



Developer's Guide
HERAS^{AF} : PAP

Department of Computer Science
University of Applied Sciences Rapperswil (HSR)
Spring Semester 2008

Artifact of HERAS^{AF}: Policy Deployment Module [NZ08PAP]

Students: Patrick Neyer, Christof Zellweger

Examiner: Wolfgang Giersche

Coaches: René Eggenschwiler, Florian Huonder

Table of contents

TABLE OF CONTENTS.....	2
PART I: INTRODUCTION	5
1 INTENDED AUDIENCE.....	6
2 HERAS ^{AF}	6
3 EXTENSIBLE ACCESS CONTROL MARKUP LANGUAGE - XACML.....	8
4 TECHNOLOGIES.....	14
5 NAMING.....	15
PART II: CONFIGURATION	16
6 OVERVIEW	17
7 HERASAF-PAP-ICEFACES-DOMAIN	18
7.1 MAVEN CONFIGURATION	18
8 HERASAF-PAP-ICEFACES-DAO	19
8.1 SPRING CONFIGURATION.....	19
8.2 MAVEN CONFIGURATION	22
9 HERASAF-PAP-ICEFACES-SERVICE.....	23
9.1 SPRING CONFIGURATION.....	23
9.2 MAVEN CONFIGURATION	25
10 HERASAF-PAP-XACML-TRANSFORMER	27
10.1 SPRING CONFIGURATION	27
10.2 MAVEN CONFIGURATION	28
11 HERASAF-PAP-ICEFACES-WS-CLIENT	30
11.1 SPRING CONFIGURATION	30
11.2 MAVEN CONFIGURATION	31
12 HERASAF-PAP-DEPLOYMENT	33
12.1 SPRING CONFIGURATION	33
12.2 MAVEN CONFIGURATION	33
13 HERASAF-PAP-ICEFACES-WEB	35
13.1 SPRING CONFIGURATION	35
13.2 MAVEN CONFIGURATION	36
13.3 SPRING WEB FLOW CONFIGURATION	37
13.4 WEB.XML	38
13.5 FACES-CONFIG.XML.....	42
13.6 ACEGI CONFIGURATION.....	45
13.7 TAGLIB CONFIGURATION	45
14 HERASAF-PDP-WS-POLICY.....	47
14.1 SPRING CONFIGURATION	47
14.2 MAVEN CONFIGURATION	49
PART III: ARCHITECTURE AND DESIGN	51

15	PAP ARCHITECTURE	52
15.1	MODULES	52
15.2	PAP INTERACTION SCENARIO	54
16	HERASAF-PAP-ICEFACES-DOMAIN	56
16.1	INTERFACES / ABSTRACT CLASSES	56
16.1.1	ABSTRACT CLASS <code>TEMPLATE</code>	56
16.1.2	ABSTRACT CLASS <code>EXPRESSION</code>	56
16.1.3	ABSTRACT CLASS <code>MATCHATTRIBUTEDEFINITION</code>	57
16.1.4	INTERFACE <code>REPLICABLE</code>	57
16.1.5	INTERFACE <code>CLONENAMINGSTRATEGY</code>	57
16.2	IMPLEMENTATION	58
16.3	DESIGN DECISIONS	59
17	HERASAF-PAP-ICEFACES-DAO	62
17.1	INTERFACES	62
17.2	IMPLEMENTATION	67
17.3	DESIGN DECISIONS	69
18	HERASAF-PAP-ICEFACES-SERVICE	71
18.1	INTERFACES	71
18.1.1	INTERFACE <code>TEMPLATEMANAGEMENT</code>	71
18.1.2	INTERFACE <code>POLICYMANAGEMENT</code>	72
18.1.3	INTERFACE <code>PDPMANAGEMENT</code>	72
18.2	IMPLEMENTATION	73
18.3	DESIGN DECISIONS	73
19	HERASAF-PAP-ICEFACES-WEB	75
19.1	INTERFACES	75
19.1.1	ABSTRACT CLASS <code>TEMPLATEEVENTS</code>	75
19.1.2	ABSTRACT CLASS <code>SORTABLETABLE</code>	77
19.1.3	ABSTRACT CLASS <code>FACESMESSAGEHELPER</code>	78
19.1.4	INTERFACE <code>SELECTIONTABLEEVENTS</code>	79
19.2	IMPLEMENTATION	80
19.2.1	FACELETS	81
19.2.2	WEB PAGE STRUCTURE	90
19.2.3	XHTML TEMPLATES	92
19.2.4	WEB FLOWS	94
19.2.5	INTERNATIONALIZATION	106
19.3	DESIGN DECISIONS	106
20	HERASAF-PAP-XACML-TRANSFORMER	109
20.1	INTERFACES	109
20.2	IMPLEMENTATION	110
20.2.1	PACKAGE <code>ORG.HERASAF.PAP.TRANSFORMER.IMPL</code>	111
20.2.2	SEQUENCE DIAGRAMS	111
20.3	TESTING	114
20.4	DESIGN DECISIONS	115
21	HERASAF-PAP-DEPLOYMENT	117
21.1	INTERFACES	117
21.1.1	INTERFACE <code>HERASDEPLOYER</code>	117
21.2	IMPLEMENTATION	118
21.3	TESTING	119
21.4	DESIGN DECISION	119

22	HERASAF-PAP-WS-CLIENT	120
22.1	INTERFACES	120
22.1.1	INTERFACE <code>HERAS^{PDP}WSCLIENT</code>	120
22.2	IMPLEMENTATION	121
22.3	TESTING	121
22.4	DESIGN DECISIONS	122
23	HERASAF-PDP-WS-POLICY	124
23.1	INTERFACES	124
23.1.1	ABSTRACT CLASS <code>ABSTRACTSAMLHANDLER</code>	124
23.2	IMPLEMENTATION	125
23.3	TESTING	125
23.4	DESIGN DECISIONS	126
 <u>PART IV: DEPLOYMENT</u>		<u>128</u>
1	OVERVIEW	129
2	APACHE MAVEN 2	129
2.1	GLOBAL MAVEN CONFIGURATION	129
2.2	MAVEN CONFIGURATION FOR FURTHER MODULES	131
 <u>APPENDIX A: ISSUES, EXTENSIONS AND REFACTORINGS</u>		<u>133</u>
1	OVERVIEW	134
2	OPEN ISSUES	134
3	EXTENSION POINTS	135
4	REFACTORINGS	136
 <u>APPENDIX B: GENERAL</u>		<u>137</u>
5	BIBLIOGRAPHY	137
5.1	HERAS ^{AF} DOCUMENTS AND THESES	137
5.2	SPECIFICATIONS AND STANDARDS	137
5.3	WEB RESOURCES	138
5.4	FRAMEWORKS AND SOFTWARE	138

Part I: Introduction

1 Intended audience

Who is this guide for?

Generally speaking this guide is for everyone who is more interested in the technical details, how exactly we solved certain problems and who wants to read some elaborations on our decisions we took.

In addition, this guide helps any developer who intends to further extend HERAS^{AF} with additional functionality and take it to the next level.

Part 1 gives an overview of HERAS^{AF} and XACML.

Part 2 lists the configuration possibilities of HERAS^{AF} Policy Administration Point (PAP) and the web service between HERAS^{AF} PAP and HERAS^{AF} PDP

Part 3 covers the architecture and design of HERAS^{AF} PAP and HERAS^{AF} PAP-PDP web service.

Part 4 provides information about deployment of HERAS^{AF} PAP and HERAS^{AF} PAP-PDP web service

2 HERAS^{AF}

Overview

HERAS^{AF} is an open-source project with the objective to realize a central manageable authorization solution. It is built upon freely available, established and future driven technologies and standards. The main focus relies on interoperability, expandability and exchangeability of the integrated components.

In today's economic world, centralized policy management and access control is not an issue for many companies although they could save a lot of effort and money once such a system had been integrated in their environment. Most applications use proprietary access control and authorization mechanisms. This often leads to inconsistencies since various rights to access a resource have to be handled separately in all applications. Having many applications up and running, it is almost impossible for an administrator to keep track of all access rights effectively in place. This leads to vulnerable and flawed deployed security policies that can turn out to be very costly for a company, for example if this becomes public or exposed to the outside world.

HERAS^{AF} takes on the challenge to solve these problems.

Background information

After 8 months of planning, conception and information gathering, HERAS^{AF} was launched as a project by René Eggenschwiler, Yan Graf and Wolfgang Giersche in January 2006.

As a proof of concept, René Eggenschwiler [EGG06] and Yan Graf [GRAF06] laid the basis for HERAS^{AF} in mid 2006. Their implementation was based on Sun's XACML implementation [SUNXACML].

As a term thesis, Massimo Cerqui and Sandro Strebel built a Policy Enforcement Point (PEP) by using SpringAOP and AspectJ [SC07PEP]. As a follow-up and diploma thesis, a role-dependant Policy Administration Point (PAP) has been implemented with Spring, Spring Web Flow, JSF and Facelets.

At the same time, Sascha Dolski, Florian Huonder and Stefan Oberholzer created a PDP web service endpoint based on Sun's XACML implementation. Later in 2007, due to several problems with Sun's XACML, they realized a new XACML implementation for HERAS^{AF} [DOH07DA] as a diploma thesis.

Objectives

The main objectives of HERAS^{AF} are as follows [EGG06]:

Hollistic approach:

- HERAS^{AF} supports authorization in its entirety.
- All access requests to secured resources are intercepted by PEPs and redirected to a PDP or multiple PDPs, where an evaluation process is being carried out that specifies whether access is granted or denied. The PEP then is responsible for enforcing the result of this evaluation.
- A sophisticated model design makes it possible for non-tech-savvy personnel to manage policies as well.

Enterprise suitability:

- HERAS^{AF} shall be a non-intrusive framework, meaning that only minor changes have to be undertaken to integrate it into an existing environment. Already integrated authorization solutions shall be used further on without any limitations.
- HERAS^{AF} is designed explicitly for adaptability and extensibility. The usage of the Spring IoC-Container ensures exchangeability of components used by HERAS^{AF}.
- The HERAS^{AF} API can be used to integrate corporate-specific components. These components simply need to use the extension points provided by HERAS^{AF}.
- HERAS^{AF} uses established and validated standards. This way it is based on solid ground and open for extension in the future. Extensibility enhances interoperability and helps integrating HERAS^{AF} in existing or future infrastructures.
- The Policy Administration Point (PAP) integrates a layer of abstraction in such a way that an administrator in the role of the *BAdmin* using the PAP does not need to know technical details to be able to create and manage policies. So-called templates, which contain all technical details and are created by an XACML-adept administrator, enable the administrator to form business-related policies. These policies contain no or only little business data. It is the job of the security officer to fill these policies with business-related data, as he knows the business domain but has no knowledge about all technical details. This way HERAS^{AF} achieves separation of business concerns.

Application security

- Access control to resources does not reside in different applications anymore: this task can be delegated to HERAS^{AF}. HERAS^{AF} provides agents, namely PEPs, which act as interceptors and can

be integrated in existing or new applications.

HERAS^{AF} components

Adhering to the recommendation of the XACML specification, the following components have been implemented in HERAS^{AF}:

HERAS^{AF} Policy Administration Point (PAP)

HERAS^{AF} PAP provides all functionality for managing policies which then can be deployed to one or many HERAS^{AF} PDP(s). For this task to be accomplished, a web-based user interface has been developed.

The JSF- and ICEfaces-based UI offers all functionality needed in a desktop-like look-and-feel environment where the user can accomplish his tasks in an intuitive and familiar environment, similar to a desktop application.

HERAS^{AF} Policy Decision Point (PDP)

This component forms the heart of access control in HERAS^{AF} and includes all logic for deciding whether an access request received by the PEP as RequestCtx is to be granted or denied. This task includes quick locating of applicable policies by using indexing algorithms [DOH07DA, page 33], evaluating these policies against the given RequestCtx and combining the obtained results.

HERAS^{AF} Policy Enforcement Point (PEP)

HERAS^{AF}-PEP provides structures and mechanisms to build agents that perform access control by intercepting access requests to resources. The so-created RequestCtxs are sent to the PDP which forms a decision. This decision is to be enforced by the PEP.

HERAS^{AF} Policy Information Point (PIP)

If not all information can be found within a specific request received from the PEP, HERAS^{AF} PDP queries the HERAS^{AF} PIP for the required information. The HERAS^{AF} PIP offers the logic and structure to realize adapters and connectors between the authorization solution and the existing infrastructure. Therefore it is possible to include data from legacy systems into the existing decision making process by just implement the provided adapters.

No context handler

There is no separate context handler involved in HERAS^{AF} since it has been put within the HERAS^{AF} PEP and HERAS^{AF} PDP components.

3 eXtensible Access Control Markup Language - XACML

Intention

This chapter provides an overview of the XACML standard, how data circulates in a XACML-supported system and gives a short explanation on the various components of XACML.

Overview

In modern times like these where the value of mere information grows so rapidly, access control inevitably becomes vital. Not only is it important for an enterprise to possess and be able to share information but also to decide, *who* is allowed to access this information and *who* is not. XACML is an approved standard by OASIS (*Organization for the Advancement of Structured Information Standards*) [OASIS], defined in XML, for access control, policy management and authorization systems. There are two separate, yet very similar languages defined by OASIS: one for policies and the other for

access control decision requests and decision responses.

The policy language enables administrators to define the general access control requirements to the resources in an environment or system. Different data types, combining algorithms and functions allow the creation of very complex policies and rules.

The second language provided by XACML defines a request/response language. It lets you form queries which are intercepted by a PEP (Policy Enforcement Point,) and are evaluated by a PDP (Policy Decision Point). Queries in short are access requests to resources managed and protected by the XACML-ready authorization system in place. Response to the query is sent back to the PEP which then enforces the decision made by the PDP.

The following figure shows the main XACML components:

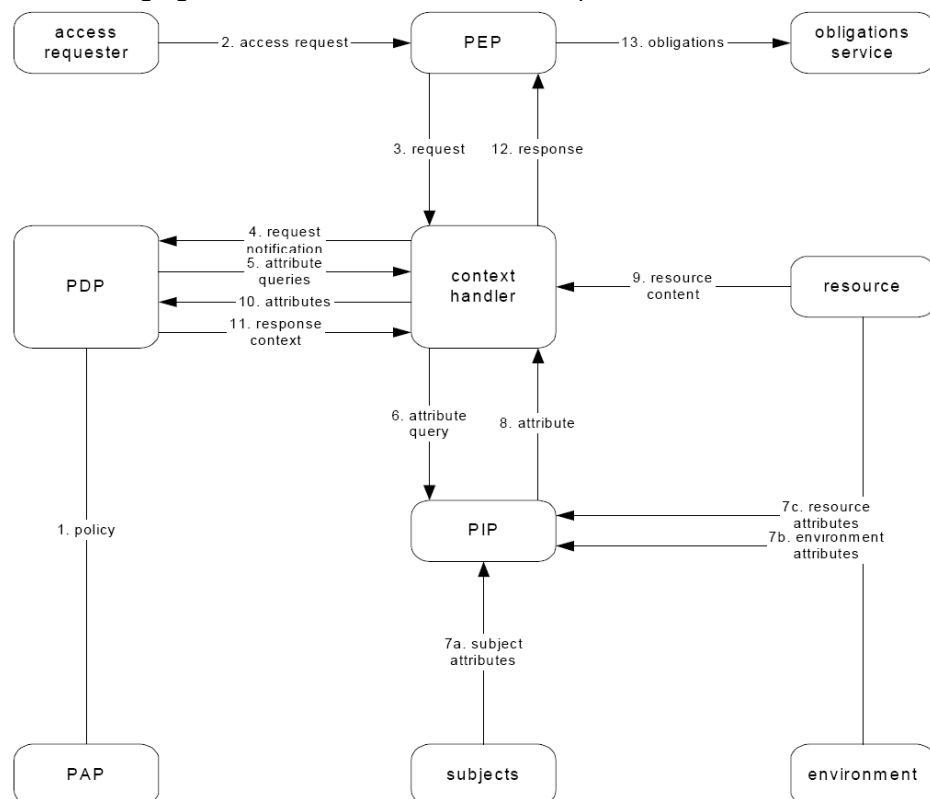


Illustration 3-1: The data-flow diagram for XACML

Data-flow

The model operates by the following steps [*XacmlSpec*]:

1. PAPs write policies and policy sets and make them available to the PDP. These policies or policy sets represent the complete policy for a specified target.
2. The access requester sends a request for access to the PEP.
3. The PEP sends the request for access to the context handler in its native request format, optionally including attributes of the subjects, resource, action and environment.
4. The context handler constructs an XACML request context and sends it to the PDP.
5. The PDP requests any additional subject, resource, action and environment attributes from the context handler.



