

ENDORSE

Deliverable D4.3

Rule Engine – Preliminary Implementation

Editor:	Thomas Kurz, Christoph Rücker, Thomas Lampoltshammer, Thomas Heistracher
Deliverable nature:	Report, Code Deliverable
Dissemination level: (Confidentiality)	Public
Contractual delivery date:	01.11.2011
Actual delivery date:	15.11.2011
Suggested readers:	Consortium
Version:	5.0 (14.10.2011)
Total number of pages:	24
Keywords:	PRDL, Rule Engine, Rules, Deployment

Abstract

This deliverable discusses the preliminary rule engine and execution environment implementation of the ENDORSE project. It starts with the preliminary ENDORSE architecture and the architecture of the prototype are discussed. Based on the requirements defined within the project the capabilities and functionality of the prototype is elaborated. To illustrate the approach a use case definition is given and the prototype functionality is shown. An installation guide of the required server and database environment to run the prototype is given. As an outlook the future work on integrating the rule engine environment into the final architecture is described.

Disclaimer

This document contains material, which is the copyright of certain ENDORSE consortium parties, and is subject to restrictions as follows:

This document is published under Creative Commons Attribution-NonCommercial-ShareAlike 3.0 License.



<http://creativecommons.org/licenses/by-nc-sa/3.0/>

Neither the ENDORSE consortium as a whole, nor a certain party of the ENDORSE consortium warrant that the information contained in this document is capable of use, or that use of the information is free from risk, and accept no liability for loss or damage suffered by any person using this information.

Impressum

Legal Technical Framework for Privacy Preserving Data Management

ENDORSE

Workpackage 4

D4.3 Rule Engine – Preliminary Implementation

Editor: Thomas Kurz, SUAS, Christoph Rücker, SUAS, Thomas Lampoltshammer, SUAS, Thomas Heistracher, SUAS.

Work-package leader: Mark McLaughlin, WIT

Estimation of PM spent on the Deliverable: 5

Copyright notice

© 2011 ENDORSE Consortium

Executive summary

The deliverable at hand presents the first effort towards a rule creating and execution environment for the ENDORSE project. Focusing on the stakeholders and the requirements described in D2.4 the privacy rule definition language was created. To boost the development of the ENDORSE framework this deliverable explains the initial prototypical implementation for an environment. This deliverable shows a very early state of the rule engine development and there are many aspects which cannot be covered such as the resolving strategies for rule conflicts, interlinks between rules and the corresponding law texts and direct conversion from PRDL rules into the Drools rule language which is executable by the engine.

The prototype described in the deliverable can show that a privacy statement can be implemented within rules and also executed within a rule engine to prevent data from being used not according to the privacy statement. As the task is ongoing further work will be conducted towards an integrated framework which handles rules and assures that the data are used fairly and lawfully.

List of authors

Participant	Author
SUAS	Thomas Kurz
SUAS	Christoph Rücker
SUAS	Thomas Lampoltshammer
SUAS	Thomas Heistracher
Internal Reviewer	Author
SOLUTA	Pierfranco Ferronato

Table of Contents

1. Introduction.....	6
2. Overall Picture of the Prototype.....	7
3. ENDORSE and Prototype Architecture.....	8
4. Rule Engine Prototype User Guide.....	9
4.1. Prototype Components.....	9
4.1.1. Login Screen.....	9
4.1.2. Create an account.....	10
4.1.3. Start Screen.....	11
4.1.4. Rule Browser.....	11
4.1.5. Rule Editor.....	12
4.1.6. Request Editor.....	13
4.1.7. Data Browser.....	14
4.2. Prototype Use Case.....	16
4.2.1. User enters a rule.....	16
4.2.2. User specifies a request.....	17
4.2.3. User executes a request.....	17
4.3. Future Development.....	18
5. Conclusion and Outlook.....	19
6. Installation & Setup.....	20
6.1. Installation of the Components.....	20
6.2. Application Server Setup.....	20
7. References.....	24

Table of Figures

Figure 1: ENDORSE Architecture.....	8
Figure 2: Prototype Initial Screen.....	9
Figure 3: Prototype Login.....	10
Figure 4: Prototype Account Creation.....	10
Figure 5: Prototype Start Screen.....	11
Figure 6: Prototype Rule Viewer.....	12
Figure 7: Prototype Rule Editor.....	13
Figure 8: Prototype Request Configuration.....	14
Figure 9: Prototype Data Browser.....	15
Figure 10: UseCase Rule Browser.....	16
Figure 11: Use Case Configure Request.....	17
Figure 12: UseCase Data Browser.....	18

1. Introduction

This deliverable describes the Rule engine preliminary implementation in the context of the ENDORSE project. The aim of the ENDORSE project is to provide interconnection between policy makers, the organisations which handle the data and last but not least the data subjects via pioneered conceptions regarding legally compliant data handling. In order to achieve this goal ENDORSE will introduce a PRDL that provides solutions for organisations that want to migrate their existing privacy policies into computationally executable rules enforced by the ENDORSE framework.

The *Privacy Rule Definition Language* (PRDL) is one of the main goals of the ENDORSE project. The language should be sufficiently expressive to define privacy policies for SMEs, it should link the wording of the data privacy laws of different European countries, it should be represented in natural language and therefore should be easy to understand. After all, it should be automatically or semiautomatically executable by a rule engine.

This deliverable deals with the first prototype of the rule engine and execution environment for the ENDORSE project and introduces the basic components needed within this environment. It shows all the components of the prototype and their basic functionality. In brief, these elements are an environment to create, edit and delete rules within the rule repository. In addition the prototype offers an environment to edit and save a request which can be executed within the rule engine. Consequently the application presents an interface where the user can check what data the system has return on the executed request and consequently whether the request was valid or not. The software prototype presented in the deliverable is available online at <http://tinyurl.com/5roe3th> and the deployable file is available at <http://tinyurl.com/67s9lmq>.

The remainder of this document is structured as follows. Chapter 2 gives an overall picture of the rules engine prototype and explains the interrelations of the prototype within the project. Chapter 3 then summarizes and discusses the preliminary architecture of the ENDORSE project and the architecture of the prototype application. Chapter 4 is split in two parts. The first one contains the description of all the prototype components while the second part is dedicated to the description of use case as well as future developments. Chapter 5 gives an outlook and conclusions on the current development and on future plans towards the final implementation of the rule engine within ENDORSE. Chapter 6 illustrates the installation and setup of the server environment needed to deploy the prototype application.

2. Overall Picture of the Rule Engine Prototype

The ENDORSE project aims on enforcing privacy protection law within companies by providing a language PRDL for the generation of rules according to privacy law and a toolbox that can enforce this rules within a company system. After the development of the preliminary PRDL specification this deliverable deals with the first prototype of the rule enforcing environment. The prototype should serve as a first step towards the complete ENDORSE architecture. From this starting point all the efforts concerning architectural components should be aligned and bring together the PRDL rule editor, the rule enforcing environment and the request editor and a consent component. The complete architecture is described in D3.1. This prototype includes the rule enforcing environment represented by the Drools Rule Engine and the request editor implemented with a small web based GUI.

The consortium decided the usage of Drools[7] during the general meeting in Zaragoza. Drools is an Object-Oriented Rule Engine for Java developed by JBoss. In addition, the rule engine is an augmented implementation of Forgy's Rete[8] algorithm tailored for the Java language. Adapting Rete to an object-oriented interface introduces more natural expression of business rules with regards to business objects. Furthermore it provides declarative logic programming and is flexible enough to match the semantics of different problem domains such as data privacy in ENDORSE context.

