with the **input-port** parameter.

Troubleshooting

Requirements.

Salvora requires at least a Java 11 or 12 runtime. Due to jaxb library removal on java SE an issue has been found on jdk 13 and above.

Frequent Questions and answers.

Can the server be accessed from external computers ?

Yes, this is a small utility intended for occasional use. Write access is prevented for any other folder outside the data folder. The continuous usage in production machines is discouraged.

Can I open two instances on the same machine ?

Yes, you can configure different ports on different routes at the same time in the same machine.

Troubleshooting.

Address already bind.

This message indicates that the port number is in use. Choose a different port number or shutdown the other process and restart.

Wrong Java Version. "Class file version XX".

Salvora requires at least a Jdk 12 or newer. The following message indicates that a previous version of the Jdk is being used or configured. Review the Jdk configuration and try again.

Stack trace message:

java.lang.UnsupportedClassVersionError: net/vionta/xfserver/Salvora has been compiled by a more recent version of the Java Runtime (class file version 55.0), this version of the Java Runtime only recognizes class file versions up to 52.0

JAXB schema binding limitations:

The current version of the JAXB implementation has issues with the configuration file format and the way the schemas are referenced.

This schema reference will FAIL

```
<?xml version="1.0" encoding="UTF-8"?>
<salvora-application
  xmlns="net:vionta:schemas:salvora:mapping:v1.0"
  >
  <collection
    name="configuration" base-path="configuration"
    write-allowed="false" file-list="false" >
    <description>Configuration path</description>
  </collection>
  ...
</salvora-application>
```

This schema reference WORKS

```
<?xml version="1.0" encoding="UTF-8"?>
<sal:salvora-application
  xmlns:sal="net:vionta:schemas:salvora:mapping:v1.0"
  >
  <collection
    name="configuration" base-path="configuration"
    write-allowed="false" file-list="false" >
    <description>Configuration path</description>
  </collection>
  ...
</salvora-application>
```

We recommend you to use the supplied config xml files as templates and repport every problem you find and it does not figure on this documentation.

Schema Description

1. **salvora-application**

   Salvora Application Mapping Description. It takes file mappings (http requests over system files) and transformations (initially xslt chains).

   - **collection**
     - A file based mapping to describe files and file based http requests. It also takes the option to serve folder file list files.
       - **collection-atts**
         - @ name

        - @ base-path [required]

          Folder base path for file access.
        - @ base-internal-path [required]

          Base Internal base path.
        - @ path

          Individual path for file access.
        - @ internal-path

          Internal Path, related to the path.
        - @ file-list [optional]

          Describes if a file list should be returned when requesting the folder path. Defaults to false.
        - @ write-allowed [optional]

          Describes if POST and PUT operations (write opperations) are allowed.
      - **description-info**
      - **Element: trigger**
    - **step**
      - **transformation-step**
        - **transformation-step-attributes**
          - @ name
          - @ source [required]
          - @ type [required] :

            Values xslt | xquery | string
        - **transformation-step-elements**
          - **path-parameter**
          - **request-parameter**
          - **transformation-parameter**

2. **Shared Elements**

   - **path-parameter**
     - **path-parameter**

       Parameter element, used to map request internal-path path parts to xslt parameters.

       - @ key
       - @ xslt-param-name
       - @ default

         Default Value
       - @ value

         In some cases we need to specify a value template that will be modified with the user provided parameter value. At this point the current parameter name, preceded with ":", will be modified with the provided value in the call.

         For example, a parameter defined as /somepath/:desiredfile.html could be taken in the desiredfile path parameter and used to adjust a value template like data/:desiredfile.xml.

         This is useful when you want to use a file as input port for some predefined xproc or xslt.
       - @ input-port (false)

         Indicates if the param should be treated as an input port (on XProc scripts). Defaults to false.
   - **request-parameter**

- **request-parameter**
  - @ key
  - @ xslt-param-name
  - @ default

    Default Value
  - @ value

    In some cases we need to specify a value template that will be modified with the user provided parameter value. At this point the current parameter name will be modified with the provided value in the call.

    For example, a "desiredfile" query parameter could be used to adjust a value template like data/:desiredfile.xml.

    This is useful when you want to use a file as input port for some predefined xproc or xslt.
  - @ input-port (false)

    Indicates if the param should be treated as an input port (on XProc scripts). Defaults to false.
- **parameter**

  A explicit parameter with name and value. The value is a plain string. It is not interpreted.
  - @ name
  - @ value
  - @ input-port (false)

    Indicates if the param should be treated as an input port (on XProc scripts). Defaults to false.
- **trigger**

  A explicit parameter with name and value. The value is a plain string. It is not interpreted.
  - @ name
  - @ source: The xproc iii file path
  - @ before[boolean]: Indicates if the process should be performed before or after the main request.

Change List

- **2.0.0**
  - **Simplified mapping file schema**. File Mapping has been simplified by removing transformation element that is similar to collection. Collections can take now both triggers and transformations.
  - **XProc error log**. XProc error logs are now sent to the log as errors making easier to spot XProc III errors.
- **Beta 0.101.0**
  - **Input ports on triggers**. Parameters can be now mapped to input ports (by default they are asigned to options) on XProcs.
- **Beta 0.9.4**
  - JRE > 12

    . JAXB dependencies has been added so the tool can run on JRE environments after JRE11.
  - Default Request Parameter Value

    . A default value can be added to the query parameters.
- **Beta 0.9.3**
  - Relative paths for XProc files

. Trigger source file paths are now defined as relative to the salvora run folder.

- Updated documentation

. Explicit changelists

- Request and Path parameters on triggers

. We can use now request, path, and explicit parameters on triggers.

- URL calculation bugfixes

. An error on url calculations mixing base paths and path parameters.

- **Beta 0.9.2**
  - Updated sample application.
  -
  -

Contact

Salvora is provided by www.neonbluetide.com. Neon Blue Tide is part of www.vionta.net.