6. The context handler requests the attributes from a PIP.
7. The PIP obtains the requested attributes.
8. The PIP returns the requested attributes to the context handler.
9. Optionally, the context handler includes the resource in the context.
10. The context handler sends the requested attributes and (optionally) the resource to the PDP. The PDP evaluates the policy.
11. The PDP returns the response context (including the authorization decision) to the context handler.
12. The context handler translates the response context to the native response format of the PEP. The context handler returns the response to the PEP.
13. The PEP fulfills the obligations.

If access is permitted, then the PEP permits access to the resource; otherwise, it denies access.

XACML Components

There are several components defined by the XACML standard that need further explanation. Note that those components do not have to be physically separated.

Policy Administration Point (PAP)

The PAP is where administrative tasks take place. New policies are created there, managed (CRUD) and deployed to the PDP.

Policy Decision Point (PDP)

A PDP is responsible for deciding, whether access to a resource is granted or if it is denied. This is done by checking policies, mostly stored on the PDP itself, for applicability to the request. The PDP gets the requests from the PEP to whom it will send back its decision-response. If there are obligations found within the applicable policy, these obligations need to be sent in the response as well.

Policy Enforcement Point (PEP)

The PEP contains all the logic necessary to interrupt access attempts to secured resources, form an authorization-request and send this request to the PDP. Depending on the outcome of the response from the PDP, the PEP enforces this decision. If there are obligations contained within the response, these obligations need to be fulfilled by the PEP before access is granted or denied respectively.

Policy Information Point (PIP)

A PIP contains additional information about resources, subjects, actions and the environment which can be referenced from within a policy. The PIP receives information-requests from the context handler.

Context Handler (CH)

There are two main tasks a context handler has to fulfill. Firstly, authorization-requests made by the PEP need to be transformed into an XACML-conform request and be sent to the PDP. Secondly, the context handler provides additional information for the PDP about a request. For the CH to obtain the additional information, it requests this information from the PIP.

XACML model

The following illustration shows the XACML 2.0 model and its various concepts.

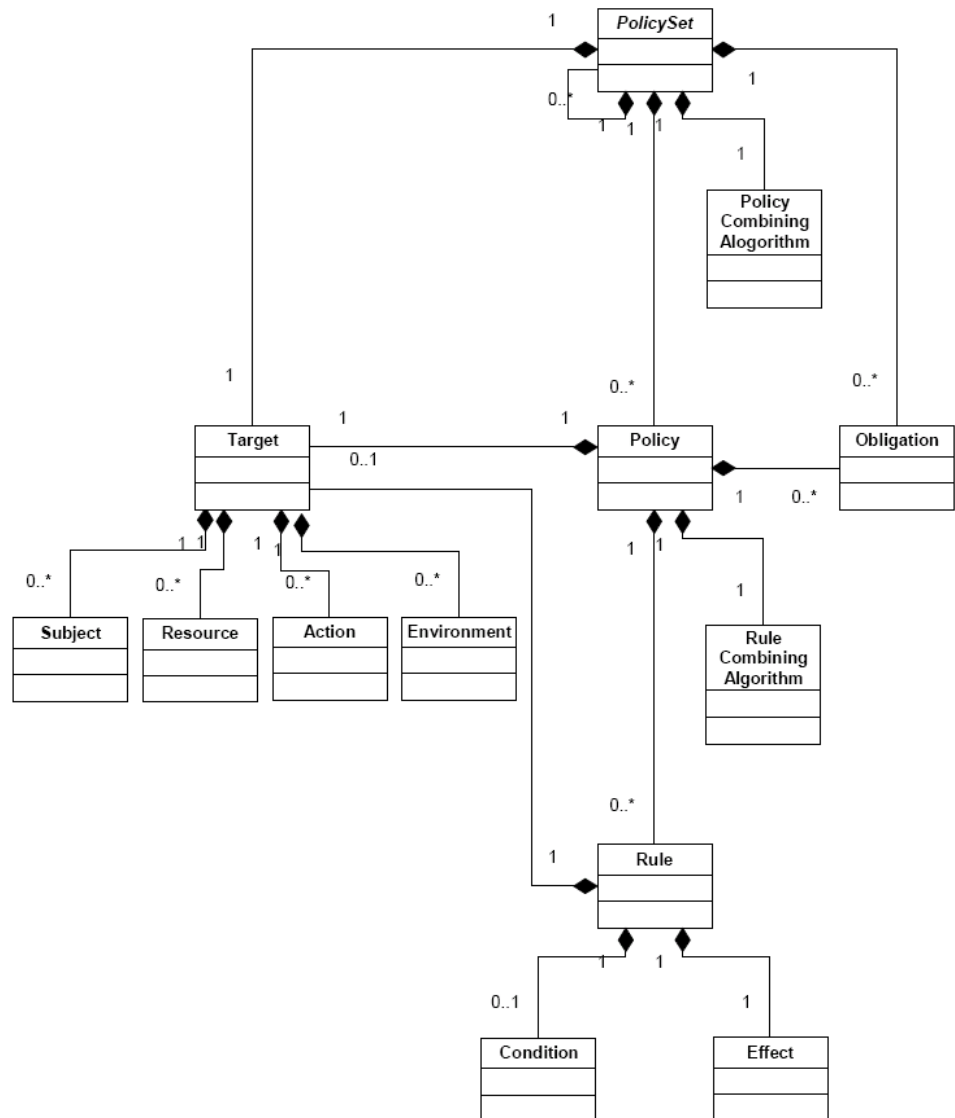


Illustration 3-2: The XACML 2.0 model.

Sample Policy

A very simple yet complete XACML 2.0 policy is given below. It allows any subject with an e-mail name in the `med.example.com` domain to perform any action on any resource.

```
<?xml version="1.0" encoding="UTF-8"?>
1) <Policy xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:oasis:names:tc:xacml:2.0:policy:schema:os
  http://docs.oasis-open.org/xacml/access_control-xacml-2.0-policy-schema-os.xsd"
  PolicyId="urn:oasis:names:tc:example:SimplePolicy1"
2) RuleCombiningAlgId="identifier:rule-combining-
  algorithm:deny-
  overrides">
  <Description>Medi Corp access control policy</Description>
  <Target/>
```



```

3) <Rule
    RuleId= "urn:oasis:names:tc:xacml:2.0:example:SimpleRule1"
    Effect="Permit">

4)   <Target>
        <Subjects>
            <Subject>
5)         <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:
                function:rfc822Name-match">
                <AttributeValue DataType="http://www.w3.org/
                    2001/XMLSchema#string">
                    med.example.com
                </AttributeValue>
6)         <SubjectAttributeDesignator AttributeId="urn:oasis:
                names:tc:xacml:1.0:subject:subject-id"
                DataType="urn:oasis:names:tc:xacml:1.0:data-
                    type:rfc822Name"/>
            </SubjectMatch>
        </Subject>
    </Subjects>
    </Target>
</Rule>
</Policy>

```

The <Policy> element contains a set of <Rule> elements and a specified procedure for combining the results of their evaluation. It is the basic unit of policy used by the PDP, and so it is intended to form the basis of an authorization decision.

Short explanation

Note the numbered lines, they mark key elements in an XACML policy [*SunXacml*]:

- 1) The *Policy*- or *PolicySet*-element is the root of every XACML policy. A *PolicySet* is a container for *Policies* or other *PolicySets*. A *Policy* element contains a set of *Rule* elements and a specified procedure for combining the results of their evaluation [*XacmlSpec*]. Only one *Policy*- or *PolicySet*-XML tag element is allowed per XACML policy.
- 2) Since a *Policy* can contain many rules with potentially different outcome in respect to the *Effect*, there has to be a mechanism to combine all rules to form one single decision. This task is accomplished by the *RuleCombiningAlgId* that specifies an algorithm for how these rules are combined. XACML 2.0 defines five default rule combining algorithms but allows for the extension with custom-made algorithms.
- 3) Rules contain the core logic of every XACML policy and consist of three elements:
 - a. *Target* (optional)
 - b. *Condition* (optional)
 - c. *Effect*

If no *Target* is specified within a *Rule*, the enclosing *Target* is used instead. If a *Rule* is applicable, its *Effect* becomes eminent: either the *Effect* can be “permit” or “deny”.
- 4) A *Target* is basically a set of simplified conditions for the *Subject*, *Resource* and *Action* that must be met for a *PolicySet*, *Policy* or *Rule* to apply to a given request. The *Target* is used by the PDP to find applicable rules to requests received by the PEP. Another very



helpful effect of the `Target`-element is that it can be used to index policies. Since a PDP must be able to find applicable policies as fast as possible, indexing them becomes of paramount importance.

- 5) A `SubjectMatch` (there are `ResourceMatch`, `ActionMatch` and `EnvironmentMatch` as well) specifies if the `Subject`-element in the actual policy matches the corresponding `Subject`-element in the request. If there are multiple `SubjectMatches` within one `Subject`, they all need to match in order for the policy to be applicable.
- 6) By using an `AttributeDesignator` it is possible to match a named attribute contained in the request with a corresponding value in the policy, for example the `subject-id`. A matching function defines how these two values have to be compared. In the example above, the `AttributeDesignator` is of type `SubjectAttributeDesignator` and references an attribute with name `urn:oasis:names:tc:xacml:1.0:subject:subject-id` in the actual request. The matching function to be used is `urn:oasis:names:tc:xacml:1.0:function:rfc822Name-match` and is defined within the `SubjectMatch`-element.

Advantages and Disadvantages of XACML

There are many advantages and some disadvantages when using XACML [*XacmlSpec*].

Advantages

XACML is a standard

By using a standard language, you are using something that has been reviewed by a large community of experts and users, you don't need to roll your own system each time, and you don't need to think about all the tricky issues involved in designing a new language. Additionally as XACML becomes more widely deployed, it will be easier to interoperate with other applications using the same standard language.

XACML is generic

Rather than trying to provide access control for a particular environment or a specific kind of resource, XACML can be used in any environment. One policy can be written and afterwards be used by many different kinds of applications. If only one common language is used, policy management becomes much easier.

XACML is distributed

A policy can be written which in turn refers to other policies kept in arbitrary locations. The result is that rather than having to manage a single monolithic policy, different people or groups can manage separate sub-policies as appropriate, and XACML knows how to correctly combine the results from these different policies into one decision.

XACML is powerful

While there are many ways the base language can be extended, many environments will not need to do so. The standard language already supports a wide variety of data types, functions, and rules about combining the results of different policies. In addition to this, there are already standard groups working on extensions and profiles that will hook XACML into other standards like SAML and LDAP, which will increase the number of ways XACML can be used.

Disadvantages

As mentioned in [*Egg06*], one major drawback in using such a flexible language is the large amount of metadata that needs to be processed in order to obtain

the actual data.

HERAS^{AF}'s own XACML implementation

Due to several bugs and flaws in Sun's XACML implementation [*SunXacml*], [*DOH07DA*] implemented their own version of the XACML specification for usage within HERAS^{AF}. Since HERAS^{AF} follows the strict credo on being a future-driven solution for holistic authorization concerns, putting it on a flawed basis regarding XACML was no option. Furthermore, XACML represents a critical component in HERAS^{AF}, so being on the wrong track here would compromise the whole project.

4 Technologies

Overview

This chapter contains a short overview of all major technologies used in this bachelor thesis. The descriptions of these technologies are all original extracts from the providers.

Acegi Security

Acegi Security is a powerful, flexible security solution for enterprise software, with a particular emphasis on applications that use Spring. Using Acegi Security provides your applications with comprehensive authentication, authorization, instance-based access control, channel security and human user detection capabilities. [*Acegi*]

Maven 2.0

Maven is a software project management and comprehension tool. Based on the concept of a project object model (POM), Maven can manage a project's build, reporting and documentation from a central piece of information. [*Maven*]

TestNG 5.7

TestNG is a testing framework for the Java programming language inspired by JUnit and NUnit. It introduces new functionality that makes it more powerful and easier in usage.

TestNG is designed to cover all categories of tests, including unit, functional, and integration tests. [*TestNG*]

Spring 2.5

The Spring Framework (or Spring for short) is an open source application framework for the Java platform.

Although the Spring Framework does not enforce any specific programming model, it has become popular in the Java community as an alternative, replacement, or even addition to the Enterprise JavaBean model.

The Spring Framework can be considered as a collection of smaller frameworks. [*Spring*]

Spring Web Flow

Spring Web Flow is a next generation Java web application controller framework that allows developers to model user actions as high-level modules called flows that are runnable in any environment.

The framework delivers improved productivity and testability while providing a strong solution to enforcing navigation rules and managing application state. [*SpringWebFlow*]

ICEfaces 1.6

ICEfaces is an integrated Ajax application framework that enables Java EE

application developers to easily create and deploy thin-client rich Internet applications (RIA) in pure Java. ICEfaces is a fully featured product that enterprise developers can use to develop new or existing Java EE applications at no cost. [Icefaces]

JavaServerFaces

JavaServer Faces (JSF) is a Java-based Web application framework intended to simplify development of user interfaces for Java EE applications. Unlike other traditional request-driven MVC web frameworks, JSF uses a component-based approach. The state of user interface components is saved when the client requests a new page and then is restored when the request is returned. Out of the box, JSF uses JavaServer Pages (JSP) for its display technology, but JSF can also accommodate other display technologies. [wikiJSF]

Facelets

Facelets is a templating language built from the ground up with the JSF component life cycle in mind. Facelets enable a developer to produce templates that build a component tree. This allows for greater reuse because composing components out of a composition of other components is possible.

5 Naming

Naming

This paragraph is important for understanding concepts we use and introduce in this thesis so please read carefully.

Technical administrator and business administrator

There are two role models defined to perform different tasks at different levels of abstraction. Since these terms are used a lot, especially also because a new role model is being introduced, we use *TAdmin* as an abbreviation for technical administrator and *BAdmin* for business administrator.

In short:

- Technical administrator: *TAdmin*
- Business administrator: *BAdmin*

xxxTemplate and xxxsTemplate

Since there are elements given in the [XACML] standard that are called nearly identically, we tried to simplify distinction by underlining the difference whenever their name is used in this guide.

It is distinguished between *ActionTemplate* and *ActionssTemplate*, *EnvironmentTemplate* and *EnvironmentssTemplate*, *ResourceTemplate* and *ResourcessTemplate* and between *SubjectTemplate* and *SubjectssTemplate*.

