Installation Guide

for

smartEditor

version 1.0

Revision History

Revision Number	Date of Publication	Author(s)	Change Description
1.0	28.01.2011	Stefan Blume	initialization, general description of setting up the software
1.0	02.02.2011	Henning Bredel	Minor additions
1.1	07.02.2001	Kristian Senkler	- Added license text to preamble - Checked spelling.

Editor(s)

Stefan Blume
con terra GmbH
Martin-Luther-King-Weg 24
48155 Muenster, Germany
Email: s.blume@conterra.de

License

This document is part of 52°North.

Contact: Andreas Wytzisk
52°North Initiative for Geospatial Open Source Software GmbH,
Martin-Luther-King-Weg 24,
48155 Münster, Germany, info@52north.org

For more information, contact:
52°North Initiative for Geospatial Open Source Software GmbH
Martin-Luther-King-Weg 24
48155 Münster, Germany
http://52north.org

Table of Contents

1 Introduction

Smart Editor is a web editor integrating both the INSPIRE metadata publication queryables for services, series, and data and the ISO 19139 metadata implementation of ISO 19115 and ISO 19119 as well.

This document describes the installation and configuration of smartEditor application as you download from 52n smartEditor web page (http://52north.org/communities/metadata).

2 Prerequisites - System requirements

2.1 Database support

smartEditor supports the following databases:

- Oracle 10
- Oracle 11g
- PostgreSQL 8.3, PostgreSQL 8.4
- MS SQL Server

2.2 Servlet Containers

- Apache Tomcat 5.5
- Apache Tomcat 6
- Oracle WebLogic 10.1
- Oracle WebLogic 11g (10.3.4)
- Sun GlassFish 2.2.1
- Sun GlassFish 3.0.1

This document assumes you are using Apache Tomcat Servlet Container.

2.3 Browsers

- Firefox 3
- Internet Explorer 7
- Internet Explorer 8
- Limited support for Google Chrome and Safari

3 Installation

There are two ways to get the smartEditor and deploy it into your servlet container respectively. For running it within your servlet container it is sufficient to download the binary distribution (see Section 3.1.1). If you want to have look into the code or make changes, you might be interested in checking out the sources via maven (see Section 3.1.2).

3.1 Introduction

3.1.1 About the downloaded binary

The smartEditor binary currently available from the 52n web page is a pre-compiled web application packaged as a war archive. The file name is:

```
smarteditor-webapp-<version>.war
```

The binary file is a zip archive having the file extension `.war`. For the configuration of this file we suggest to unpack it, apply the proposed changes in this document and put it onto your servlet container afterwards.

3.1.2 About checking out the sources

The sources are hosted by the metadata community's subversion repository of 52°North. Check out the sources via your favourite SVN client like the following console example:

```
$ svn co https://svn.52north.org/svn/metadata/smarteditor/trunk/ editor
```

Having the Maven tool in your system path, you can build the web application via the maven goal:

```
$ mvn install
```

After that, you should find the `smarteditor-webapp-<version>.war` file in the `editor/smartEditor/target` directory.

3.1.3 Internal file structure

The binary distribution contains a set of folders which will be explained in the following:

`images\`	Contains all images used in smartEditor
`js\`	Contains all internal JavaScript resources
`META-INF\`	Contains context information and information about the build process
`styles\`	Contains all CSS files for rendering the UI
`tooltips\`	Bundles the html tooltips on metadata elements (multi lingual)
`WEB-INF\`	

classes\	Contains configuration and transformation files and message resources as well
defs\	Configuration files for Apache Tiles page layout
config\sql	SQL files for database setup
jsp\	All Java Server Pages constituting the web frontend
lib\	Set of internal and third-party libraries (jar files)
tld\	Used tag libraries
beans-definitions.xml	Configuration of metadata beans
dispatcher-servlet.xml	Main Spring Framework configuration file.
dao-definitions.xml	Definition of businesstier beans
web.xml	Servlet Container Deployment Descriptor
weblogic.xml	Deployment Descriptor Extension for Oracle WebLogic Server
THIRD_PARTY_LICENSES.txt	List of all third-party libraries and licenses used
index.jsp	Welcome Page

This folder will be referenced as `$EDITOR_HOME` in the next chapters.

3.2 Database preparation

smartEditor has support for Oracle, PostgreSQL and SQL Server. Please run the scripts according to your database environment. The scripts can be obtained from

https://svn.52north.org/svn/metadata/smarteditor/trunk/smarteditor-api/src/main/config/sql/

Additionally these SQL files are part of the binary distribution, see folder `WEB-INF/config/sql`.

It is recommended to create the tables for an appropriate database user. If no user exists, create e.g. a user “smartEditor” and a belonging tablespace.

The following SQL Scripts are shipped with the distribution:

- oracle.sql
- postgres.sql
- sqlserver.sql

The script creates two tables (`SAVED_TEMPLATES` and `LOCKING`) for the storage of smartEditor metadata templates and for maintaining the list of “locked” metadata documents currently being edited by another user.

3.3 Servlet container configuration (Apache Tomcat)

This Apache Tomcat folder will be referenced as `$TOMCAT_HOME` in the next chapters.

3.3.1 Container managed database connections - JNDI configuration

We recommend letting the servlet container provide the management of JDBC connections to your underlying database.

For Apache Tomcat this can be done by creating a global resource configuration.

Open the file `$TOMCAT_HOME\conf\server.xml` and add the following snippet to the `<GlobalNamingResources>`-Element:

```
<Resource name="editor" auth="Container" type="javax.sql.DataSource"

            maxActive="100" maxIdle="0" maxWait="10000"

            defaultAutoCommit="false"

            username="[db user]" password="[db password]"

            driverClassName="oracle.jdbc.OracleDriver"

            url="jdbc:oracle:thin:@[database host]:1521:[SID]"/>
```

Note: This snippet has the Oracle JDBC driver class and JDBC URL connection template pre-configured.