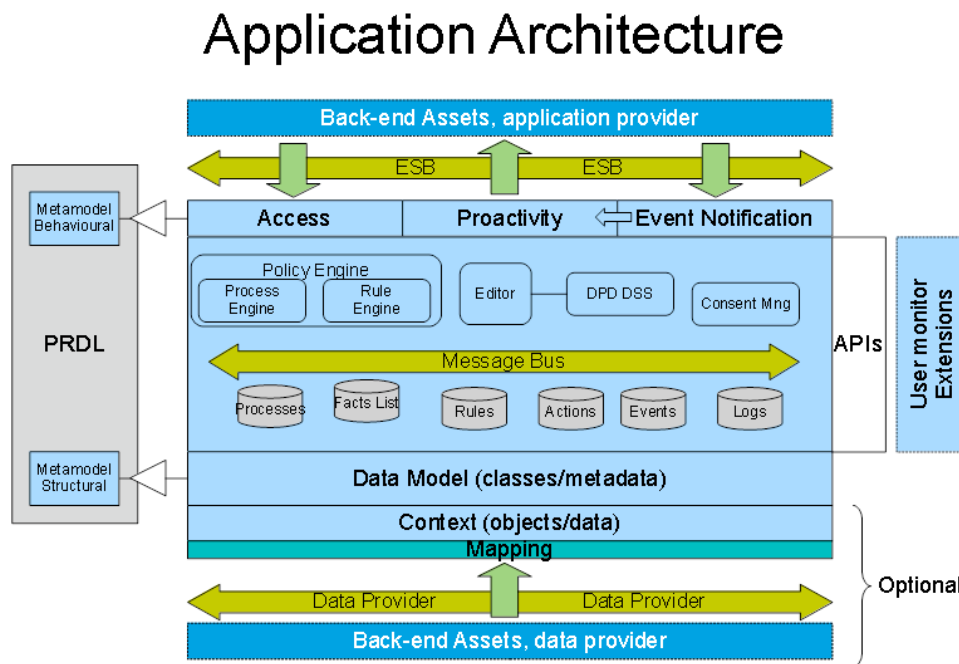
The Drools rule engine should provide the needed functionality for the ENDORSE project to enforce privacy rules within a company environment. The engine can be deployed as a standalone application or as a service on a server within a company network. Rules that can be enforced within the engine have to be written in Drools rule language but this language is not suitable for non technical personal which legitimates the introduction and development of the PRDL language. To enforce PRDL rules within the rule engine, the language has to be transformed. For the prototype implementation, the transformation has not been feasible but for the demonstrator one rule will be transformed as a proof of concept.

To give an overview the prototype is designed as a web based standalone application. It is deployed on a Glassfish open source application server and uses an Apache Derby DB for internal data persistence. Developed in Java and using many Java Enterprise technologies such as Java Server pages, Java Beans and Hibernate persistence the application can be seen as state of the art. It combines the rule engine, an editor for writing Drools rules and a small application for writing a request and execute it with the system. The rule engine is embedded into the application and is initialized once and every time the rule base changes, the rule engine is reinitialized to assure that all the rules are deployed in the engine when a request is created. When a user logs into the application, he can create rules in Drools language which are persistent within the system. Furthermore, the user can specify a request for data from the application example data. In case of the demonstrator this data repository consists of files on the deploying system. When browsing the files the rule engine executes the request and shows the files which valid to show according to the rules within the engine.

A use case and the user interface is described in the following chapters as well as the installation guide for the needed server environment. The demonstrator can be seen as a starting point for the further development of the rule executing and enforcing environment. Efforts will be made in the integration of the rule engine into the overall ENDORSE architecture.

3. ENDORSE and Prototype Architecture

The application architecture of the ENDORSE framework and toolset has been defined to fit all the requirements identified by the consortium, described in the deliverables 2.1 – 2.4. Figure 1 illustrates the preliminary architecture and shows the relation of the rule or policy engine and the editor within the project. As the architecture component defining process is still on-going and the interfaces to other components are not entirely defined, the preliminary implementation of the rule engine has been conducted into a standalone web application. Adapting the provided functionality into a fully fledged architecture will not impose critical problems and the architecture will not be affected. A fully description of the ENDORSE architecture can be found in deliverable D3.1 Functional Architecture.



Permalink: February 2011 Software Network: Endorse Project, Creative Commons: Share Alike, Non-Commercial with Attribution

Figure 1: ENDORSE Architecture [9]

As stated before, the prototype has its own internal architecture which is decoupled from the ENDORSE architecture. The central component is the rule or policy engine Drools [7]. This rule engine is developed by JBoss¹ and also provides a community developed open source variant. The rule engine component in the prototype is embedded in the application and does not need a running JBoss instance. To persistent store rules an Apache Derby database is used as repository. The prototype itself is an web application based on Enterprise Java Beans[5] and Java Server Pages [6]. It can be deployed on any web server and the installation and setup manual is given in Chapter 2. The following chapter will describe the functionality and components of the rule implementation prototype followed by usage scenarios.

¹<http://www.jboss.org/>

4. Rule Engine Prototype User Guide

This section illustrates the components of the rule engine preliminary implementation and shows the main features of the prototype. The prototype was developed to show the capabilities of the Drools Rule Engine and to start the development of the rule engine component within the ENDORSE architecture. Although the prototype is currently developed as a standalone application the components should fit into the future ENDORSE architecture. Additionally this section introduces a manual for the developed prototype and explains the capabilities for the end user. The application should furthermore serve as a basis for tackling the goals and requirements of the ENDORSE project elaborated in the deliverables 2.1 – 2.4. Besides the efforts concerning the rule engine also the process of developing PRDL is ongoing. The meta access model (see D.3.2) developed to represent the request a data controller wants to execute on a system has taken into account and is implemented within the prototype application. The next section will visualize all the components of the application and provide a user guide.

4.1. Prototype Components

The chapter shows the components and their specific contexts in relation to the project and the project architecture. Also future developments will be announced and the importance of further developments is stated. This prototype will lead to an integrated approach with all the other components developed within the project and will end up in a toolset together with the PRDL editor and other components needed to integrate the ENDORSE framework into existing company environments. The next chapter will focus on a use case model that can be accomplished with the rule engine prototype.

Starting of with the different views of the application user interface this chapter will introduce all the features that the rule engine prototype can offer the user. It will also give further information concerning future development of the language and the meta-models. To start with, the first screen that is showed within the browser when starting the web application is started, is given in Figure 2. The user has two choices to go forward. On the one hand, if an account exists, to go directly to the login or on the other hand to register an account within the system. Accounts are stored in the database and are available for later usage.