Part II: Configuration

6 Overview

<i>Introduction</i>	<p>The modules of the HERAS^{AF} PAP implementation are configurable through the Spring Framework application context. Configurability is a very important issue because it has various extension and adaption points. It opens a wide extensibility to different implementations without changing a single line of code. This chapter explains what can be configured and how it is done.</p>
<i>Configuration Topics</i>	<p>The following listing shows the configuration topics of the HERAS^{AF} PAP implementation.</p> <p>Spring configuration – Module and Components wiring The modules and the components are wired through the Spring Framework. This allows customizing the HERAS^{AF} PAP implementation with different components. For a detailed future list visit [Spring].</p> <p>Maven configuration In order to handle dependencies between different projects, Maven2 was used. It simplifies the build process because it provides a uniform build system by using project object model files (POM). Additionally it separates test code from main code, thus not deploying test code to the end consumer. For further features of maven, visit [Maven].</p> <p>Note: Because every pom file of every project has become larger and larger, only visual dependency trees are shown in following maven chapters. These trees are generated by an eclipse plugin [Q4Eclipse].</p> <p>Web project configuration The description of the configuration of the web project module is divided into its configuration files. Configurations of Spring Web Flow, web.xml, faces-config.xml, Acegi security and tag libraries are described</p>
<i>Configuration files</i>	<p>The configuration of the HERAS^{AF} PAP implementation is divided into several significant configuration files. Each configurable part of the implementation is described in the following chapters.</p>

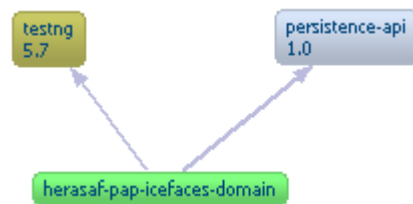
7 herasaf-pap-icefaces-domain

7.1 Maven configuration

Overview

This chapter covers the dependencies between the domain project and other projects

pom.xml visualized



Legend

persistence-api	There exists a dependency on javax.persistence due to usage of persistence-annotations in the domain model.
testng	Testng is used to test the domain model

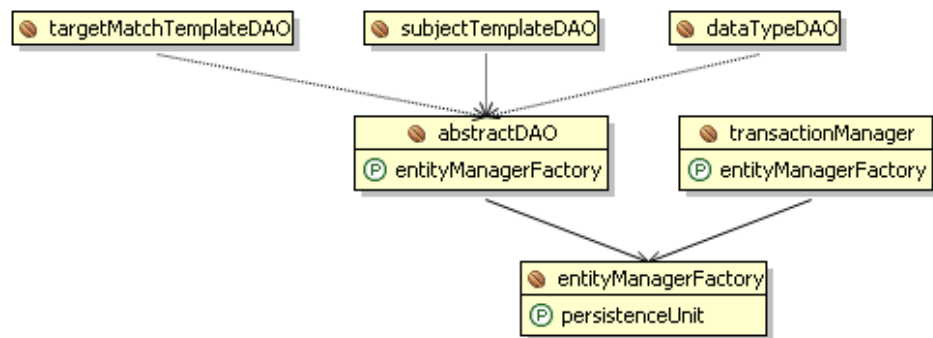
8 herasaf-pap-icefaces-dao

8.1 Spring configuration

Overview

This chapter explains how the current `herasaf-pap-icefaces-dao` module is configured

alldaos.ctx.xml
configuration
Spring graph



There are 19 `xxxDAO` beans totally. Only 3 are displayed above for overview reasons.

alldaos.ctx.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:tx="http://www.springframework.org/schema/tx"

  xsi:schemaLocation="http://www.springframework.org/schema/beans
  http://www.springframework.org/schema/beans/spring-beans-2.0.xsd
  http://www.springframework.org/schema/tx
  http://www.springframework.org/schema/tx/spring-tx.xsd">

  <tx:annotation-driven transaction-
  manager="transactionManager" />

  <bean id="transactionManager"

  class="org.springframework.orm.jpa.JpaTransactionManager">
    <property name="entityManagerFactory"
      ref="entityManagerFactory" />
  </bean>

  <bean id="entityManagerFactory"

  class="org.herasaf.pap.jpa.util.EntityManagerFactoryFactory
  Bean">
    <property name="persistenceUnit">
      <value>heras-pap</value>
    </property>
  </bean>

  <bean id="abstractDAO" abstract="true">
    <property name="entityManagerFactory">

```

1

2

3



```

        <ref bean="entityManagerFactory" />
    </property>
</bean>

<bean id="targetMatchTemplateDAO" parent="abstractDAO"
class="org.herasaf.pap.dao.jpa.JpaTargetMatchTemplateDAO">
</bean>

<!-- All DAOs with their implementation follow here -->
</beans>

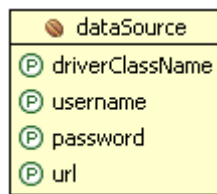
```

4

Legend

- | | |
|---|---|
| 1 | Enable the configuration of transactional behavior based on annotations. |
| 2 | Set the custom EntityManagerFactory in the Spring JpaTransactionManager |
| 3 | Initialize the custom HERAS ^{AF} PAP EntityManagerFactory with the HERAS ^{AF} PAP persistence unit. |
| 4 | The entityManagerFactory is set in every Jpa-implementation of every DAO. |

datasource.xml
configuration
Spring graph



datasource.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans-2.0.xsd">

    <bean id="dataSource"

class="org.springframework.jdbc.datasource.DriverManagerDat
aSource">
        <property name="driverClassName"
value="org.postgresql.Driver"></property>
        <property name="username" value="root"/>
        <property name="password" value="root"/>
        <property name="url"
value="jdbc:postgresql://localhost:5432/HERAS_PAP"/>
    </bean>
</beans>

```

1

Legend

-
- 1 Configures a plain old JDBC Driver via bean properties.

/META-
INF/Persistence
.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<persistence>
  <persistence-unit name="heras-pap"
    transaction-type="RESOURCE_LOCAL">
    <properties>
      <property name="hibernate.connection.driver_class"
        value="org.postgresql.Driver" />
      <property name="hibernate.connection.username"
        value="root" />
      <property name="hibernate.connection.password"
        value="root" />
      <property name="hibernate.connection.url"
        value="jdbc:postgresql://localhost:5432/HERAS_PAP"
      />
      <property name="hibernate.cache.provider_class"
        value="org.hibernate.cache.HashtableCacheProvider"
      />
      <property name="hibernate.dialect"
        value="org.hibernate.dialect.PostgreSQLDialect" />
      <property name="hibernate.show_sql" value="true" />
    </properties>

    <class>org.herasaf.pap.domain.ActionTemplate</class>
    <!-- ... -->
    <!-- All classes that are persisted follow here -->
    <!-- ... -->
  </persistence-unit>
</persistence>
  
```



Legend

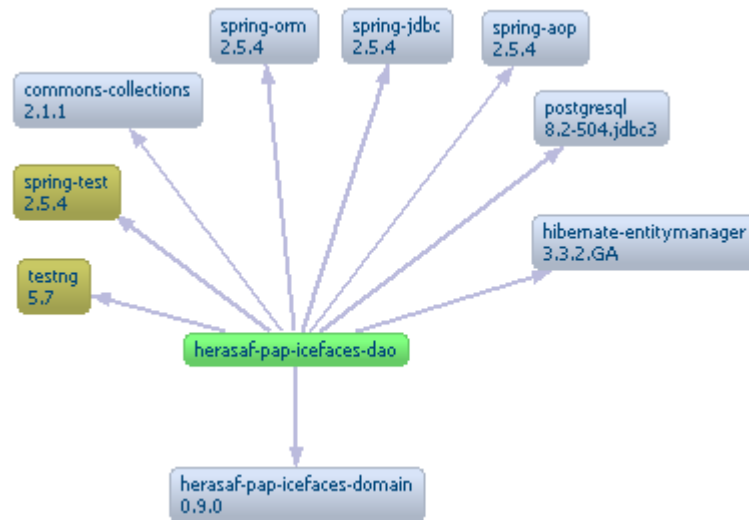
-
- 1 Defines the persistence-unit HERAS^{AF}-PAP
-
- 2 A Hibernate implementation of JPA is used [Hibernate]. The hibernate connection pooling is configured.
- Note: The hibernate connection pooling is not intended for use in a production system as it is very rudimentary.
-
- 3 Listing of the entity classes
-
- 4 The entityManagerFactory is set in every Jpa-implementation of the every DAO.

8.2 Maven configuration

Overview

This chapter covers the dependencies between the `herasaf-pap-icefaces-dao` and other modules.

pom.xml visualized



Legend

<code>herasaf-pap-icefaces-domain</code>	The domain is needed for having references to templates. They need to be known so they can be saved.
<code>commons-collections</code>	Is needed by the Spring framework.
<code>spring-orm</code>	The JPA integration layer is used from this Spring library.
<code>postgresql</code>	Integrates the postgres database driver, needed to connect to the database.
<code>spring-jdbc</code>	The Spring implementation of the standard JDBC DataSource interface is used.
<code>spring-aop</code>	Is needed for Spring transactional proxy.
<code>postgresql</code>	This library refers to the database management system driver used.
<code>hibernate-entitymanager</code>	Hibernate EntityManager implements the interfaces and lifecycle rules as defined by the EJB3 persistence specification.
<code>testng</code>	Testng is used to test the doa module
<code>spring-test</code>	Spring testing is used to simulate transactions, so no transactions are really committed to the database.

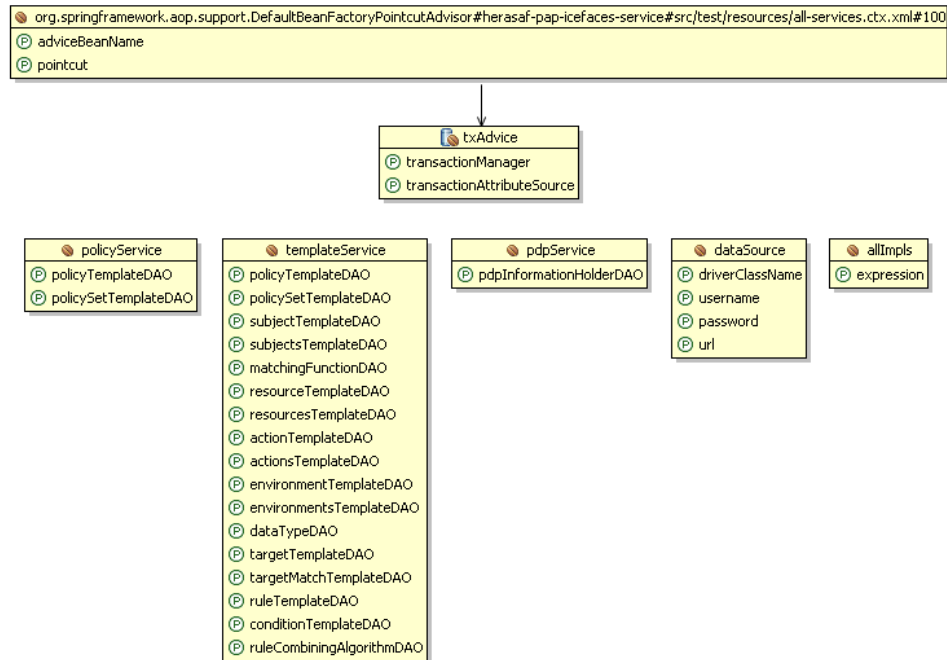
9 herasaf-pap-icefaces-service

9.1 Spring configuration

Overview

This chapter explains how the service module is configured

*all-
services.ctx.xml
Spring graph*



*all-
services.ctx.xml*

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:aop="http://www.springframework.org/schema/aop"
  xmlns:tx="http://www.springframework.org/schema/tx"
  xsi:schemaLocation="
    http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans-
    2.5.xsd
    http://www.springframework.org/schema/tx
    http://www.springframework.org/schema/tx/spring-tx-2.5.xsd
    http://www.springframework.org/schema/aop
    http://www.springframework.org/schema/aop/spring-aop-
    2.5.xsd">

  <import resource="all-daos.ctx.xml" />

  <bean id="policyService"

    class="org.herasaf.pap.service.templates.impl.PolicyManagem
    entImpl">
    <property name="policyTemplateDAO">
      <ref bean="policyTemplateDAO" />
    </property>
    <property name="policySetTemplateDAO">
      <ref bean="policySetTemplateDAO" />
    </property>
  </bean>

```



```

    </property>
  </bean>

  <bean id="templateService"

class="org.herasaf.pap.service.templates.impl.TemplateManag
ementImpl"
  <property name="policyTemplateDAO">
    <ref bean="policyTemplateDAO" />
  </property>
  <!-- Insert here DAOs -->
  <property name="ruleCombiningAlgorithmDAO">
    <ref bean="ruleCombiningAlgorithmDAO" />
  </property>
</bean>

<bean id="pdpService"

class="org.herasaf.pap.service.templates.impl.PdpManagement
Impl">
  <property name="pdpInformationHolderDAO">
    <ref bean="pdpInformationHolderDAO" />
  </property>
</bean>

<tx:advice id="txAdvice" transaction-
manager="transactionManager">
  <tx:attributes>
    <tx:method name="retrieve*" propagation="SUPPORTS" />
    <tx:method name="*" propagation="REQUIRED" />
  </tx:attributes>
</tx:advice>

<aop:config>
  <aop:pointcut id="allImpls"
    expression="execution(*
org.herasaf.pap.service.templates.impl.*Impl.*(..))" />
  <aop:advisor advice-ref="txAdvice" pointcut-
ref="allImpls" />
</aop:config>

<!-- The DataSource -->
<bean id="dataSource"

class="org.springframework.jdbc.datasource.DriverManagerDat
aSource">
  <property name="driverClassName"
    value="org.postgresql.Driver">
  </property>
  <property name="username" value="root" />
  <property name="password" value="root" />
  <property name="url"
    value="jdbc:postgresql://localhost:5432/HERAS_PAP" />
</bean>

</beans>

```

2

3

4

5

6

Legend

-
- 1 Definition of the `policyService` and its wiring

 - 2 Definition of the `templateService` and its wiring

 - 3 Definition of the `pdpService` and its wiring

 - 4 The transactional advice with its transactional semantics is set. All methods starting with 'retrieve' have the transactional propagation behavior *SUPPORTS*, and all other methods are to execute with the propagation *REQUIRED* behavior.

The *transaction-manager* attribute of the `<tx:advice/>` tag is set to the transaction manager defined in `all-daos.ctx.xml`, that is going to actually drive the transactions (in this case the *transactionManager* bean).

-
- 5 The `<aop:config/>` definition ensures that the transactional advice defined by the `txAdvice` bean (in this case our transaction manager) actually executes at the appropriate points in the program.

First we define a pointcut that matches the execution of any operation defined in any class ending with *Impl* and is located in `org.herasaf.pap.service.templates.impl`. This reflects the classes `TemplateManagementImpl`, `PolicyManagementImpl` and `PdpManagementImpl`.

Then we associate the pointcut with the `txAdvice` using an advisor. The result indicates that at the execution of any method in one of the tree "impl-classes", the advice defined by `txAdvice` will be run.

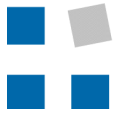
-
- 6 The `DataSource` for database access

all-daos.ctx.xml Refer to 8.1 (page 19, herasaf-pap-icefaces-dao - Spring configuration) for further information.

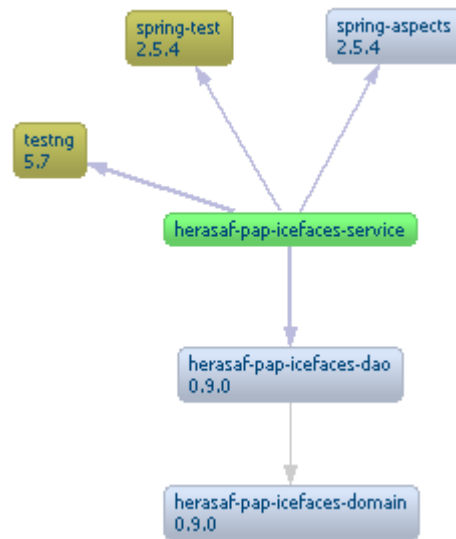
META-INF/Persistence.xml Refer to 8.1 (page 19ff, herasaf-pap-icefaces-dao - Spring configuration) for further information

9.2 Maven configuration

Overview This chapter covers the dependencies between the service project and other projects.



*pom.xml
visualized*



Legend

herasaf-pap-icefaces-dao	The service layer is in need of the DAO layer.
spring-aspects	Spring Aspects are used for configuration of the transactional behavior.
testng	Testng is used to test the service module.
spring-test	Spring testing is used.