3.3.2 Copy JDBC Database Driver

When using container managed database connections the jar file containing the database vendor specific JDBC implementation needs to be accessible to the container.

The JDBC driver that matches the database must be copied into the directory `$TOMCAT_HOME/common/lib`. The driver appropriate to the database product used is usually supplied with the product and can also be downloaded from the manufacturer's website.

3.3.3 Context configuration

If you do not want to run smartEditor within your `$TOMCAT_HOME/webapps` folder, it is necessary to create a `smartEditor.xml` context file to be put into the `$TOMCAT_HOME/conf/Catalina/localhost` folder.

The file definition looks like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<Context reloadable="true"
         crossContext="false"
         antiJARLocking="false"
         docBase="[path to smartEditor]" debug="0"
         antiResourceLocking="false">
```

```
    <!-- jndi support -->
    <ResourceLink name="jdbc/editor"
                  global="editor"
                  type="javax.sql.DataSource"/>
    <!-- disable persistent sessions -->
    <Manager distributable="false"
             saveOnRestart="false"
             pathname=""/>
</Context>
```

Please note the `docBase` attribute that needs to point to the smartEditor folder. The ResourceLink provides a web application internal JNDI resource named `jdbc/editor` but delegates it to the global resource `editor` configured in chapter 3.3.1.

3.4 smartEditor application configuration

3.4.1 Configuring application.properties

The file `$EDITOR_HOME/WEB-INF/classes/application.properties` is the main configuration file for the whole smartEditor application. The following table shows all configuration attributes and adds some explanatory information:

attribute name	default value	explanation
locale.default	en-GB	default application locale
csw.discovery	<none>	URL to the CSW discovery service bound to this smartEditor instance (necessary when starting up with an ISO fileIdentifier
csw.manager	<none>	URL to the CSW-T endpoint. smartEditor will send documents to this URL when publishing metadata
import.xslt.dir	/internal/import	internal configuration (do not change)
external.xslt.dir	/internal/external	internal configuration (do not change)
dictionary.resource.xml	/codelist_enumeration.xml	internal configuration (do not change)
dictionary.resource.messages	/isolist	internal configuration (do not change)

map.service.url	http://vmap0.tiles.osgeo.org/wms/vmap0?	default OGC WMS map service to be displayed in the smartEditor
layer.names	Basic	layer names to be displayer
bbox.minx	-30.0	default BBOX parameter
bbox.miny	35.0	
bbox.maxx	50.0	
bbox.maxy	70.0	
projection	EPSG:4326	default CRS
image.format	image/png	default image output format for map widget
db.jndi.name	java:comp/env/jdbc/editor	default lookup name for container managed database resource
db.hibernate.dialect	org.hibernate.dialect.Oracle9iDialect org.hibernate.dialect.Oracle10gDialect org.hibernate.dialect.PostgreSQLDialect org.hibernate.dialect.MySQLDialect	specific SQL dialect implementation for hibernate, choose the one that fits to your database vendor
db.hibernate.schemaUpdate	False	do not allow hibernate to create tables if they do not exist
db.hibernate.showSQL	False	debug SQL statements of hibernate
when running smartEditor in conjunction with 52n security components or sdi.suite securityManager these parameter needs to be defined for a proper validation of the SAML tokens		
security.keystore.pwd	defaults to 'changeit'	password to access the keystore
security.keystore.path	/.keystore	path to the keystore containing the certificate that is used to sign the SAML assertions within the SAML token. Path needs to be classpath

		resolvable
security.certificate.alias	for example “gpt-security” if integrating with ESRI Geoportal Server	alias of the named certificate
security.certificate.pwd	defaults to ‘changeit’	password to access the certificate

3.4.2 Configuring the log level of smartEditor

You may need to increase the log level of smartEditor. Therefore open the file `$EDITOR_HOME/WEB-INF/classes/log4j.xml`.

Basically the file contains the following information:

```
<!DOCTYPE log4j:configuration SYSTEM "log4j.dtd">

<log4j:configuration xmlns:log4j="http://jakarta.apache.org/log4j/">
    <appender name="CONSOLE" class="org.apache.log4j.ConsoleAppender">
        <param name="Threshold" value="WARN"/>
        <layout class="org.apache.log4j.PatternLayout">
            <param name="ConversionPattern"
value="[%-5p] %d [%t] %c - %m%n"/>
        </layout>
    </appender>
    <appender name="LOGFILE" class="org.apache.log4j.RollingFileAppender">
        <param name="File" value="${catalina.base}/logs/smarteditor-
webapp.log"/>
        <param name="Threshold" value="INFO"/>
        <param name="MaxFileSize" value="5000KB"/>
        <param name="MaxBackupIndex" value="3"/>
        <layout class="org.apache.log4j.PatternLayout">
            <param name="ConversionPattern" value="[%-5p] %d{ISO8601} [%t]
%c - %m%n"/>
        </layout>
    </appender>
    <logger name="de.conterra.smarteditor">
        <level value="INFO"/>
    </logger>
    <logger name="org.springframework">
        <level value="WARN"/>
    </logger>
    <root>
        <level value="WARN"/>
        <appender-ref ref="CONSOLE"/>
        <appender-ref ref="LOGFILE"/>
    </root>
</log4j:configuration>
```

If necessary the output location of the logfile can be changed. The level of detail of the logging can be determined by setting the level value from `INFO` to `DEBUG`.

4 Test application

After carrying out the steps described in the chapters above (re)start your Apache Tomcat and open a Browser. Type `http://[host]:port/smartEditor` and see the application start.

Enjoy!