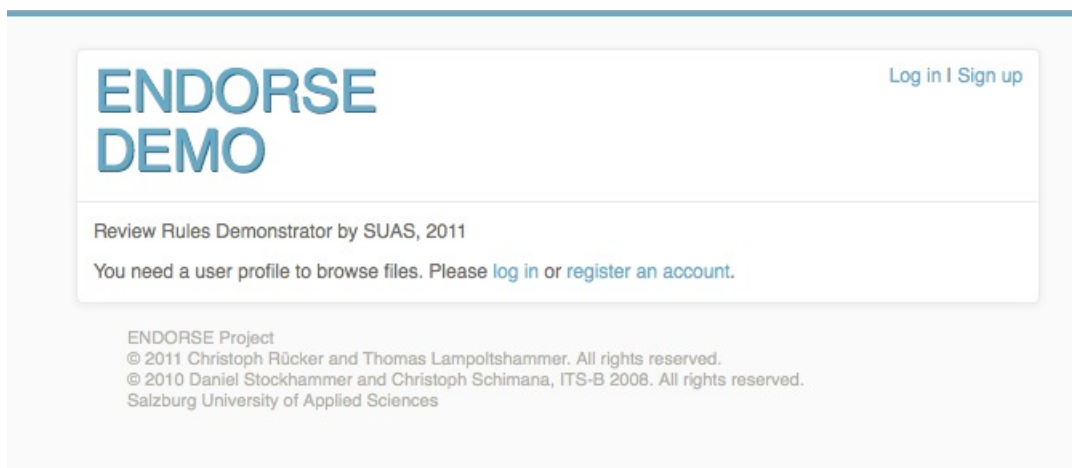
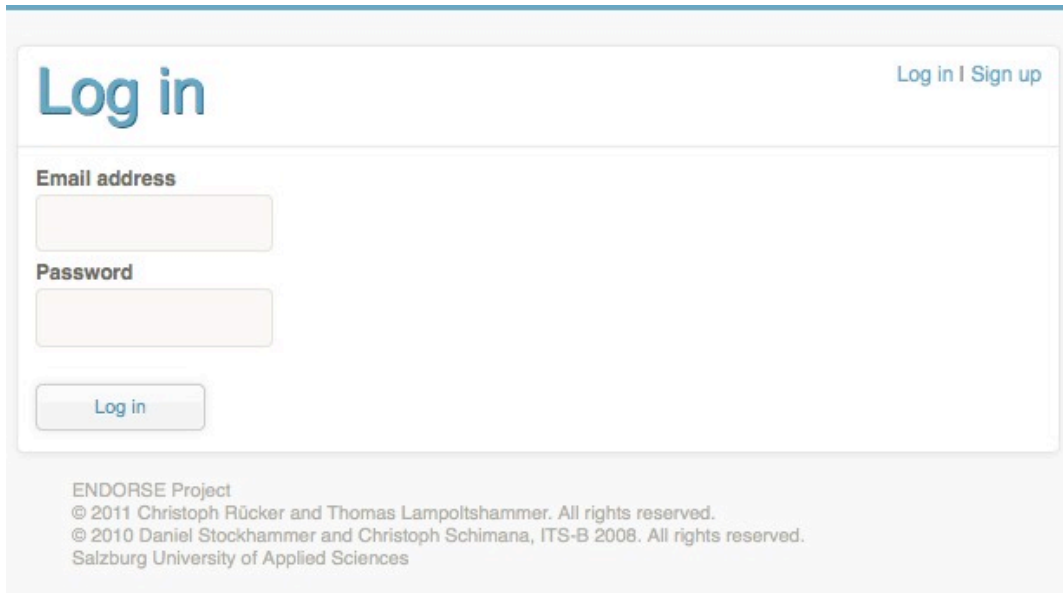


Figure 2: Prototype Initial Screen

4.1.1. Login Screen

When the user has chosen to log in because she has an account the system will lead to the login screen depicted in Figure 3. The user has to state the email address used for the registration and the valid password to the online form. When the credentials are valid the system will login the user and show the start screen of the application. In case of wrong credentials the system will display an credential error and will ask for the retyping of the credentials. In case the user has not created an account yet the application offers the possibility to do so.

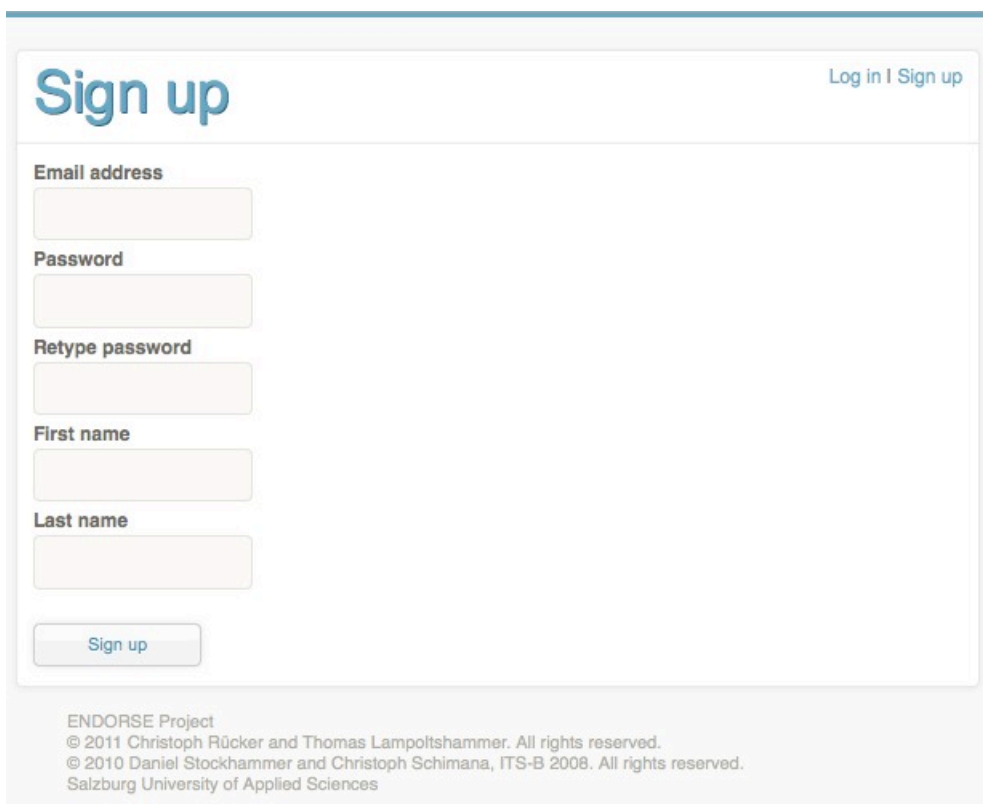


The image shows a web form titled "Log in" in a large blue font. In the top right corner, there are links "Log in" and "Sign up". The form contains two input fields: "Email address" and "Password", both with light orange borders. Below these fields is a "Log in" button with a blue border. At the bottom of the form, there is a footer with the text: "ENDORSE Project", "© 2011 Christoph Rücker and Thomas Lampoltshammer. All rights reserved.", "© 2010 Daniel Stockhammer and Christoph Schimana, ITS-B 2008. All rights reserved.", and "Salzburg University of Applied Sciences".

Figure 3: Prototype Login

4.1.2. Create an account

To assure that only registered user can work with the example data in the prototype the login with credentials is obligatory. In order to create a valid account the user has to specify an email address and a password. Additionally the first and last name of the user is stored with the credentials to make the person identifiable. For a signup the user has to type all the parameters in the online form and click the Sign Up Button shown in Figure 4. When the registration or the login is completed successfully the start screen of the application is shown and described in the next section.



The image shows a web form titled "Sign up" in a large blue font. In the top right corner, there are links "Log in" and "Sign up". The form contains five input fields: "Email address", "Password", "Retype password", "First name", and "Last name", all with light orange borders. Below these fields is a "Sign up" button with a blue border. At the bottom of the form, there is a footer with the text: "ENDORSE Project", "© 2011 Christoph Rücker and Thomas Lampoltshammer. All rights reserved.", "© 2010 Daniel Stockhammer and Christoph Schimana, ITS-B 2008. All rights reserved.", and "Salzburg University of Applied Sciences".