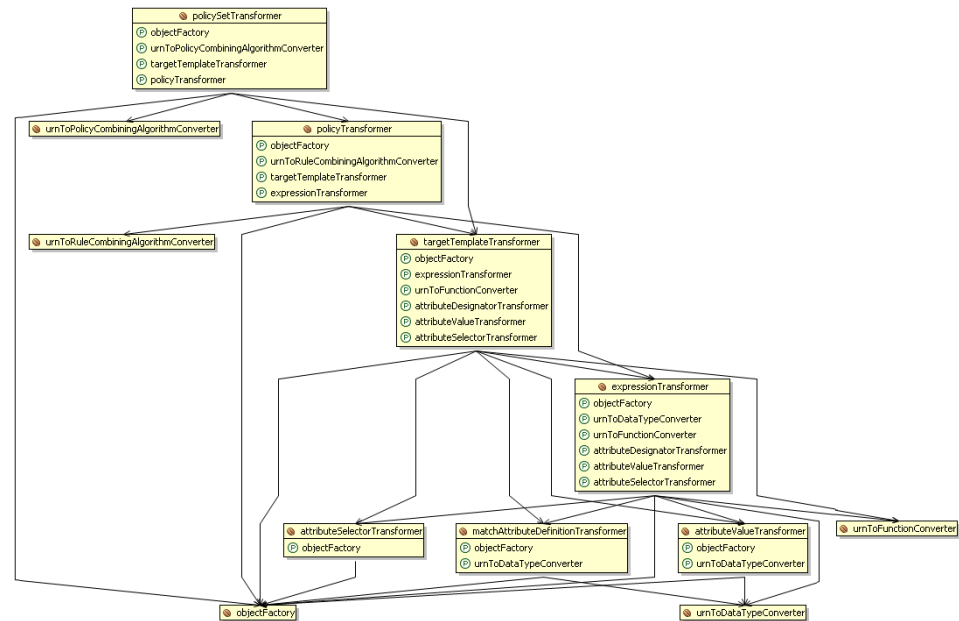
10 herasaf-pap-xacml-transformer

10.1 Spring configuration

Overview

This chapter explains how the transformer module is configured

AppContext.ctx. xml Spring graph



AppContext.ctx. xml partially

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans.xsd">

  <import resource="CombiningAlgorithms.xml" />
  <import resource="DataTypeAttributes.xml" />
  <import resource="Functions.xml" />
  <import resource="ContextAndPolicyConfiguration.xml" />

  <bean id="objectFactory"
    class="org.herasaf.xacml.core.policy.impl.ObjectFactory">
  </bean>

  <!-- Further configuration inserted here -->

  <bean id="urnToPolicyCombiningAlgorithmConverter"
    class="org.herasaf.xacml.core.converter.URNToPolicyCombiningAlgorithmConverter">
  </bean>
```

1

2

```

<bean id="policySetTransformer"

class="org.herasaf.pap.transformer.impl.PolicySetTransforme
rImpl">
  <property name="objectFactory" ref="objectFactory" />
  <property name="urnToPolicyCombiningAlgorithmConverter"
ref="urnToPolicyCombiningAlgorithmConverter" />
  <property name="targetTemplateTransformer"
ref="targetTemplateTransformer" />
  <property name="policyTransformer"
ref="policyTransformer" />
</bean>

<!-- Further configuration inserted here -->

<bean id="attributeSelectorTransformer"

class="org.herasaf.pap.transformer.impl.AttributeSelectorTr
ansformerImpl">
  <property name="objectFactory" ref="objectFactory" />
</bean>
</beans>

```

Legend

- | | |
|---|---|
| 1 | Imports of further resources, explained later |
| 2 | Spring IoC is used to inject all transformer with necessary data, like references to other transformers or the resulting XACML classes a transformed <code>PolicyTemplate</code> or <code>PolicySetTempalte</code> will be type of (namely <code>Evaluatables</code>). |

*CombiningAlgor
ithms.xml*

Initializes HERAS^{AF} XACML *URNToPolicyCombiningAlgorithmConverter* and *PolicyCombiningAlgorithms*. See developer guide of HERAS^{AF} XACML [DOH07DADev] for further information.

*ContextAndPoli
cyConfiguration.
xml*

Configures the policy marshalling and unmarshalling. Refer to HERAS^{AF} XACML developer guide [DOH07DADev] for detailed information.

*Data TypeAttribu
tes.xml*

Initializes HERAS^{AF} XACML *URNToDataTypeConverter*. See developer guide of HERAS^{AF} XACML [DOH07DADev] for further information.

Functions.xml

Initializes HERAS^{AF} XACML *URNToFunctionConverter*. See developer guide of HERAS^{AF} XACML [DOH07DADev] for further information.

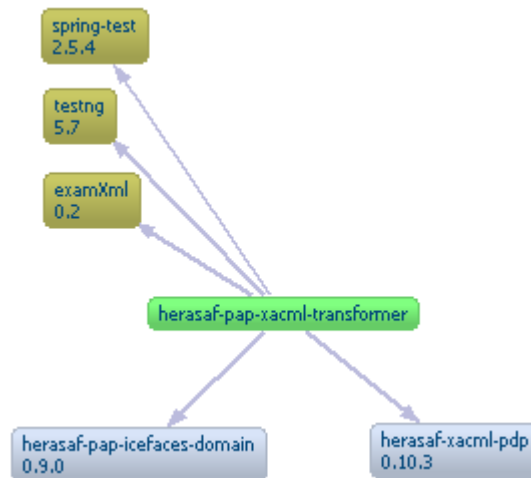
10.2 Maven configuration

Overview

This chapter covers the dependencies between the transformer project and other projects.



pom.xml
visualized



Legend

herasaf-pap-icefaces-domain	The transformer is in need of the domain objects.
herasaf-xacml-pdp	The transformer transforms <i>Policy</i> - and <i>PolicySetTemplates</i> to <i>Evaluatables</i> that are XACML classes.
examXml	ExamlXml is used for comparing two xml files for differences. Refer to 20.3 (page 114, herasaf-pap-xacml-transformer - Testing) for information why examXML is used.
testng	Testng is used to test the transformation.
spring-test	Spring testing is used.

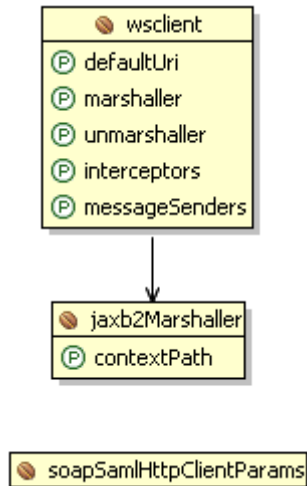
11 herasaf-pap-icefaces-ws-client

11.1 Spring configuration

Overview

This chapter explains how the web service client module is configured

ws-client.xml Spring graph



ws-client.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

  xsi:schemaLocation="http://www.springframework.org/schema/b
eans http://www.springframework.org/schema/beans/spring-
beans-2.5.xsd">

  <bean id="wsclient"
class="org.herasaf.pap.ws.client.impl.PapWsClient">
  <property name="defaultUri"
    value="http://localhost:14480/herasaf-pdp-ws-
policy/pdpPolicyService" />
  <property name="marshaller" ref="jaxb2Marshaller" />
  <property name="unmarshaller" ref="jaxb2Marshaller" />
  <property name="interceptors">
    <bean
class="org.herasaf.saml.springws.SoapActionClientIntercepto
r" />
  </property>
  <property name="messageSenders">
    <bean
class="org.springframework.ws.transport.http.CommonsHttpMes
sageSender">
      <property name="httpClient">
        <bean
class="org.apache.commons.httpclient.HttpClient">

```

1

2

3



```

        <property name="params"
            ref="soapSamlHttpClientParams" />
    </bean>
</property>

</bean>
</property>
</bean>

<bean id="soapSamlHttpClientParams"

class="org.herasaf.saml.springws.SoapSamlHttpClientParams"
/>

<bean id="jaxb2Marshaller"

class="org.herasaf.saml.marshaller.Jaxb2NamespaceMarshaller"
">
    <property name="contextPath"

value="org.herasaf.xacml.core.context.impl:org.herasaf.xacm
l.core.policy.impl:org.herasaf.saml.jaxb.saml.assertion:org
.herasaf.saml.jaxb.saml.protocol:org.herasaf.saml.jaxb.w3.x
mldsig:org.herasaf.saml.jaxb.w3.xmlenc:org.herasaf.saml.jax
b.xacml.assertion:org.herasaf.saml.jaxb.xacml.protocol" />
    </bean>
</beans>

```

4

Legend

- | | |
|---|---|
| 1 | Defining the HERAS ^{AF} PAP ws client. As the ws client implementation extends Spring <i>WebServiceGatewaySupport</i> , we can define a defaultURI, a marshaller and an unmarshaller. |
| 2 | Definition of <i>SoapActionClientInterceptor</i> . This HERAS ^{AF} SAML interceptor is needed to add the SAML-specific <i>SOAPAction</i> header field to the HTTP request. |
| 3 | Define the messageSenders to be used by the gateway. A Spring <i>CommonsHttpMessageSender</i> is used that is based on a apache <i>HttpClient</i> , which HERAS ^{AF} SAML parameters are added (<i>SoapSamlHttpClientParams</i>) |
| 4 | Definition of the (un-)marshaller, a custom HERAS ^{AF} JAXB marshaller that allows mapping a prefix of a namespace with a defined name. The JAXB [JAXB] context path is set too. |

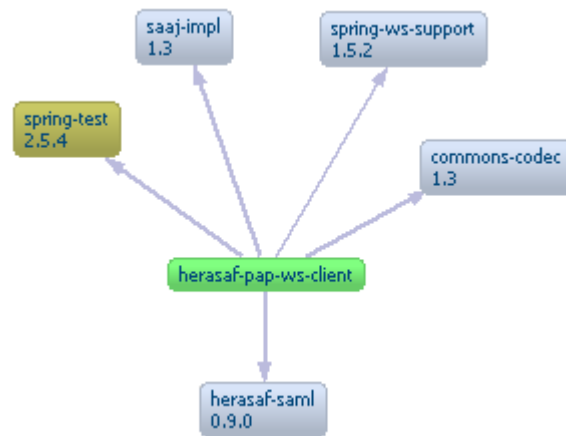
11.2 Maven configuration

Overview

This chapter covers the dependencies between the web service project and other projects.



*pom.xml
visualized*



Legend

herasaf-saml	Saml is needed for the web service client to function.
saaj-impl	SOAP with Attachments API for Java (SAAJ)
spring-ws-support	Provides general helper classes for Spring Web Services.
commons-codec	Provides implementations of common encoders and decoders such as Base64, Hex, Phonetic and URLs.
spring-test	Spring testing is used

12 herasaf-pap-deployment

12.1 Spring configuration

Overview

This chapter explains how the deployment module is configured.

ApplicationCont ext.xml Spring graph



ApplicationCont ext.xml

```

<bean id="deployer"
class="org.herasaf.pap.deployment.Deployer">
  <property name="wsclient" ref="wsclient" />
  <property name="policyTransformer"
ref="policyTransformer" />
</bean>
  
```

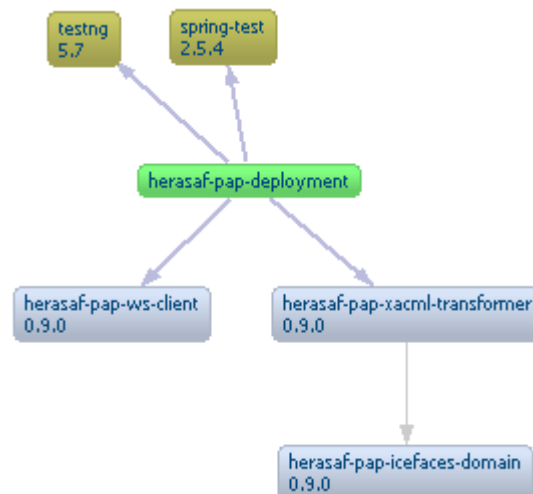
Using Spring's dependency injection, the ws client and the transformere are defined and injected. Refer to 11.1 (page 30, herasaf-pap-icefaces-ws-client - Spring configuration) and 10.1 (page 27, herasaf-pap-xacml-transformer - Spring configuration) for the corresponding configuration.

12.2 Maven configuration

Overview

This chapter covers the dependencies between the deployment project and other projects.

pom.xml visualized



Legend

herasaf-pap-ws-client Needed for communication reasons with a



	PDP. This module is the final sending entity.
herasaf-pap-xacml-transformer	Needed for transformation of <code>PolicyTemplates</code> and <code>PolicySetTemplates</code> .
testng	Testng is used to test the deployment module.
spring-test	Spring testing is used.

13 herasaf-pap-icefaces-web

13.1 Spring configuration

<i>Overview</i>	This chapter explains how the web module is configured.
<i>acegi.ctx.xml</i>	Defines the Acegi configuration. Refer to 13.6 (page 45, <i>Acegi configuration</i>) for the configuration.
<i>all-daos.ctx.xml</i>	Defines all Data Access Objects (DAOs). Refer to 9.1 (page 23, <i>herasaf-pap-icefaces-service - Spring configuration, all-daos.ctx.xml</i>) for the configuration.
<i>all-services.ctx.xml</i>	Defines to service module. Refer to 9.1 (page 23, <i>herasaf-pap-icefaces-service - Spring configuration, all-services.ctx.xml</i>) for the configuration. Additionally, <i>all-daos.ctx.xml</i> is imported.
<i>application-context.ctx.xml</i>	<pre><import resource="acegi.ctx.xml" /> <import resource="all-services.ctx.xml" /> <import resource="ContextAndPolicyConfiguration.xml" /> <import resource="CombiningAlgorithms.xml" /> <import resource="DataTypeAttributes.xml" /> <import resource="deployment.xml" /> <import resource="Functions.xml" /> <import resource="ttransformer.xml" /></pre> <p>All resources are imported.</p>
<i>CombiningAlgorithms.xml</i>	Initializes HERAS ^{AF} XACML <i>URNTToPolicyCombiningAlgorithmConverter</i> and <i>PolicyCombiningAlgorithms</i> . See developers guide of HERAS ^{AF} XACML [DOH07DADev] for further information.
<i>ContextAndPolicyConfiguration.xml</i>	Configures the policy marshalling and unmarshalling. Refer to HERAS ^{AF} XACML developer guide [DOH07DADev] for detailed information.
<i>DataTypeAttributes.xml</i>	Initializes HERAS ^{AF} XACML <i>URNTToDataTypeConverter</i> . See developer guide of HERAS ^{AF} XACML [DOH07DADev] for further information.
<i>deployment.xml</i>	Defines to service module. Refer to 12.1 (page 33, <i>herasaf-pap-deployment - Spring configuration, ApplicationContext.xml</i>). Additionally, <i>ws-client.xml</i> is imported.
<i>Functions.xml</i>	Initializes HERAS ^{AF} XACML <i>URNTToFunctionConverter</i> . See developer guide of HERAS ^{AF} XACML [DOH07DADev] for further information.
<i>Transformer.xml</i>	Defines the Template to XACML transformer. Refer to 10.1 (page 27, <i>herasaf-pap-xacml-transformer - Spring configuration</i>) for further information.
<i>ws-client.xml</i>	Defines the PAP web service client. Refer to 11.1 (page 30, <i>herasaf-pap-icefaces-ws-client, Spring configuration</i>) for further information.

13.2 Maven configuration

Overview

This chapter covers the dependencies between the web project and other projects.

pom.xml visualized



Legend

herasaf-pap-icefaces-service	The service module to handle the persistence of <i>Templates</i> .
herasaf-pap-deployment	The deployment module to communicate with PDPs.
spring-webflow	Spring Web Flow is used to model the user interaction flows.
acegi-security	This library is needed for user authentication and authorization.
ehcache	The <i>CacheManager</i> is needed by Acegi security.



tomahawk	Extends apache myfaces by custom components.
backport-util-concurrent	Needed to be explicitly defined because of dependency exclusions.
el-ri	Needed when using facelets with ICEfaces.
el-api	Needed when using facelets with ICEfaces.
standard	The Apache JSTL implementation.
icefaces	The ICEfaces framework.
icefaces-comps	Offers ICEfaces components.
icefaces-facelets	The ICEfaces facelets API.
myfaces-impl	The Apache implementation of the JSF standard.
servlet-api	The javax.servlet API is needed when using ICEfaces facelets. Furthermore it is needed for a maven build, therefore this dependency has scope:provided. This means it is not included in the final war file, because the container (i.e. Apache Tomcat) provides its own servlet implementation at runtime.
xml-apis	Needs to be explicitly defined because of dependency exclusions.
commons-beanutil	Needs to be explicitly defined because of dependency exclusions.
commons-fileupload	ICEfaces needs this library.
testng	Testng is used to test Spring Web Flows.