Figure 4: Prototype Account Creation

4.1.3. Start Screen

After a successful login the first screen appearing gives the user the possibility to change his first and last name and the navigation within the application is shown as illustrated in Figure 5. The navigation section in the top right corner informs the user of who is logged in and the three main functionalities of the prototype. These are the Data Browser, the Rule Repository/Editor and the Request Configurator. Additionally the user has the ability to log himself out and leave the application. In the following all the main features of the three functionalities are explained in detail.

Figure 5: Prototype Start Screen

4.1.4. Rule Browser

As a rule repository is one of the main goals of this prototype the Rule Browser shows all the rules that are present in the system. The rules are written in Drools Rule Language at the moment and can directly be transferred into the knowledge base of the rule engine. In Deliverable 3.2 there is a set of examples of Drools rules given and explained. The basic syntax of a Drools rule will be described in the following. This rule is a simple example for a rule controlling data access on telephone numbers within a system. A user which has the registered position “admin” has access to all the telephone numbers within the system if requested.

```
rule "rule_name"
when
    $r: Request()
    $u : User(position == "Admin")
    $f : Data (name == "telephone_numbers")
then
    $r.grantAccess();
end
```

All the rules in the rule repository are shown in the Figure 6. Every rule can be deleted and edited with the rule editor, which is described in the next section. Additionally the rule browser does a code formatting and a basic syntax highlighting. The next component described will be the rule editor and its capabilities.

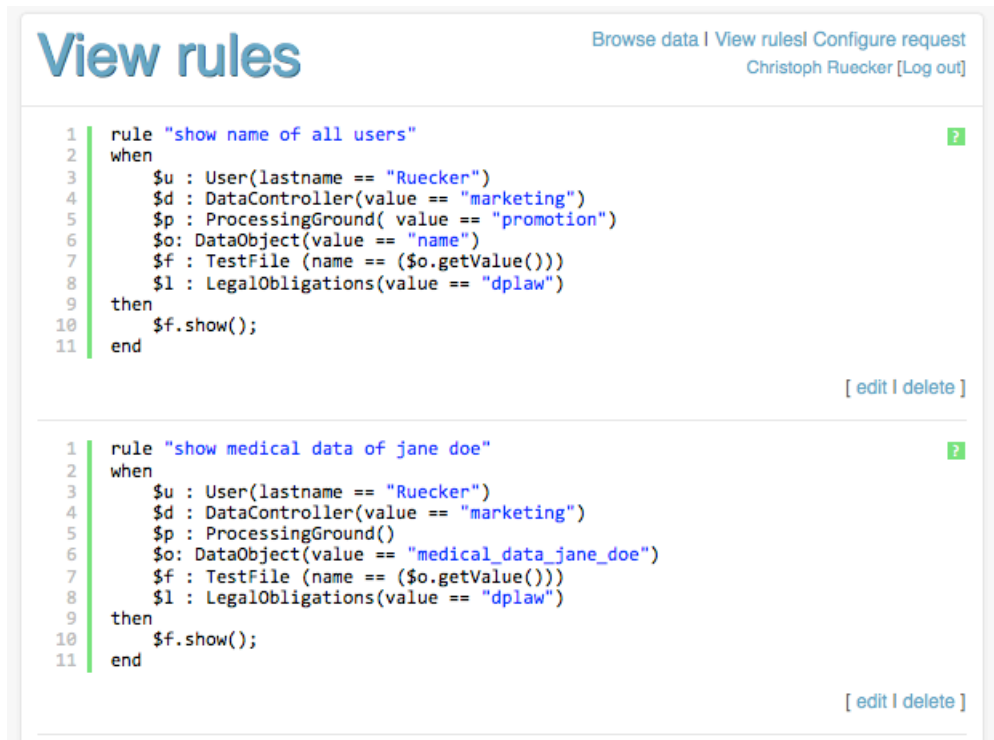


Figure 6: Prototype Rule Viewer

4.1.5. Rule Editor

The rule editor includes a small editing environment where all the supported objects are shown on the top of the page. The editor has to be distinguished from the editing environment for PRDL described in D.4.1 which will be replacing editor for Drools syntax by offering a transformation from PRDL into Drools. As this transformation is under active development this interim solution was implemented to show the rules which are present in the rule engine. To describe the editor a little further, it consists of an editable field the user can use to write valid Drools rules and use an automated syntax check with error messages returned before saving. The supported objects for the Drools rules represent the access meta model described in D.2.3. This model describes what objects the data controller has to specify for executing an request for data within the system. Every object can be used within the rule to represent certain data protection regulations and is shortly described in the following.

The *User* object represents the currently logged in person and allows to directly bind rules to the first and last name of this person. Further development will introduce more properties for the user object for example position within the company. Next, the *DataController* object enables the creation of rules that are linked to the actual data controller who wants to process. These will not be a distinct person normally but for example the member of a certain department in a company e.g. marketing. The *ProcessingGround* object was requested by the legal partners of the consortium as for them apart from a purpose there has to be a ground why processing happens. It can be used to define that ground and further enhance the rule. One of the core objects is the *DataObject*. It represents the actual data that will be affected by the rule and is essential for every rule. The *LegalObligations* object also enriches the rule by tagging it to certain policies derived from Data Protection Law. Also essential is for every rule is the *Purpose* for what the rule has to grant access to data to a data controller. The analysis of the consortium partners privacy statement showed that most of the rules will contain one or even many *Constraints*. Therefore the Constraint object can also be seen as essential for the rules. Finally the *TestFile* represents that instance of data that the rule engine handles within the demonstrator. Within the final architecture the data will be provided by the system via appropriate connectors to data repositories, e.g. databases.

Add rule [Browse data](#) [View rules](#) [Configure request](#)
Christoph Ruecker [Log out]

Supported Objects:

User <ul style="list-style-type: none"> • <i>String</i> firstname • <i>String</i> lastname DataController <ul style="list-style-type: none"> • <i>String</i> value ProcessingGround <ul style="list-style-type: none"> • <i>String</i> value DataObject <ul style="list-style-type: none"> • <i>String</i> value LegalObligations <ul style="list-style-type: none"> • <i>String</i> value 	TestFile <ul style="list-style-type: none"> • <i>String</i> name • <i>String</i> path • <i>boolean</i> isDirectory • <i>boolean</i> isFile • <i>boolean</i> isHidden • <i>long</i> size • <i>long</i> lastModified Constraint <ul style="list-style-type: none"> • <i>String</i> value Purpose <ul style="list-style-type: none"> • <i>String</i> value
---	---

Rule

ENDORSE Project
© 2011 Christoph Rücker and Thomas Lampoltshammer. All rights reserved.
© 2010 Daniel Stockhammer and Christoph Schimana, ITS-B 2008. All rights reserved.
Salzburg University of Applied Sciences

Figure 7: Prototype Rule Editor

4.1.6. Request Editor

The request editor allows the user of the prototype to specify all the objects needed execute a proper request for data within the system. It is depicted in Figure 8. For the demonstrator the specification of only single value objects is possible but this will be changed in future development, allowing data controllers for example to request a set of data for their processing purpose. Also included in future development will be the persistent storage of requests facilitating the re-usage of requests and allowing the storage of complete request sets. Currently the request editor only persists the one request that can be stored with the according button and only for the current user session.

Configure request

[Browse data](#) | [View rules](#) | [Configure request](#)
Christoph Ruecker [Log out]

DataController

Processing Ground

Data Objects

Legal Obligations

Constraint

Purpose

[Save Request](#)

ENDORSE Project
© 2011 Christoph Rücker and Thomas Lampoltshammer. All rights reserved.
© 2010 Daniel Stockhammer and Christoph Schimana, ITS-B 2008. All rights reserved.
Salzburg University of Applied Sciences

Figure 8: Prototype Request Configuration

4.1.7. Data Browser

The Data Browser is responsible for presenting the results of a request executed by the data controller and is visualized in Figure 9. It shows all the data in file format where folder icons represent the complete set of a data type e.g. all telephone numbers and the file icons illustrate a single data record. In the figure all the data included within the test environment is shown. In future development this component will also be changed towards a more convenient approach where the user can view the data the actual data. For the prototype the goal was to show the howl process from the request over the rule engine towards granted access to data. The final implementation due in October 2012 in D.4.9 will bring all the user interface features to grant a good usability.

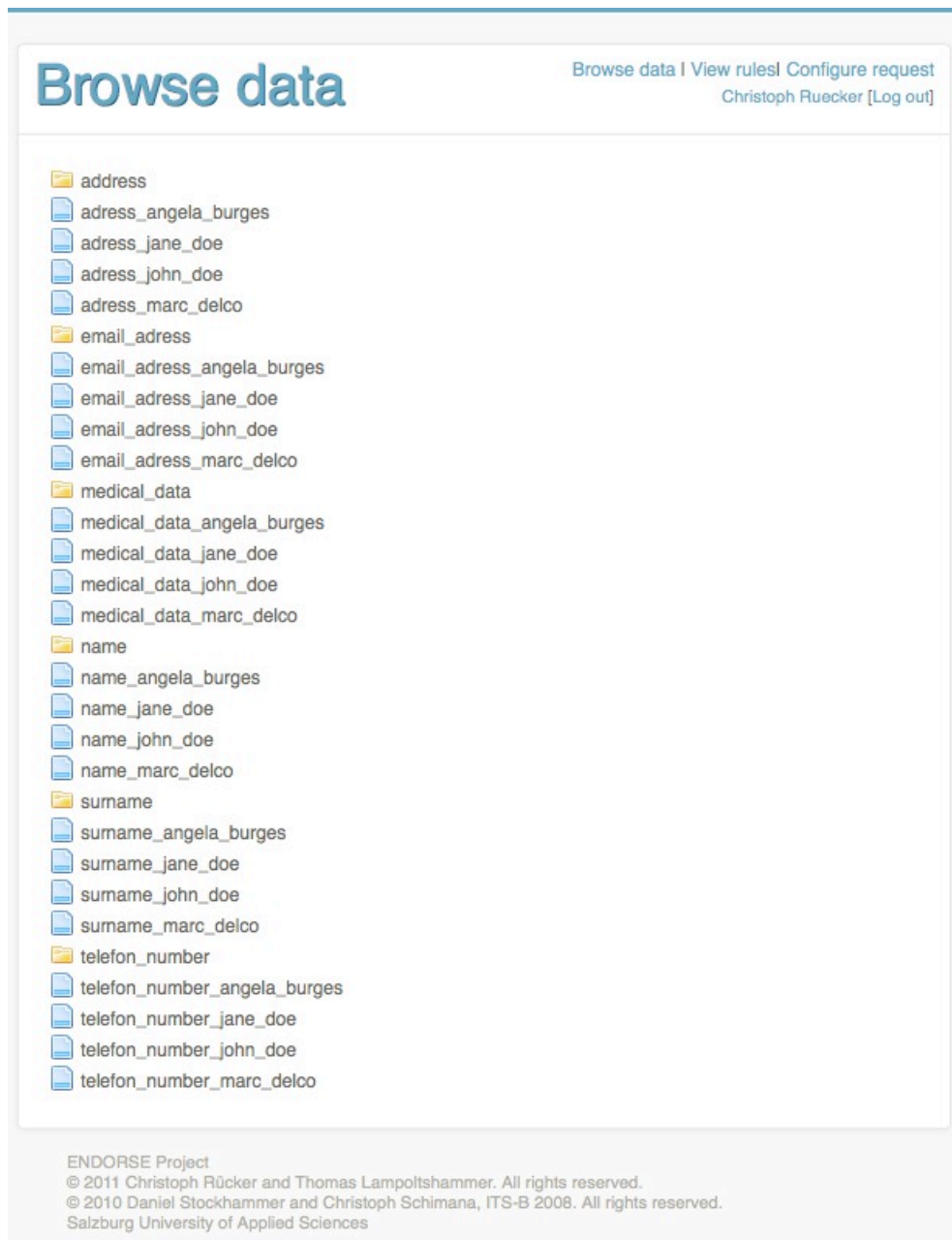


Figure 9: Prototype Data Browser

4.2. Prototype Use Case

The use case that is presented within this chapter covers normal usage scenario for the rule engine prototype. Covered shortly a data controller from EurA logs into the ENDORSE system for retrieving data. She wants to launch an online competition and needs specific data sets for that. These data sets could be name, email address and telephone numbers for example. In addition, she works at the marketing department and is aware of the fact that the customers of EurA have given consent that their data is used for marketing purposes. By using the ENDORSE system she wants to check whether she can use all the desired data for her competition and the usage is compliant on the one hand with the company policies of EurA and on the other hand with the Data Protection law.

A second employee of EurA who works at the legal department agreed on writing all the rules for making the system compliant into the rules editor so that the marketing department can execute requests for data. The following chapters shortly describe the entering of a rule, the specification of a request and the interpretation of the results after execution. They will give an overview on the capabilities of the prototype. To test the preliminary implementation of the rule engine on a configured server (see Chapter 1) it is available online under <http://tinyurl.com/5roe3th>.

4.2.1. User enters a rule

The first step for using the rule engine prototype is to specify the rules that should go into the system and therefore handle the access to data in the system. In the use case example an employee of the legal department of EurA wants to specify a rule that allows the marketing department to use the “name” data set of all customers for the usage of promoting the company. The customers have agreed to the usage of their data for marketing purposes via accepting the privacy statement of the company and additionally the data protection law is satisfied because the data is collected for a distinct purpose and the users consent is given.