13.3 Spring Web Flow configuration

Overview

This chapter covers the configuration of Spring Web Flow

webflow-config.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

  xmlns:flow="http://www.springframework.org/schema/webflow-
  config"
  xsi:schemaLocation="
    http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans-
    2.0.xsd
    http://www.springframework.org/schema/webflow-config
    http://www.springframework.org/schema/webflow-
    config/spring-webflow-config-1.0.xsd">
```



```

    <flow:executor id="flowExecutor" registry-
ref="flowRegistry">
    <flow:repository type="continuation" max-
continuations="-1"/>
    <flow:execution-attributes>
    <flow:alwaysRedirectOnPause value="false" />
    </flow:execution-attributes>
    </flow:executor>

    <flow:registry id="flowRegistry">
    <flow:location path="/WEB-INF/flows/**/*-flow.xml"/>
    </flow:registry>
</beans>

```

2

Legend

-
- 1 Launches new flow executions and resumes existing executions.
It will place no limit on the maximum number of snapshots that can be taken per conversation.
alwaysRedirectOnPause is set to false, thus we have more control over when a redirect is issued. Using a "redirect" prefix in the view attribute of the view states, it is still possible to apply the "redirect after submit"-idiom. This will cause the view to be rendered after a client side redirect.

 - 2 Creates the registry of flow definitions for this application.
Every file ending with "*-flow.xml*" located in */WEB-INF/flows* is recognized as a Spring Web Flow.
This is of importance if new flows are created in near future. Always assure newly create Spring Web Flows end with *-flow.xml*.

13.4 web.xml

Overview

This chapter covers the configuration of the web xml file.
Structure: A part of the web.xml is inserted, afterwards explained, next part of web.xml is inserted and so on.

context-params

```

<context-param>
  <param-name>contextConfigLocation</param-name>
  <param-value>
    /WEB-INF/context/application-context.ctx.xml
    /WEB-INF/webflow-config.xml
  </param-value>
</context-param>

```

Explanation

The context configuration location is specified.
In application-context.ctx.xml the Spring beans are declared.



In webflow-config.xml Spring Web Flows are configured.

The contextConfigLocation is used for Spring's WebApplicationContext (see ContextLoaderListener further down).

```
<context-param>
  <param-name>javax.faces.DEFAULT_SUFFIX</param-name>
  <param-value>.xhtml</param-value>
</context-param>
```

Explanation

The default suffix for pages that contain JSF elements is set to *.jsp*. This is changed here to *.xhtml*, because mostly facelets and ICEfaces elements are used. Furthermore the used technology is not obvious to a user, when *.xhtml* is used as an extension.

```
<context-param>
  <param-name>facelets.DEVELOPMENT</param-name>
  <param-value>>true</param-value>
</context-param>
```

Explanation

Setting *facelets.DEVELOPMENT* to true allows usage of facelets debug logging

```
<context-param>
  <param-name>com.sun.faces.validateXml</param-name>
  <param-value>>true</param-value>
</context-param>
```

Explanation

By setting these context parameters to true, strict verification of XML syntax in JSF is enabled.

```
<context-param>
  <param-name>javax.faces.STATE_SAVING_METHOD</param-name>
  <param-value>server</param-value>
</context-param>
```

Explanation

It is specified where the view state is stored.

The state is stored on server side, not in the client browser.

```
<context-param>
  <param-name>com.icesoft.faces.uploadDirectory</param-
name>
  <param-value>upload</param-value>
</context-param>
```



Explanation

Specifies the directory location where uploaded files are stored.

```
<context-param>
  <param-name>com.icesoft.faces.concurrentDOMViews</param-
name>
  <param-value>>false</param-value>
</context-param>
```

Explanation

To allow multiple windows, concurrent DOM views must be enabled.

```
<context-param>
  <param-name>com.icesoft.faces.synchronousUpdate</param-
name>
  <param-value>>true</param-value>
</context-param>
```

Explanation

Synchronous ICEfaces updates are turned on.

```
<context-param>
  <param-name>facelets.LIBRARIES</param-name>
  <param-value>/WEB-INF/herasaf.taglib.xml</param-value>
</context-param>
```

Explanation

The custom HERAS^{AF} tag library is defined so facelets knows of its presence.

filter

```
<filter>
  <filter-name>Acegi Filter Chain Proxy</filter-name>
  <filter-class>
    org.acegisecurity.util.FilterToBeanProxy
  </filter-class>
  <init-param>
    <param-name>targetClass</param-name>
    <param-value>
      org.acegisecurity.util.FilterChainProxy
    </param-value>
  </init-param>
</filter>
<filter-mapping>
  <filter-name>Acegi Filter Chain Proxy</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>
```

Explanation

Configuration for Acegi. Every URL beneath the webapp directory is



analyzed if the current user has a validated session.

listener

```
<listener>
  <listener-class>
    org.springframework.web.context.ContextLoaderListener
  </listener-class>
</listener>
```

Explanation

Listener to start up Spring's root `WebApplicationContext`. It reads `contextConfigLocation` files (defined above) and creates the `WebApplicationContext` for the web application. Thus we can use Spring application context within the context of the web application server.

servlet

```
<servlet>
  <servlet-name>Faces Servlet</servlet-name>
  <servlet-class>javax.faces.webapp.FacesServlet</servlet-
class>
  <load-on-startup>1</load-on-startup>
</servlet>
```

Explanation

Defines the Faces servlet, the engine of all JSF applications.

```
<servlet>
  <servlet-name>Persistent Faces Servlet</servlet-name>
  <servlet-class>
    com.icesoft.faces.webapp.xmlhttp.PersistentFacesServlet
  </servlet-class>
  <load-on-startup>1</load-on-startup>
</servlet>
```

Explanation

ICEfaces own version of `FacesServlet`.

```
<servlet>
  <servlet-name>Blocking Servlet</servlet-name>
  <servlet-class>
    com.icesoft.faces.webapp.xmlhttp.BlockingServlet
  </servlet-class>
  <load-on-startup>1</load-on-startup>
</servlet>
```

Explanation

Additional ICEfaces servlet for handling asynchronous updates.

servlet-mapping

```
<servlet-mapping>
  <servlet-name>Persistent Faces Servlet</servlet-name>
  <url-pattern>*.iface</url-pattern>
</servlet-mapping>
```



Explanation

PersistentFacesServlet (the ICEfaces servlet implementation) is mapped to **iface*.

```
<servlet-mapping>
  <servlet-name>Persistent Faces Servlet</servlet-name>
  <url-pattern>/xmlhttp/*</url-pattern>
</servlet-mapping>
```

Explanation

PersistentFacesServlet (the ICEfaces servlet implementation) is mapped to */xmlhttp/**.

This mapping is for ICEfaces' internal use.

```
<servlet-mapping>
  <servlet-name>Blocking Servlet</servlet-name>
  <url-pattern>/block/*</url-pattern>
</servlet-mapping>
```

Explanation

BlockingServlet (ICEfaces) is mapped to */block/**.

This mapping is for ICEfaces' internal use.

session-config

```
<session-config>
  <session-timeout>3600</session-timeout>
</session-config>
```

Explanation

This tag provides a way to specify the timeout for HTTP sessions, overriding the default time of the container.

welcome-file-list

```
<welcome-file-list>
  <welcome-file>index.jsp</welcome-file>
</welcome-file-list>
```

Explanation

The welcome file is provided when the web application is accessed.

13.5 faces-config.xml

Overview

This chapter covers faces-config.xml

The faces config is used to configure any managed-beans that are required for JSF.



```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE faces-config PUBLIC "-//Sun Microsystems,
Inc.//DTD JavaServer Faces Config 1.1//EN"
"http://java.sun.com/dtd/web-facesconfig_1_1.dtd">
<faces-config
xmlns="http://java.sun.com/JSF/Configuration">

<!-- Insert all managed beans here -->

<navigation-rule>
  <navigation-case>
    <from-outcome>logout</from-outcome>
    <to-view-id>/pages/logon/login.iface</to-view-id>
  </navigation-case>
</navigation-rule>

<navigation-rule>
  <navigation-case>
    <from-action>
      #{authenticationController.authenticate}
    </from-action>
    <from-outcome>success</from-outcome>
    <to-view-id>/index.iface</to-view-id>
    <redirect />
  </navigation-case>
  <navigation-case>
    <from-action>
      #{authenticationController.authenticate}
    </from-action>
    <from-outcome>denied</from-outcome>
    <to-view-id>/accessDenied.jsp</to-view-id>
    <redirect />
  </navigation-case>
</navigation-rule>

<application>

  <variable-resolver>
    org.springframework.web.jsf.DelegatingVariableResolver
  </variable-resolver>

  <view-handler>
    com.icesoft.faces.facelets.D2DFaceletViewHandler
  </view-handler>

  <navigation-handler>
    org.springframework.webflow.executor.jsf.FlowNavigationHand
    ler
  </navigation-handler>

  <variable-resolver>
    org.springframework.webflow.executor.jsf.DelegatingFlowVari
    ableResolver

```

1

2

3

4

5

6

7



```

</variable-resolver>

<message-bundle>
  org.herasaf.pap.jsf.locale.messages
</message-bundle>

<locale-config>
  <default-locale>en</default-locale>
  <supported-locale>de</supported-locale>
</locale-config>

</application>

<lifecycle>
  <phase-listener>

org.springframework.webflow.executor.jsf.FlowPhaseListener
  </phase-listener>
</lifecycle>

</faces-config>

```

8

9

10

Legend

- | | |
|---|--|
| 1 | All managed beans are defined for usage in JSF. The complete listing is omitted, because of overview reasons. |
| 2 | Defines navigation after a successful logout.
Redirects to login page. |
| 3 | Navigations for login screen, handled by Acegi security. Defines two pages for a user who successfully or unsuccessfully tried to login. |
| 4 | JSF <code>VariableResolver</code> that first delegates to the original resolver of the underlying JSF implementation, then to the Spring root <code>WebApplicationContext</code> .

All JSF expressions can then implicitly refer to the names of Spring-managed service layer beans, for example in property values of JSF-managed beans. |
| 5 | Sets the application view handler for JSF. Uses <code>D2DViewHandler</code> , an ICEfaces <code>Facelet ViewHandler</code> implementation. |
| 6 | A Spring implementation of a JSF <code>NavigationHandler</code> that provides integration with Spring Web Flow. Responsible for delegating to Spring Web Flow to launch and resume flow executions, treating JSF action outcomes (like a command button click) as Web Flow events. |
| 7 | Defines a Spring variable resolver that searches the current flow execution for variables to resolve. |
| 8 | Defines the path to the message bundle. |
| 9 | Defines the languages that can be used. Default is English (depends on preferences in the web browser that accesses the web pages), if |

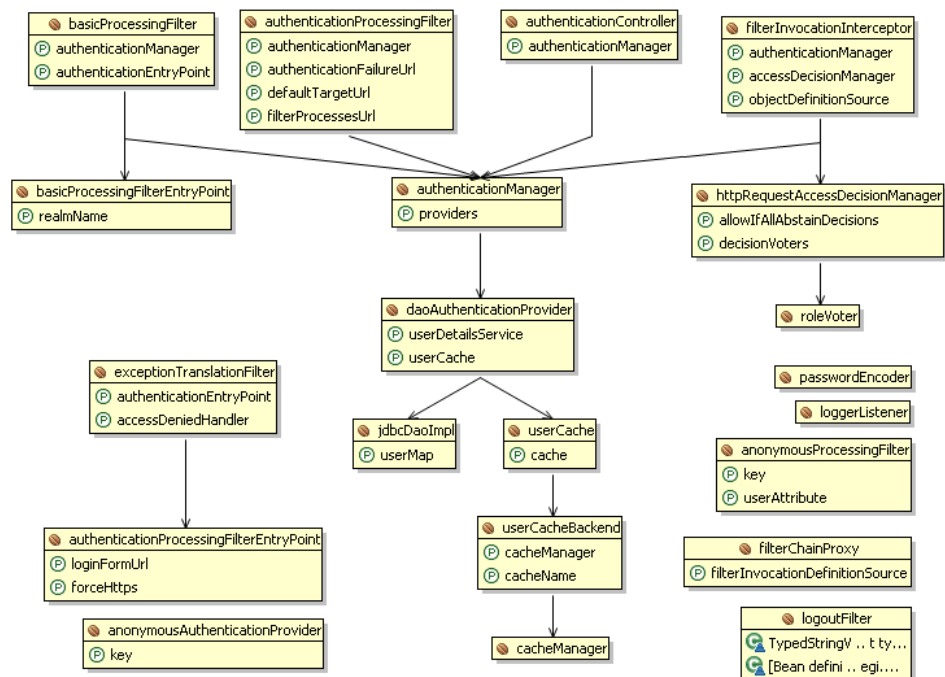
not German is set. Can be expanded by any language desired. Refer to 19.2.5 (page 106, *Implementation - Internationalization*) for instructions.

10 The JSF phase listener is responsible for managing the FlowExecution object lifecycle in a JSF environment.

13.6 Acegi configuration

Overview

This chapter covers the configuration of Acegi security.



Legend

The Acegi configuration visually displayed.

13.7 Taglib configuration

Overview

This chapter covers the configuration of HERAS^{AF} PAP tag libraries that were written. Refer to 19.2.1 (page 81, herasaf-pap-icefaces-web - Implementation - Facelets) for detailed information on how these tag libraries can be used in *.xhtml* documents.

herasaf.taglib.xml

```
<?xml version="1.0"?>
<!DOCTYPE facelet-taglib PUBLIC
  "-//Sun Microsystems, Inc.//DTD Facelet Taglib 1.0//EN"
  "facelet-taglib_1_0.dtd">
```



```

<facelet-taglib>
  <namespace>http://www.herasaf.org/jsf</namespace>
  <tag>
    <tag-name>backButton</tag-name>
</facelet-taglib>
<source>../pages/common/facelets/button_back.xhtml</source>
</tag>
<!-- All further tags are inserted here -->
</facelet-taglib>

```

Legend

namespace	This is the namespace that has to be included in pages where this tag library is used. For example if access to a back button is needed, it can be achieved by following code snippet:
-----------	--

```

<html>
<f:view
xmlns:f="http://java.sun.com/jsf/core"
xmlns:herasaf="http://www.herasaf.org/jsf">
  <head />
  <body>
    <herasaf:backButton />
  </body>
</f:view>
</html>

```

The namespace has to match with the one defined in the tag library and the used tag needs to be defined in the tag library too.

tag	A tag, includes a tagname and a source
tagname	The name of the tag. This will be used in pages to access the referenced source, for example: <pre><herasaf:backButton /></pre> where backButton is the actual tagname defined in herasaf.taglib.xml
source	A source file. In HERAS ^{AF} PAP these source pages are JSF facelets compositions, thus they can be reused at any location in web pages.

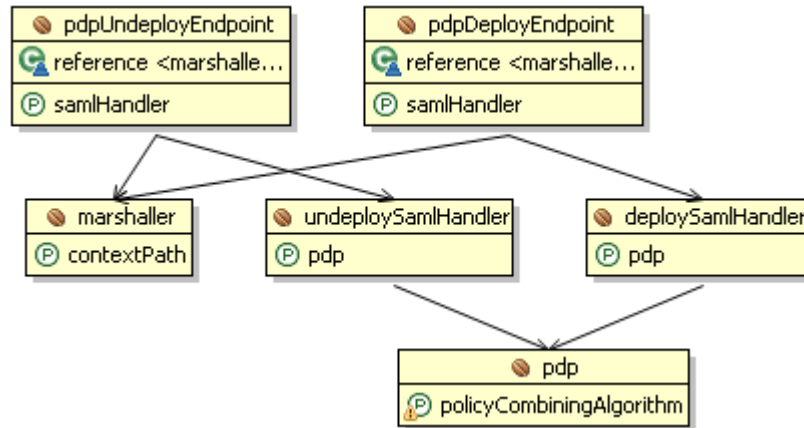
14 herasaf-pdp-ws-policy

14.1 Spring configuration

Overview

This chapter explains how the web service client module is configured.

Spring graph from integration tests



herasaf-pdp-ws-servlet.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:context="http://www.springframework.org/schema/context"
  xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-2.0.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-2.5.xsd">
  <context:annotation-config />
  <import resource="./context/ApplicationContext.ctx.xml" />
</beans>

<bean id="pdpUndeployEndpoint"
  class="org.herasaf.pdp.ws.policy.endpoints.PdpEndpoint">
  <constructor-arg ref="marshaller" />
  <property name="samlHandler" ref="undeploySamlHandler" />
</bean>

<bean id="pdpDeployEndpoint"
  class="org.herasaf.pdp.ws.policy.endpoints.PdpEndpoint">
  <constructor-arg ref="marshaller" />
  <property name="samlHandler" ref="deploySamlHandler" />
</bean>

<bean id="deploySamlHandler"
  class="org.herasaf.pdp.ws.policy.saml.DeploySamlHandler">
  <property name="pdp" ref="pdp" />
</bean>
```