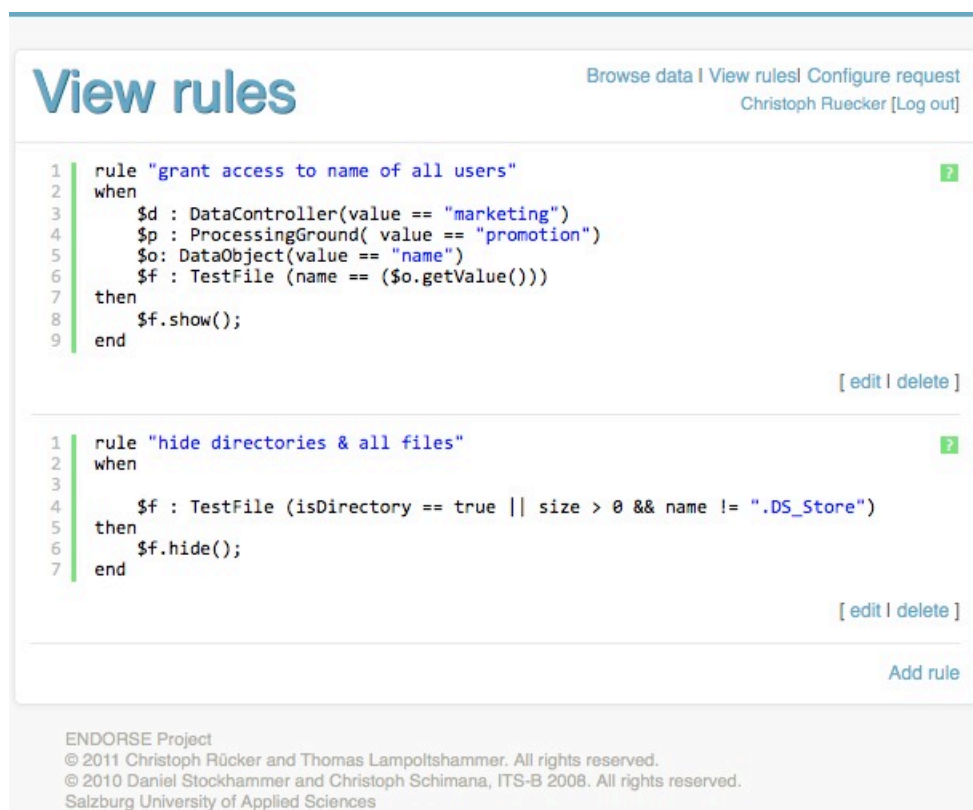


Figure 10: UseCase Rule Browser

Figure 10 shows the two rules needed to fulfil the requirement of providing the names of all the customers in the system to the requesting data controller. The first rule specifies that the data controller from marketing has the right to access the names of all customers for the processing ground purpose. The test file represents the data and is shown when the request meets the described conditions. As in case of the prototype all the access to all data sets is shown the second rule basically enforces the concealing of all data sets where access is explicitly marked granted. Now that all the rules needed are in rule repository the next section illustrates how the employee of the marketing department specifies the appropriate request.

4.2.2. User specifies a request

Figure 11 shows the request editor user interface with already filled form fields for the request describe in the previous sections. The employee only had to edit the appropriate fields and save the request with the underlying button. Currently the system only supports single request configurations but will be enhanced with a request repository where all the request are stored for further usage during further development. Once the user has finished the editing of the request she can execute it and look whether the system returns the required data or the access is prohibited. The results that the prototype returns are shortly described in the next section.

The screenshot displays the 'Configure request' web interface. At the top, there is a header with the title 'Configure request' and navigation links 'Browse data | View rules | Configure request'. Below the header, the user 'Christoph Ruecker [Log out]' is logged in. The main form area contains several labeled input fields: 'DataController' (filled with 'marketing'), 'Processing Ground' (filled with 'promotion'), 'Data Objects' (filled with 'name'), 'Legal Obligations' (empty), 'Constraint' (empty), and 'Purpose' (empty). A 'Save Request' button is located at the bottom of the form. The footer of the page contains the following text: 'ENDORSE Project', '© 2011 Christoph Rücker and Thomas Lampoltshammer. All rights reserved.', '© 2010 Daniel Stockhammer and Christoph Schimana, ITS-B 2008. All rights reserved.', and 'Salzburg University of Applied Sciences'.

Figure 11: Use Case Configure Request

4.2.3. User executes a request

When the user hits the “Browse data” link within the application the Data Browser will appear automatically as shown in Figure 12. What happens in background is the reload of the knowledge base and the resulting check whether all rules are currently persistent in the rule engine or not. When this process has finished the request is transferred into the rule engine as well and the reasoning checks whether the data sets have to be shown or not. In this deliverable the request has been approved as valid and the rule engine grants access to

all the names as shown in the Figure. Summing up, the capabilities of the application are limited at the moment but the development process from rules over request leading to successful execution is shown and will be further developed in the future of the project.

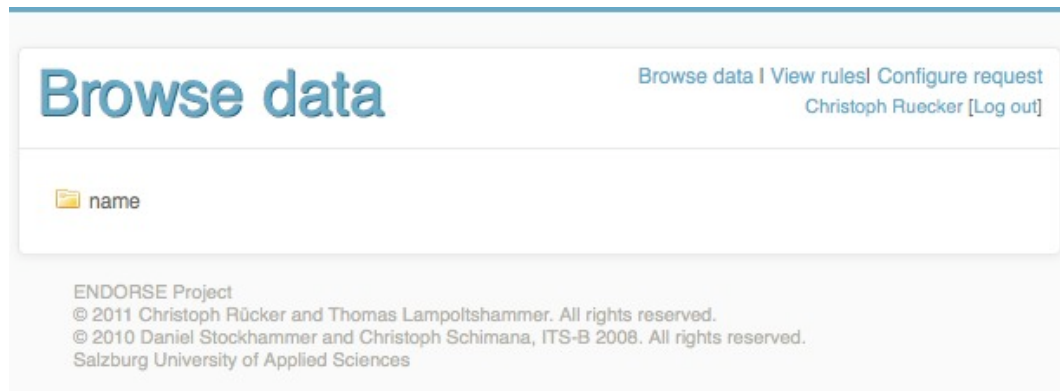


Figure 12: UseCase Data Browser

4.3. Future Development

This deliverable elaborated the first steps towards an integrated, (as a component of the ENDORSE component, or framework (as service used by the ENDORSE component) approach for the ENDORSE toolset. It shows the capabilities of the prototyped and lists challenges that have to be taken into account for the final implementation of the rule engine. The main aspects of the future development will be set on an easy usability of the software and its components. Integrating the other ENDORSE components like the editor and an external rule and probably request repository will be key. Also a lot of effort of the consortium has to be invested on the PRDL rule language and its capability of being transformable into Drools rules. Bringing together the legal people of the consortium as well as the social scientists to design a user-friendly tool that can be used without an advanced knowledge of the field is the vision behind the development started with this prototype.

5. Conclusion and Outlook

The deliverable at hand presented the first efforts on creating the rule enforcing environment needed within the ENDORSE framework. It presents the general architecture of the ENDORSE framework and explains where the integration to the future planned architecture will take place. All the main components of the application are shown and described. Features that are provided by the application are illustrated and a first concept for handling the requests for personal data is given in form of a use case description.

This use case covers the scene of a data controller within EurA, a part of Generali Insurance Group, where an employee within the marketing department wants to launch an online competition. For these competition a certain data set is needed and the employee wants to use the ENDORSE system to check whether she can use the data for the purpose of an online competition. Before this scene can happen an employee of the legal department of EurA has to enter all the rule required within the prototype framework. Within the use case description one rule is exemplified to show the process of creating a rule. Finally the marketing employee can configure her request and execute it within the system.

Concluding this deliverable imposes a first step towards an integrated ENDORSE framework for ensuring data privacy via rules derived from data protection law. The focus for the future work conducted on the rule engine lies on an architecture integrated approach which offers a user-friendly rule deployment environment. The other ENDORSE components such as the rules editor supporting PRDL and a rule repository, including a set of rule sets, will also be integrated in this final architecture.

6. Installation & Setup

This chapter is dedicated to the whole installation process regarding the needed server components as well as their configuration. As server operating system CentOS 5.5 [1] was used. Regarding the Web-server components, Glassfish 3.1.1[3] and Apache Derby 10.8.1.2.[2] were utilized. This description assumes that the Linux operation system is already up and running as well as an up-to-date Java JDK Version 1.6x+. A detailed description of the installation process of the OS can be found here [4].

6.1. Installation of the Components

First of all, the components have to be acquired. This is done by using the *wget* command as follows:

```
[user]# wget http://download.java.net/glassfish/3.1.1/release/glassfish-3.1.1-ml.zip
```

After the download has finished, move the zip File to the directory */usr/share* for example:

```
[user]# mv glassfish-3.0.1.zip /usr/share/glassfish-3.0.1.zip
```

Next change the current working directory to */usr/share* and unzip the file

```
[user]#cd /usr/share
```

```
[user]#unzip -q glassfish-3.1.1-ml.zip
```

The rules database is then copied into the Glashfish subdirectory */javadb/rules*

Before starting the Glassfish Application Server, the Derby DB has to be started. NOTICE: Both commands have to be executed inside the *glassfish/bin* directory:

```
./asadmin start-database -dbhost 0.0.0.0 -dbhome /usr/share/glassfish3/javadb/
```

The parameter *dbhost* with the value 0.0.0.0 stands for universal access from every client (possible restrictions can be configured here). The parameter *dbhome* specifies the location of the rule database. After the successful start of the database, the application server can be powered on:

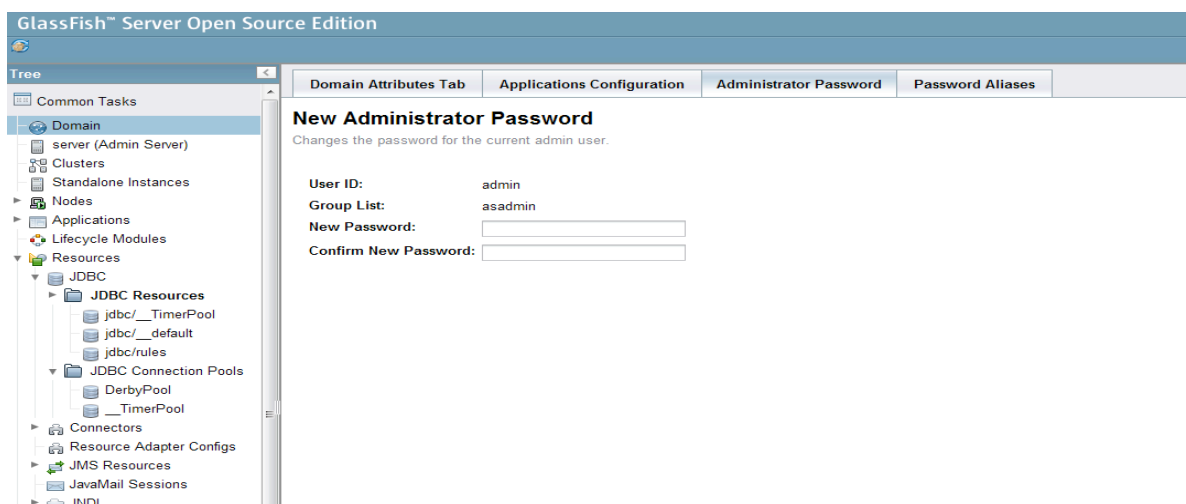
```
./asadmin start-domain domain1
```

The Web-interface of the server is now available at:

```
http://[IP of Server]:4848
```

6.2. Application Server Setup

First of all, out of security reasons, the standard admin account should be modified. By standard, both the fields for user and password are blank. This can be altered by visiting the admin registers in the server as shown below:



Tree

Common Tasks

Domain

- server (Admin Server)

Clusters

- Standalone Instances

Nodes

- Applications
- Lifecycle Modules

Resources

- JDBC
 - JDBC Resources
 - jdbc/_TimerPool
 - jdbc/_default
 - jdbc/rules
 - JDBC Connection Pools
 - DerbyPool
 - _TimerPool
- Connectors
- Resource Adapter Configs
- JMS Resources
- JavaMail Sessions
- JNDI
- Configurations
 - default-config
 - server-config
- Update Tool

Edit JDBC Connection Pool

Modify an existing JDBC connection pool. A JDBC connection pool is a group of reusable connections for a particular database.

Load DefaultsFlushPing

General Settings

Pool Name:

DerbyPool

Resource Type:

javax.sql.DataSource

Must be specified if the datasource class implements more than 1 of the interface.

Datasource Classname:

org.apache.derby.jdbc.ClientDataSource

Vendor-specific classname that implements the DataSource and/or XADataSource APIs

Driver Classname:

Vendor-specific classname that implements the java.sql.Driver interface.

Ping:

☒ Enabled

When enabled, the pool is pinged during creation or reconfiguration to identify and warn of any erroneous values for its attributes

Description:

Pool Settings

Initial and Minimum Pool Size:

8

Connections

Minimum and initial number of connections maintained in the pool

Maximum Pool Size:

32

Connections

Maximum number of connections that can be created to satisfy client requests

Pool Resize Quantity:

2

Connections

Number of connections to be removed when pool idle timeout expires

Idle Timeout:

300

Seconds

Maximum time that connection can remain idle in the pool

Max Wait Time:

60000

Milliseconds

Amount of time caller waits before connection timeout is sent

Transaction

Non Transactional Connections:

☒ Enabled

Returns non-transactional connections

Transaction Isolation:

If unspecified, use default level for JDBC Driver

Isolation Level:

☒ Guaranteed

All connections use same isolation level; requires Transaction Isolation

Page 21 of (24)

Match Connections: ☒ **Enabled**
Turns connection matching for the pool on or off

Max Connection Usage :
Connections will be reused by the pool for the specified number of times, after which they will be closed. 0 implies the feature is not enabled.

Connection Validation

Connection Validation: ☒ **Required**
Validate connections, allow server to reconnect in case of failure

Validation Method:

Table Name:
If table validation is selected, select or enter the table name.

Validation Classname:
If custom-validation is selected, specify validation classname.

On Any Failure: ☒ **Close All Connections**
Close all connections and reconnect on failure, otherwise reconnect only when used

Allow Non Component Callers: ☒ **Enabled**
Enable the pool to be used by non-component callers such as ServletFilters

GlassFish™ Server Open Source Edition

Tree

- Common Tasks
- Domain
 - server (Admin Server)
 - Clusters
 - Standalone Instances
 - Nodes
 - Applications
 - Lifecycle Modules
 - Resources
 - JDBC
 - JDBC Resources
 - jdbc/_TimerPool
 - jdbc/_default
 - jdbc/rules
 - JDBC Connection Pools
 - DerbyPool
 - _TimerPool
 - Connectors
 - Resource Adapter Configs
 - JMS Resources
 - JavaMail Sessions
 - JNDI
 - Configurations
 - default-config
 - server-config
 - Update Tool

General Advanced Additional Properties

Edit JDBC Connection Pool Properties
Modify properties of an existing JDBC connection pool.