1

2

3



```

</bean>

<bean id="undeploySamlHandler"
      class="org.herasaf.pdp.ws.policy.saml.
            UndeploySamlHandler">
  <property name="pdp" ref="pdp" />
</bean>

<bean id="pdp"
      class="org.herasaf.xacml.pdp.impl.PDPImpl">
  <property name="policyCombiningAlgorithm">
    <ref bean="onlyOneApplicableAlgorithm" />
  </property>
</bean>

<bean id="marshaller"
      class="org.herasaf.saml.marshaller.
            Jaxb2NamespaceMarshaller">
  <property name="contextPath"
    value="org.herasaf.xacml.core.context.impl:org.herasaf.xacml.core.policy.impl:org.herasaf.saml.jaxb.saml.assertion:org.herasaf.saml.jaxb.saml.protocol:org.herasaf.saml.jaxb.w3.xmlldsig:org.herasaf.saml.jaxb.w3.xmlenc:org.herasaf.saml.jaxb.xacml.assertion:org.herasaf.saml.jaxb.xacml.protocol" />
</bean>

<bean id="wsdl"
      class="org.springframework.ws.wsdl.
            wsdl11.SimpleWsdl11Definition">
  <constructor-arg
    value="/WEB-INF/wsdl/herasaf-pdp-ws-policy.wsdl" />
</bean>

<bean id="endpointMapping"
      class="org.springframework.ws.soap.
            addressing.server.SimpleActionEndpointMapping">
  <property name="mappings">
    <props>
      <prop
        key="http://localhost:8081/herasaf-pdp-ws-policy-integrationtests/pdppolicy-service/deploy-policies">
        pdpDeployEndpoint
      </prop>
      <prop
        key="http://localhost:8081/herasaf-pdp-ws-policy-integrationtests/pdppolicy-service/undeploy-policies">
        pdpUndeployEndpoint
      </prop>
    </props>
  </property>
</bean>
</beans>

```

4

5

6

7

Legend

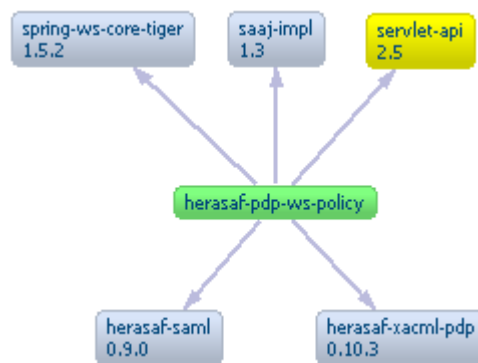
- | | |
|---|---|
| 1 | Defining a bean for the undeployment of policies. By setting the <code>samlHandler</code> , it is decided what sort of endpoint this will be. |
| 2 | Defining a bean for the deployment of policies. By setting the <code>samlHandler</code> , it is decided what sort of endpoint this will be. |
| 3 | The <code>undeploySamlHandler</code> undeploys from the specified <code>pdp</code> . |
| 4 | The <code>deploySamlHandler</code> deploys to the specified <code>pdp</code> . |
| 5 | Definition of the PDP implementation from HERAS ^{AF} XACML 2.0. |
| 6 | By simply defining this bean, Spring's <code>MessageDispatcherServlet</code> automatically detects any WSDL definitions in the container. These WSDLs get exposed via the <code>WsdldefinitionHandlerAdapter</code> . |
| 7 | This is the most important bean: by defining the <code>SimpleActionEndpointMapping</code> , messages get routed to the defined endpoint bean depending on the WS-addressing action header in the SOAP message header. |

14.2 Maven configuration

Overview

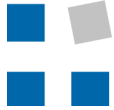
This chapter covers the dependencies between the HERAS^{AF}-pdp-ws-policy module and other projects.

pom.xml visualized



Legend

herasaf-saml	Needed for handling SOAP messages and for exception handling.
herasaf-xacml-pdp	Needed so the SamlHandlers can deploy policies.
spring-ws-core-tiger	One of the two central WS-core packages. This package has a dependency to <code>spring-ws-core</code> . Those two are needed to implement web services using Spring.



saaj-impl	saaj-impl is needed for sending SOAP-messages. Additionally, saaj-impl itself has a dependency to saaj-api which allows for example to manipulate SOAP envelopes or SOAP headers.
servlet-api	servlet-api is needed for a maven build, therefore this dependency has scope:provided. This means it is not included in the final war file, because the container (i.e. Apache Tomcat) has its own servlet implementation.

Part III: Architecture and Design

15 PAP Architecture

Overview

This chapter provides insight on the architecture of HERAS^{AF} PAP implementation.

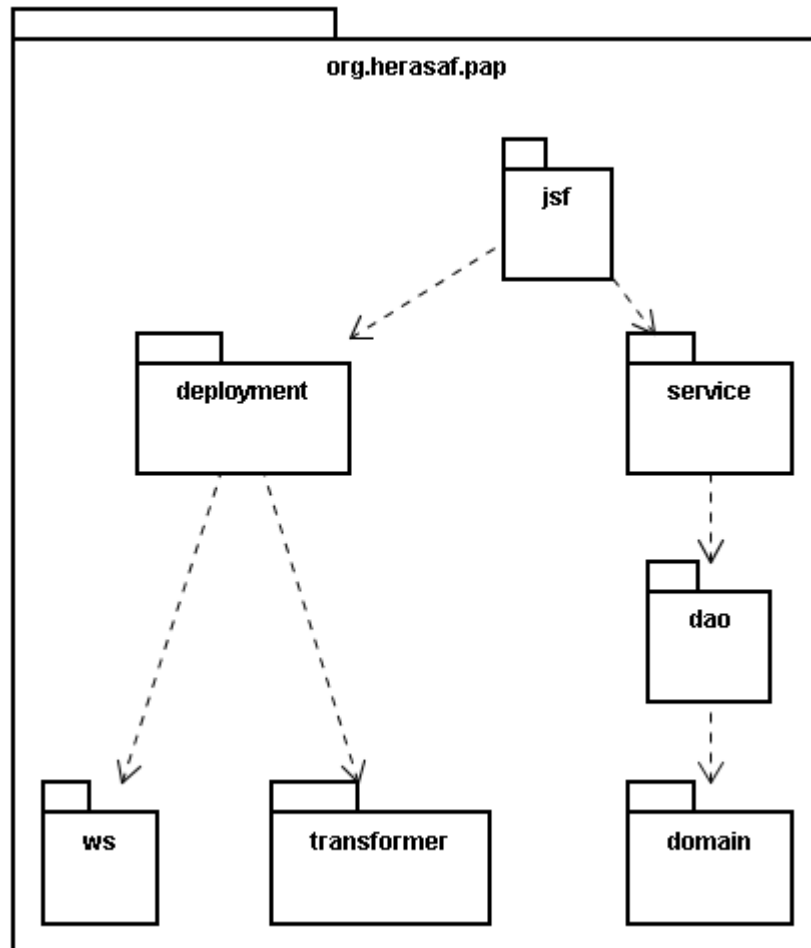
In order to understand the big picture of interaction between the PAP and further XACML parts, refer to the XACML Developers Guide [DOH07DADev].

15.1 Modules

Overview

This chapter shows how the PAP modules are wired and provides a short description of each module.

Module diagram



Module domain

The domain module contains all templates and the wiring between the different templates.

Module dao

The `dao` module handles the persistence of the objects in the domain module.

Module service

The service module acts as a façade to the JSF module and delegates database queries to the DAO layer.

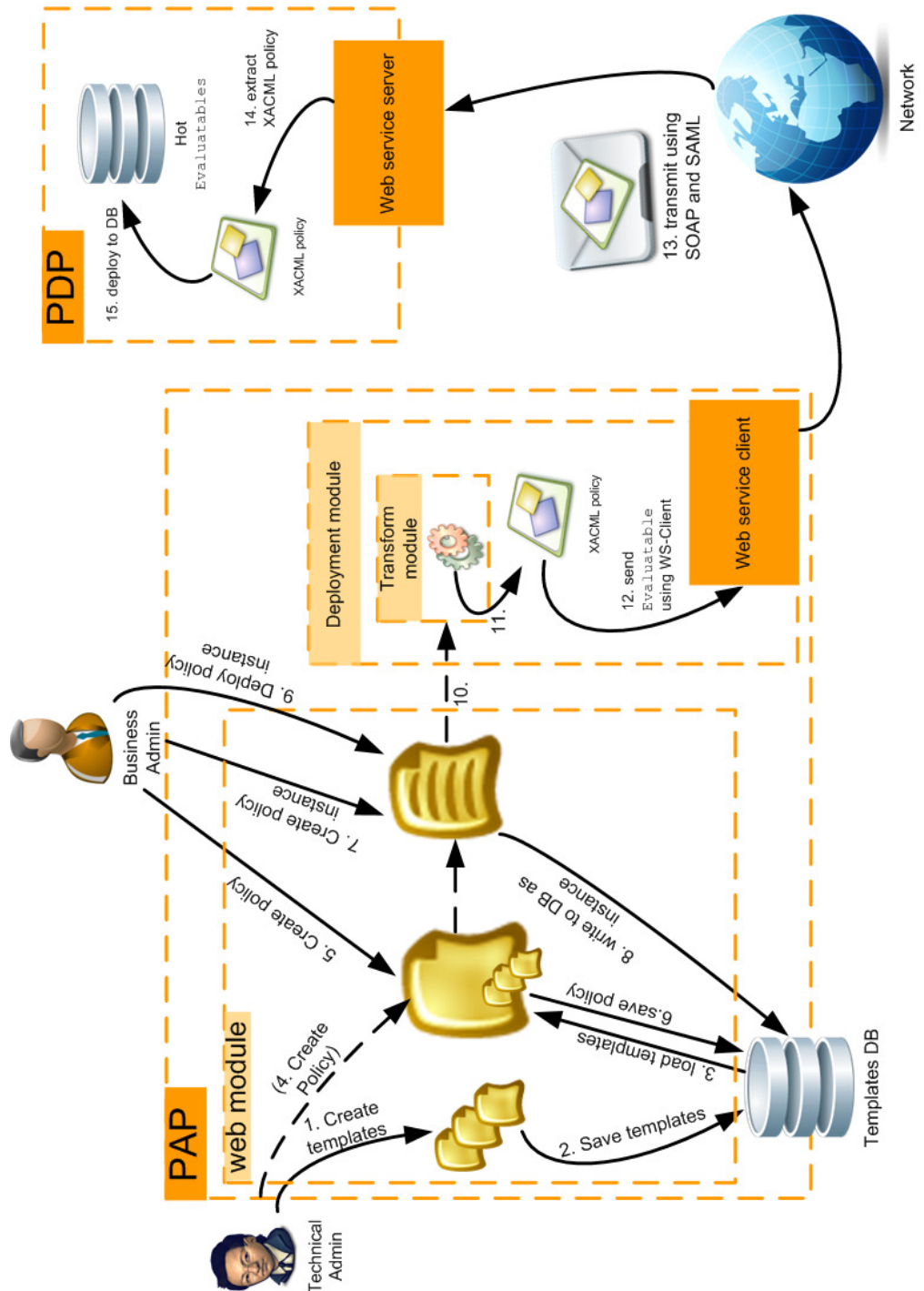
<i>Module jsf</i>	The jsf module is the web front end where administrators can CRUD templates and deploy them.
<i>Module deployment</i>	The deployment module is responsible deployment of given templates. Firstly it transforms given templates to valid XACML and hand them on to the ws module.
<i>Module ws</i>	This module handles communication with the HERAS ^{AF} PDP ws policy module.
<i>Module transformer</i>	The transformer module is responsible for transforming templates into their corresponding XACML representation.

15.2 PAP interaction scenario

Overview

This chapter explains how the interaction scenario of the HERAS^{AF} PAP looks like. It gives you an overview of the most important parts in the process.

Visualization of the deployment process



Steps explained

The following list shortly explains every point in the illustration from above:

Creating and storing steps

-
- (1) & (2) The technical administrator creates templates and saves them into the database.
 - (3), (4) & (5) Either the technical administrator or the business administrator creates a `PolicyTemplate` or `PolicySetTemplate` by loading and selecting the desired templates that make them up. `PolicyTemplates` and `PolicySetTemplates` are representations in the HERAS^{AF} template model of XACML's `PolicyType` and `PolicySetType`.
 - (6) The `PolicyTemplate` or `PolicySetTemplate` is saved to the database.
 - (7) & (8) The business administrator fills out business-related data. Alternatively, the technical administrator already may have entered part or all of the data. After the `PolicyTemplate` or `PolicySetTemplate` is completed with all necessary data, it is saved into the database by creating a clone. See **Error! Reference source not found.**, page **Error! Bookmark not defined.** **Error! Reference source not found.** for more information on this topic.

Deployment steps

-
- (9) The business administrator wants to deploy a `PolicyTemplate` or `PolicySetTemplate` and therefore selects the desired one in the web user interface.

The following steps are carried out automatically by the deployment module without user interaction.
 - (10) The `PolicyTemplate` or `PolicySetTemplate` is handed over to the deployment module, after the `policyId` or `policySetId` has been set. These ids unambiguously identify a `PolicyType` or `PolicySetType`.
 - (11) & (12) The passed in `PolicyTemplate` or `PolicySetTemplate` is transformed into an HERAS^{AF}XACML 2.0 `Evaluatable` before being handed over to the web service.
 - (13) The `Evaluatable` is encapsulated into SOAP and SAML before being transmitted.
 - (14) & (15) Once received by the PDP web service endpoint, the `Evaluatable` is unwrapped and deployed into the PDP's database. This `Evaluatable` is actively being integrated into the evaluation process of the PDP for incoming requests.

16 herasaf-pap-icefaces-domain

Overview This chapter covers the package `org.herasaf.pap.domain`.

16.1 Interfaces / Abstract classes

Overview This chapter explains the important interfaces and abstract classes inside the domain module. The structure and the important methods are described.

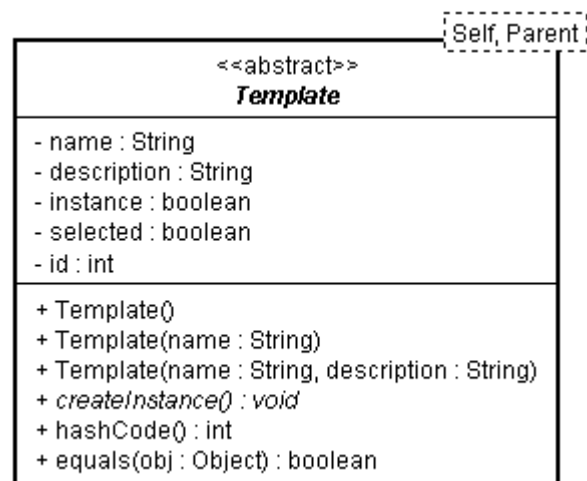
16.1.1 Abstract class `Template`

Description The abstract class `Template<Self, Parent>` is the super class of all `Templates`. It contains fundamental information every template must have, therefore this class was introduced. It implements `Replicable<Self, Parent>` (see 16.1.4).

The abstract method `createInstance()` has to be implemented by every template itself because every template has another wiring with its children templates.

Because the abstract class `Template` implements the `Replicable` interface, all subclasses need to implement the methods of `Replicable`.

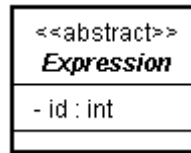
UML



16.1.2 Abstract class `Expression`

Description The abstract class `Expression` is the super class of `Apply`, `AttributeValue`, `MatchAttributeDefinition`, `MatchingFunction` and `VariableReference`. Its purpose is to mark its subclasses as `Expressions` and provide them with the setter and getter methods for the `id`. For example, `Apply` holds a `Set<Expressions>`.