Pool Name: DerbyPool

Additional Properties (7)

Name	Value	Description:
PortNumber	1527	
Password	app	
User	app	
serverName	localhost	
DatabaseName	rules	
URL	jdbc:derby://localhost:1527/rules	
driverClass	org.apache.derby.jdbc.ClientDriver	

Save Cancel

After that, a new JDBC resource can be created, using the recently defined JDBC Connection Pool:

GlassFish™ Server Open Source Edition

Tree

- Common Tasks
- Domain
 - server (Admin Server)
 - Clusters
 - Standalone Instances
 - Nodes
 - Applications
 - Lifecycle Modules
 - Resources
 - JDBC
 - JDBC Resources
 - jdbc/_TimerPool
 - jdbc/_default
 - jdbc/rules
 - JDBC Connection Pools
 - DerbyPool
 - _TimerPool
 - Connectors
 - Resource Adapter Configs
 - JMS Resources
 - JavaMail Sessions
 - JNDI
 - Configurations
 - default-config
 - server-config
 - Update Tool

Edit JDBC Resource
Edit an existing JDBC data source.

Load Defaults

JNDI Name: jdbc/rules

Pool Name:

Use the JDBC Connection Pools page to create new pools

Description:

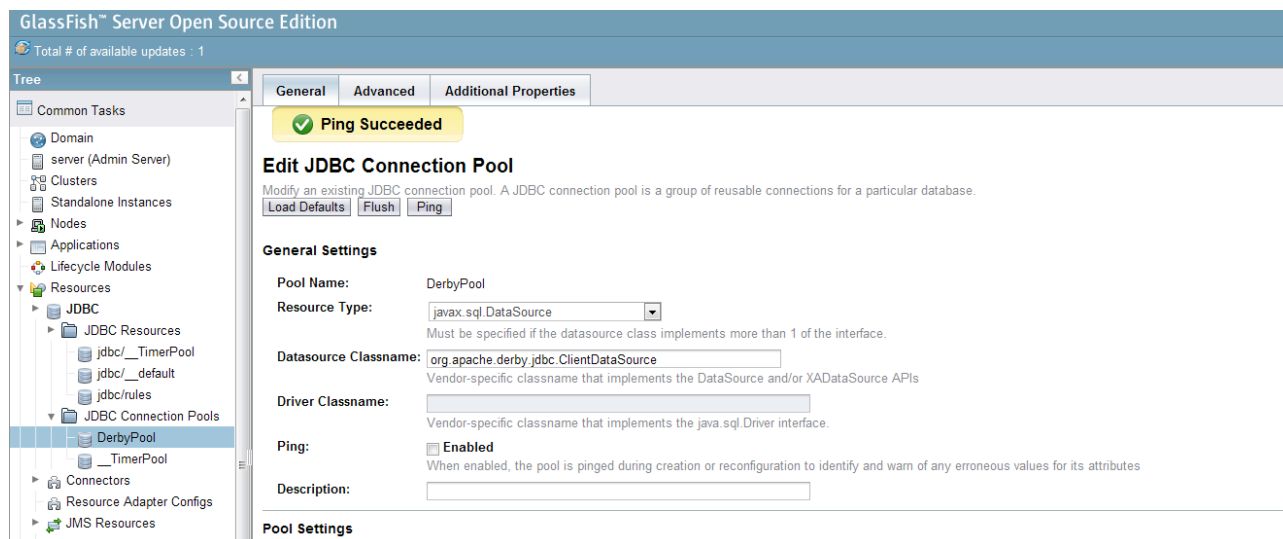
Status: ☒ **Enabled**

Additional Properties (0)

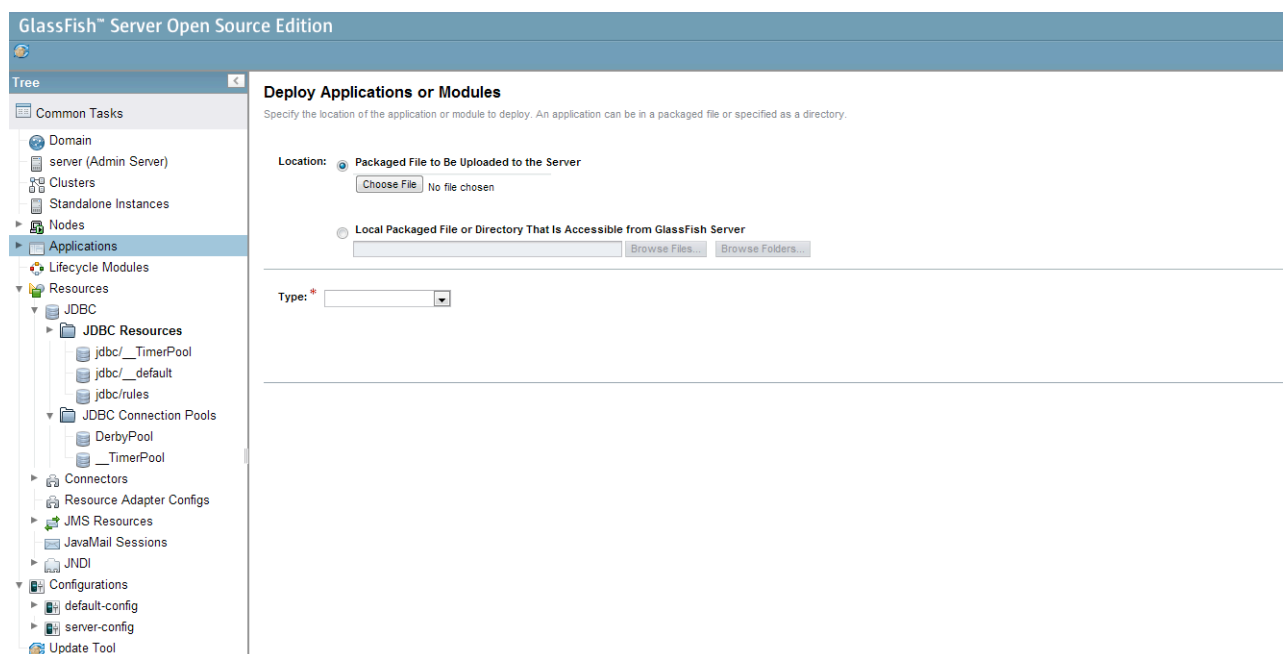
Add Property Delete Properties

Name	Value	Description:
No items found.		

Now the database source should be reachable. This can be tested by using the ping tool integrated into the Web-interface:



Finally, the application is going to be deployed on the server. As *Type* “Web Application” is being selected:



The application is now available at the server at the following address:

[http://\[IP of Server\]:8080/\[Name of Application\]-war/](http://[IP of Server]:8080/[Name of Application]-war/)

7. References

- [1] CentOS – The Community Enterprise Operating System, <http://www.centos.org/>, last accessed on 13/10/2011.
- [2] Apache Derby DB, <http://db.apache.org/derby/>, last accessed on 13/10/2011.
- [3] Glassfish Application Server, <http://glassfish.java.net/>, last accessed on 13/10/2011.
- [4] Red Hat Enterprise Linux Installation Guide, http://www.centos.org/docs/5/html/5.1/Installation_Guide/, last accessed on 13/10/2011.
- [5] Java Beans Specification, <http://download.oracle.com/otndocs/jcp/7224-javabeans-1.01-fr-spec-oth-JSpec/>, last accessed on 13/10/2011.
- [6] Java Server Pages, <http://www.oracle.com/technetwork/java/javase/jsp/index.html>, last accessed on 13/10/2011.
- [7] Drools - The Business Logic integration Platform, <http://www.jboss.org/drools>, last accessed on 13/10/2011.
- [8] Forgy, C., Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem, *Artificial Intelligence*, 19, pp 17–37, 1982.
- [9] Ferronato, P., Bordin, M., D3.1 - Functional Architecture – ENDORSE Project, 2011.