UML



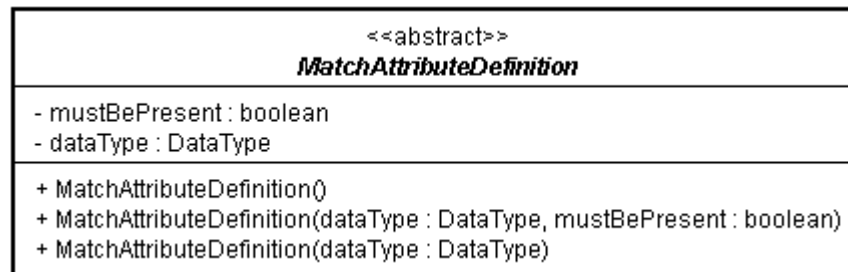
16.1.3 Abstract class MatchAttributeDefinition

Description

The abstract class `MatchAttributeDefinition` is the super class of `AttributeDesignator` and `AttributeSelector`.

The purpose of this class is to unite attributes that every sub class needs.

UML



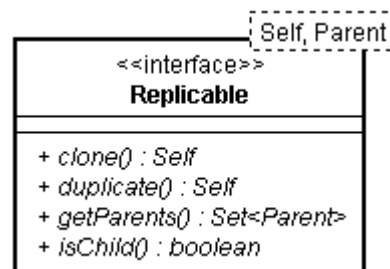
16.1.4 Interface Replicable

Description

The interface `Replicable<Self, Parent>` defines the replication methods that need to be implemented by every `Template`.

`Self` stands for the current template that implements this interface, `Parent` stands for the parent template the implementing template has references to.

UML



16.1.5 Interface CloneNamingStrategy

Description

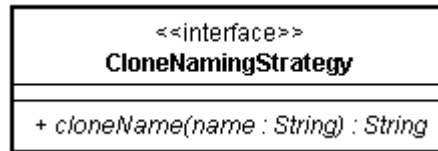
The interface `CloneNamingStrategy` generates a cloned name for a given name.

Every implementation of `CloneNamingStrategy` has to implement the method



```
cloneName().
```

UML



16.2 Implementation

Overview

This chapter covers the implementation of `org.herasaf.pap.domain`. This package forms the core of the HERAS^{AF} PAP. It is an abstraction of the OASIS XACML 2.0 standard with focus of reusability by providing a templating mechanism.

Furthermore, as it is an abstraction, it shall simplify the usage of XACML 2.0.



- The *BAdmin* can only choose from `ActionTemplates`, `EnvironmentTemplates`, `ResourceTemplates` and `SubectTemplates` created by the *TAdmin* to be included in his `PolicyTemplates`, `RuleTemplates` and `PolicySetTemplates` respectively. Thus, he does not have to know many technical details that are necessary when creating `ActionTemplates`, `EnvironmentTemplates`, `ResourceTemplates` or `SubjectTemplates`.
- The *BAdmin* can use `ActionTemplates`, `EnvironmentTemplates`, `ResourceTemplates` and `SubectTemplates` without the help of the *TAdmin* but he is not allowed to create them.

Relationship to Target- Template

Discussion our solution:

The type and cardinality of the relationships between `TargetTemplate` and `PolicySetTemplate`, `TargetTemplate` and `RuleTemplate` and `TargetTemplate` and `PolicyTemplate` are set to be 1:1. This is not the case for example in the HERAS^{AF} XACML implementation.

This decision was made because of usability reasons. The user does not have to explicitly create a `TargetTemplate`. The Reusability of the element `TargetTemplate` is doubtful, as it only consists of a collection of `ActionTemplates`, `EnvironmentTemplates`, `ResourceTemplates` or `SubectTemplates`. Thus a useful name has to be given to that `TargetTemplate` too, which in fact would be a concatenation of every single child name.

Thus a `TargetTemplate` is explicitly created. If the `PolicySetTemplate`, `PolicyTemplate` or `RuleTemplate` is being deleted, the `TargetTemplate` gets deleted as well through proper cascading.

Typification of Attribute- Designator

In order to be able to distinguish which `ActionTemplate`, `EnvironmentTemplate`, `ResourceTemplate` or `SubectTemplate` an `AttributeDesignator` has been referenced by, two alternatives are cogitable:

- Create a subclass for every `AttributeDesignator`
- `AttributeDesignator` references an enum so it can distinguish what template class it has been referenced by

Discussion of our solution:

We introduced the enum *Type*. This was mainly because the three classes `ResourceAttributeDesignator`, `ActionAttributeDesignator` and `EnvironmentAttributeDesignator` would be “marker” classes only with exactly the same semantics. Furthermore, the usage of `instanceof` can lead to exceptions and thus stop HERAS^{AF} from working.

Subject- Attribute- Designator

Since there is an optional field `SubjectCategory` in XACML [XacmlSpec, page 64], it has to be possible to hand in this information in the class `AttributeDesignator`.



Discussion of our solution:

We have chosen to introduce a sub class of `AttributeDesignator` named `SubjectAttributeDesignator`. The alternative would have been to define a field `Category` in `AttributeDesignator`. However, this would violate the information expert principle [InfoExpert], which we wanted to avoid.

17 herasaf-pap-icefaces-dao

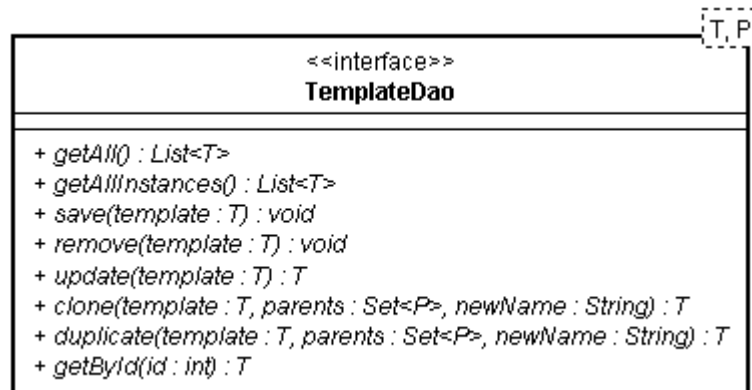
Overview This chapter covers the package `org.herasaf.pap.dao`

17.1 Interfaces

Overview This chapter covers the interfaces `org.herasaf.pap.dao`.

Because there are many `xxxTemplateDaos` which have the same functionality, sub chapters for every type were omitted.

TemplateDao



Common operations

All `xxxTemplateDAOs` implement the interface `TemplateDao` corresponding to their underlying `Template`.

Operation	Description
<code>getAll()</code>	Returns a list of all corresponding templates where <code>instance</code> is set to <code>false</code> . This ensures that no instantiated templates can be used.
<code>getAllInstances()</code>	Returns a list of all corresponding templates where <code>instance</code> is set to <code>true</code> .
<code>save()</code>	Persists a corresponding template into the database.
<code>remove()</code>	Deletes a corresponding template from the database.
<code>update()</code>	Actualizes an already existing template in the database.
<code>clone()</code>	Clones the supplied template. Additionally the wiring with parents is adjusted if any parents were passed.
<code>duplicate()</code>	Duplicates the supplied template. Additionally the wiring with parents is adjusted if any parents were passed
<code>getById()</code>	Returns the corresponding template by its database-id.


 Actions_
 TemplateDAO

<<interface>> ActionsTemplateDAO
+ getAll() : List<ActionsTemplate> + getAllInstances() : List<ActionsTemplate> + save(actionsTemplate : ActionsTemplate) : void + remove(actionsTemplate : ActionsTemplate) : void + update(actionsTemplate : ActionsTemplate) : ActionsTemplate + clone(actionsTemplate : ActionsTemplate, parents : Set<TargetTemplate>, newName : String) : ActionsTemplate + duplicate(actionsTemplate : ActionsTemplate, parents : Set<TargetTemplate>, newName : String) : ActionsTemplate + getById(id : int) : ActionsTemplate

 Action-
 TemplateDAO

<<interface>> ActionTemplateDAO
+ getAll() : List<ActionTemplate> + getAllInstances() : List<ActionTemplate> + save(actionTemplate : ActionTemplate) : void + remove(actionTemplate : ActionTemplate) : void + update(actionTemplate : ActionTemplate) : ActionTemplate + clone(actionTemplate : ActionTemplate, parents : Set<ActionsTemplate>, newName : String) : ActionTemplate + duplicate(actionTemplate : ActionTemplate, parents : Set<ActionsTemplate>, newName : String) : ActionTemplate + getById(id : int) : ActionTemplate

 Condition-
 TemplateDAO

<<interface>> ConditionTemplateDAO
+ getAll() : List<ConditionTemplate> + getAllInstances() : List<ConditionTemplate> + remove(conditionTemplate : ConditionTemplate) : void + save(conditionTemplate : ConditionTemplate) : void + update(conditionTemplate : ConditionTemplate) : ConditionTemplate + clone(conditionTemplate : ConditionTemplate, parents : Set<RuleTemplate>, newName : String) : ConditionTemplate + duplicate(conditionTemplate : ConditionTemplate, parents : Set<RuleTemplate>, newName : String) : ConditionTemplate + getById(id : int) : ConditionTemplate

 Environments_
 TemplateDAO

<<interface>> EnvironmentsTemplateDAO
+ getAll() : List<EnvironmentsTemplate> + getAllInstances() : List<EnvironmentsTemplate> + save(environmentsTemplate : EnvironmentsTemplate) : void + remove(environmentsTemplate : EnvironmentsTemplate) : void + update(environmentsTemplate : EnvironmentsTemplate) : EnvironmentsTemplate + clone(environmentsTemplate : EnvironmentsTemplate, parents : Set<TargetTemplate>, newName : String) : EnvironmentsTemplate + duplicate(environmentsTemplate : EnvironmentsTemplate, parents : Set<TargetTemplate>, newName : String) : EnvironmentsTemplate + getById(id : int) : EnvironmentsTemplate

 Environment-
 TemplateDAO

<<interface>> EnvironmentTemplateDAO
+ getAll() : List<EnvironmentTemplate> + getAllInstances() : List<EnvironmentTemplate> + save(environmentTemplate : EnvironmentTemplate) : void + remove(environmentTemplate : EnvironmentTemplate) : void + update(environmentTemplate : EnvironmentTemplate) : EnvironmentTemplate + clone(environmentTemplate : EnvironmentTemplate, parents : Set<EnvironmentsTemplate>, newName : String) : EnvironmentTemplate + duplicate(environmentTemplate : EnvironmentTemplate, parents : Set<EnvironmentsTemplate>, newName : String) : EnvironmentTemplate + getById(id : int) : EnvironmentTemplate

PolicySet-
TemplatedAO

<<interface>> PolicySetTemplateDAO	
<pre>+ getAll() : List<PolicySetTemplate> + getAllInstances() : List<PolicySetTemplate> + remove(policySetTemplate : PolicySetTemplate) : void + save(policySetTemplate : PolicySetTemplate) : void + update(policySetTemplate : PolicySetTemplate) : PolicySetTemplate + clone(policySetTemplate : PolicySetTemplate, parents : Set<PolicySetTemplate>, newName : String) : PolicySetTemplate + duplicate(policySetTemplate : PolicySetTemplate, parents : Set<PolicySetTemplate>, newName : String) : PolicySetTemplate + getById(id : int) : PolicySetTemplate + getAllDeployed() : List<PolicySetTemplate> + getAllUndeployed() : List<PolicySetTemplate></pre>	

Operation	Description
getAll- Deployed()	Returns a list of all PolicySetTemplates where deployed is set to true. This implies that these PolicySetTemplates are deployed on a PDP.
getAll- Undeployed()	Returns a list of all PolicySetTemplates where deployed is set to false. This implies that these PolicySetTemplates are not yet deployed on any PDPs.

Policy-
TemplatedAO

<<interface>> PolicyTemplateDAO	
<pre>+ getAll() : List<PolicyTemplate> + getAllInstances() : List<PolicyTemplate> + getAllDeployed() : List<PolicyTemplate> + remove(policyTemplate : PolicyTemplate) : void + save(policyTemplate : PolicyTemplate) : void + update(policyTemplate : PolicyTemplate) : PolicyTemplate + clone(policyTemplate : PolicyTemplate, parents : Set<PolicySetTemplate>, newName : String) : PolicyTemplate + duplicate(policyTemplate : PolicyTemplate, parents : Set<PolicySetTemplate>, newName : String) : PolicyTemplate + getById(id : int) : PolicyTemplate + getAllUndeployed() : List<PolicyTemplate></pre>	

Operation	Description
getAllDeploy ed()	Returns a list of all PolicyTemplates where deployed is set to true. This implies that these PolicyTemplates are deployed on a PDP.
getAllUndepl oyed()	Returns a list of all PolicyTemplates where deployed is set to false. This implies that these PolicyTemplates are not yet deployed on any PDPs.

Resources-
TemplatedAO

<<interface>> ResourcesTemplateDAO	
<pre>+ getAll() : List<ResourcesTemplate> + getAllInstances() : List<ResourcesTemplate> + save(resourcesTemplate : ResourcesTemplate) : void + remove(resourcesTemplate : ResourcesTemplate) : void + update(resourcesTemplate : ResourcesTemplate) : ResourcesTemplate + clone(resourcesTemplate : ResourcesTemplate, parents : Set<TargetTemplate>, newName : String) : ResourcesTemplate + duplicate(resourcesTemplate : ResourcesTemplate, parents : Set<TargetTemplate>, newName : String) : ResourcesTemplate + getById(id : int) : ResourcesTemplate</pre>	



Resource-
TemplateDAO

<<interface>> ResourceTemplateDAO
<pre>+ getAll() : List<ResourceTemplate> + getAllInstances() : List<ResourceTemplate> + remove(resourceTemplate : ResourceTemplate) : void + save(resourceTemplate : ResourceTemplate) : void + update(resourceTemplate : ResourceTemplate) : ResourceTemplate + clone(resourceTemplate : ResourceTemplate, parents : Set<ResourceTemplate>, newName : String) : ResourceTemplate + duplicate(resourceTemplate : ResourceTemplate, parents : Set<ResourceTemplate>, newName : String) : ResourceTemplate + getByid(id : int) : ResourceTemplate</pre>

Rule-
TemplateDAO

<<interface>> RuleTemplateDAO
<pre>+ getAll() : List<RuleTemplate> + getAllInstances() : List<RuleTemplate> + save(ruleTemplate : RuleTemplate) : void + remove(ruleTemplate : RuleTemplate) : void + update(ruleTemplate : RuleTemplate) : RuleTemplate + clone(ruleTemplate : RuleTemplate, parents : Set<PolicyTemplate>, newName : String) : RuleTemplate + duplicate(ruleTemplate : RuleTemplate, parents : Set<PolicyTemplate>, newName : String) : RuleTemplate + getByid(id : int) : RuleTemplate</pre>

Subjects_
TemplateDAO

<<interface>> SubjectsTemplateDAO
<pre>+ getAll() : List<SubjectsTemplate> + getAllInstances() : List<SubjectsTemplate> + save(subjectsTemplate : SubjectsTemplate) : void + remove(subjectsTemplate : SubjectsTemplate) : void + update(subjectsTemplate : SubjectsTemplate) : SubjectsTemplate + clone(subjectsTemplate : SubjectsTemplate, parents : Set<TargetTemplate>, newName : String) : SubjectsTemplate + duplicate(subjectsTemplate : SubjectsTemplate, parents : Set<TargetTemplate>, newName : String) : SubjectsTemplate + getByid(id : int) : SubjectsTemplate</pre>

Subject-
TemplateDAO

<<interface>> SubjectTemplateDAO
<pre>+ getAll() : List<SubjectTemplate> + getAllInstances() : List<SubjectTemplate> + remove(subjectTemplate : SubjectTemplate) : void + save(subjectTemplate : SubjectTemplate) : void + update(subjectTemplate : SubjectTemplate) : SubjectTemplate + clone(subjectTemplate : SubjectTemplate, parents : Set<SubjectsTemplate>, newName : String) : SubjectTemplate + duplicate(subjectTemplate : SubjectTemplate, parents : Set<SubjectsTemplate>, newName : String) : SubjectTemplate + getByid(id : int) : SubjectTemplate</pre>

TargetMatch-
TemplateDAO

<<interface>> TargetMatchTemplateDAO	
<pre> + getAll() : List<TargetMatchTemplate> + getAllByType(type : String) : List<TargetMatchTemplate> + getAllInstances() : List<TargetMatchTemplate> + remove(targetMatch : TargetMatchTemplate) : void + save(targetMatch : TargetMatchTemplate) : void + update(targetMatch : TargetMatchTemplate) : TargetMatchTemplate + clone(targetMatchTemplate : TargetMatchTemplate, parents : Set<Template>, newName : String) : TargetMatchTemplate + duplicate(targetMatchTemplate : TargetMatchTemplate, parents : Set<Template>, newName : String) : TargetMatchTemplate + getById(id : int) : TargetMatchTemplate </pre>	

Operation	Description
-----------	-------------

<pre> getAllByType() </pre>	<p>Returns a list of all TargetMatchTemplates where instance is set to false and Type is either "Action", "Environment", "Resource" or "Subject". This Type-parameter is used to execute a database query so only TargetMatchTemplates with AttributeDesignators of specific Type or AttributeSelectors are returned.</p>
-----------------------------	---

This can be used to prevent wrong types showing up in a specific case. For example if a user wants to add a TargetMatchTemplate to a ResourceTemplate, it is advised only those TargetMatchTemplates are listed that have elements (for example an AttributeDesignator) of the correct type (here: Resource).

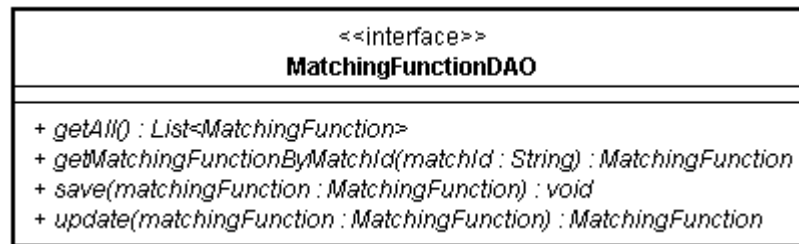
Target-
TemplateDAO

<<interface>> TargetTemplateDAO	
<pre> + getAll() : List<TargetTemplate> + getAllInstances() : List<TargetTemplate> + save(targetTemplate : TargetTemplate) : void + remove(targetTemplate : TargetTemplate) : void + update(targetTemplate : TargetTemplate) : TargetTemplate + clone(targetTemplate : TargetTemplate, parents : Set<Template>, newName : String) : TargetTemplate + duplicate(targetTemplate : TargetTemplate, parents : Set<Template>, newName : String) : TargetTemplate + getById(id : int) : TargetTemplate </pre>	

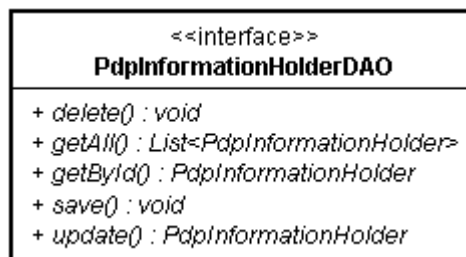
DataTypeDAO

<<interface>> DataTypeDAO	
<pre> + getAll() : List<DataType> + getDataTypeByDataTypeId(dataTypeId : String) : DataType + save(dataType : DataType) : void + update(dataType : DataType) : DataType </pre>	

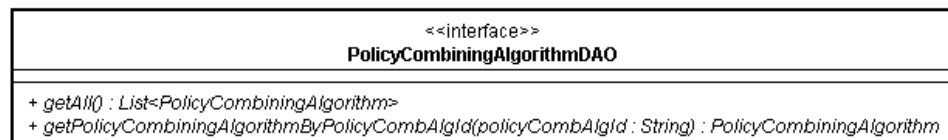
Matching-
FunctionDAO



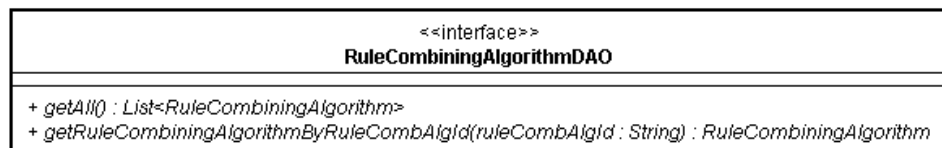
Pdp-
Information-
HolderDAO



Policy-
Combining-
AlgorithmDAO



Rule-
Combining-
AlgorithmDAO

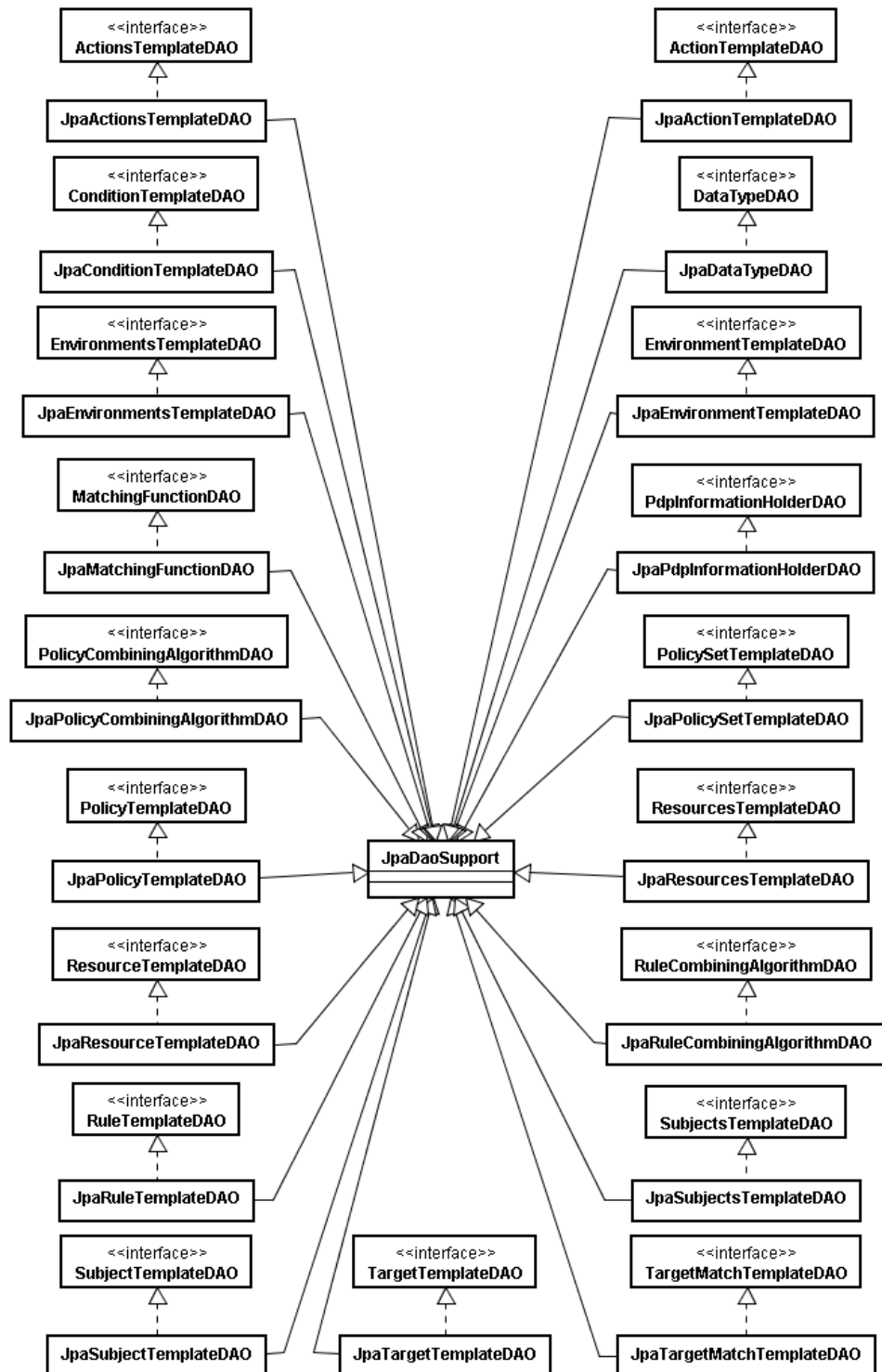


17.2 Implementation

Overview

This chapter covers the implementation of `org.herasaf.pap.dao` and `org.herasaf.pap.dao.jpaa`

Class diagram



Packages

org.herasaf.pap.dao

Contains all interfaces of the Data Access Objects (DAOs). An implementation of these interfaces is located in `org.herasaf.pap.dao.jpa`.

org.herasaf.pap.dao.jpa



Contains the JPA implementation of the DAO interfaces from the `org.herasaf.pap.dao` package.

These Implementations extend the class `JpaDaoSupport` of the Spring framework, a convenient super class for JPA data access objects, intended for `JpaTemplate` usage.

`JpaDaoSupport` offers a setter for a `javax.persistence.EntityManagerFactory` and creates implicitly a `JpaTemplate` instance that can be used in our implementation by using the method `getJpaTemplate()`. Thus no explicit configuration of the JPA template is necessary.

17.3 Design decisions

Overview This chapter explains the most important design decisions made in the `herasaf-pap-interfaces-dao` module.

Interfaces for DAOs Interfaces are provided which define methods to access the database. The concrete implementation can therefore be chosen by the developer.

Discussion of our solution

By providing these interfaces, the concrete implementation how to access the database can be switches without changing dependent modules that use DAOs to access the database. This increases maintainability and changeability as it is possible to change the database engine without affecting other modules

Replication Replication is a teamwork operation of the domain and DAO layer. Replication of templates itself is done in the domain whereas the DAO layer is wiring the parents.

Discussion of our solution

This was necessary due to a subtlety concerning managed objects and the entity manager. The template to be replicated gets passed to the `clone()` or `duplicate()` method in the DAO layer and is being cloned or duplicated. This cloning or duplicating only concerns the template itself and child templates.

Then the cloned or duplicated template is merged into the database whereas through `CascadeType.MERGE`, the child template do not get saved into the database again. This template is now a managed entity and received an id from the entity manager.

Wiring the parents is then done by getting every entity directly from the database, thus these parents are managed by now, and adding the still-managed replicated template as a child.

Why wiring the parents does not work when done in the domain

If wiring of the parents was done in the domain, the replicated template *with id 0* is wired to a parent template, that is already persisted in the database and thus this parent template does reference a child template with id 0.

Due to this fact, an exception is thrown when trying to save this replicated template.



No delete

`DataTypeDAO`, `MatchingFunctionDAO`, `PolicyCombiningAlgorithmDAO`, `RuleCombiningAlgorithmDAO` have no functions to delete.

Discussion of our solution

To stay consistent regarding the HERAS^{AF} template model representation of those four artifacts and the HERAS^{AF} XACML 2.0 core, we decided not to provide erasing functionality.

18 herasaf-pap-icefaces-service

Overview This chapter covers the package `org.herasaf.pap.service`.

The service layer is used to expose methods for other modules. In our case, these service methods always call methods on DAOs.

18.1 Interfaces

Overview This chapter explains the interfaces inside the service module.

18.1.1 Interface `TemplateManagement`

Description The interface `TemplateManagement` contains operations that can be executed on `TemplatesDAOs`.

UML



Operation	Description
<code>clone()</code>	Clones a Template.



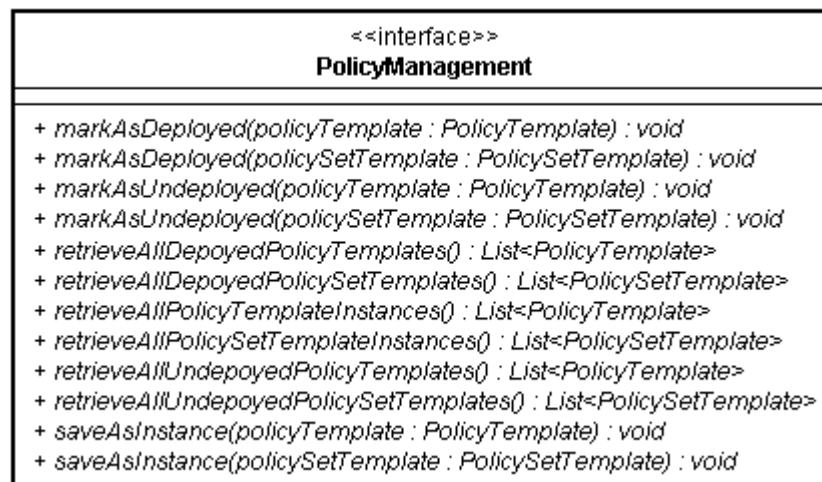
<code>delete()</code>	Deletes a <code>Template</code>
<code>deleteWith-Children()</code>	Recursively deletes either a <code>PolicyTemplate</code> or <code>PolicySetTemplate</code> with all its children.
<code>get()</code>	Gets a given <code>Template</code> from the database.
<code>retrieve-XXX()</code>	Retrieves all of the <code>XXXTemplates</code> from the database
<code>store()</code>	Stores a <code>Template</code> in the database

All methods above delegate to their Data Access Object. Refer to 17.1 (page 62, herasaf-pap-icefaces-dao *Interfaces*) for further details.

18.1.2 Interface `PolicyManagement`

Description The interface `PolicyManagement` contains operations that can be executed on `PolicyTemplates` or `PolicySetTemplates` data access objects.

UML



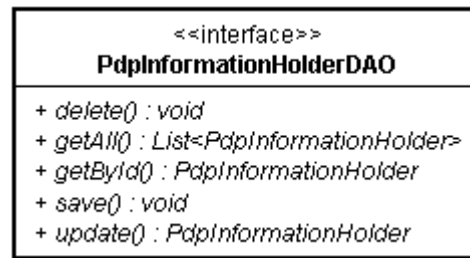
Operation	Description
any	All methods above delegate to their data access object. Refer to 17.1 (pages 62, herasaf-pap-icefaces-dao <i>Interfaces</i>) for further details.

18.1.3 Interface `PdpManagement`

Description The interface `PdpManagement` contains operations that can be executed on `PdpInformationHolder` DAOs.



UML



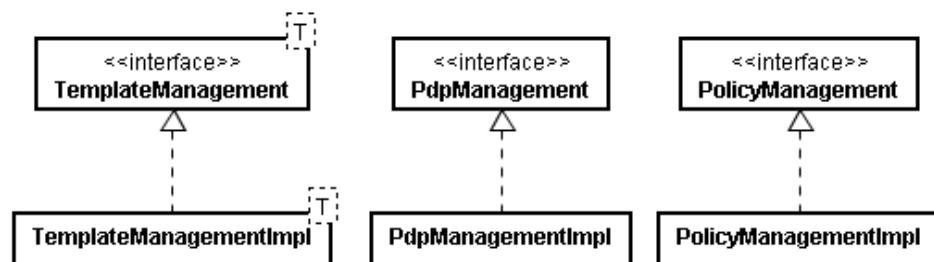
Operation	Description
any	All methods above delegate to their data access object. Refer to 17.1 (pages 62, herasaf-pap-icefaces-dao - Interfaces) for further details.

18.2 Implementation

Overview

This chapter covers the implementation of `org.herasaf.pap.service.templates` and `org.herasaf.pap.service.templates.impl`

Class diagram



Packages

org.herasaf.pap.service.templates

Contains all interfaces for accessing the Data Access Objects (DAOs). An implementation of these is located in `org.herasaf.pap.service.templates.impl`. Additionally, transactional attributes have been added to the service layer and thus these interface definitions.

org.herasaf.pap.service.templates.impl

Contains the implementation of the interfaces from the `org.herasaf.pap.service.templates` package. These Implementations delegate to their corresponding DAO to fulfill the method request.

18.3 Design decisions

Overview

This chapter shall give insights on the design decisions we took while extending and refactoring this interface.

There has been one bigger refactoring in the `herasaf-pap-icefaces-service` module that needs some explanation.



*Store and
delete use
Templates as
arguments*

In the service layer, we only have one `store()` and one `delete()` method to save or delete templates from the database. The problem at hand with this solution is to find the correct `TemplateDao` if only a template class is given.

Discussion of our solution

We introduced a typed map to address the problem mentioned above. By using the class-object as key for the map and the corresponding `TemplateDao` as the value object, we exactly gain the information we need.

Whenever `store()` or `delete()` is invoked, we use the `get()` operation of the `java.util.Map` to obtain the correct reference to a corresponding `TemplateDao` injected by Spring.

The traditional approach would have been to introduce a `store()` and `delete()` method for every template itself. We rejected this solution as it is neither flexible nor maintainable. Additionally, the only reason for introducing all these methods is exactly because there is a different `TemplateDao` to be used for every template.

19 herasaf-pap-icefaces-web

Overview This chapter explains the web module.

The web module is the front end of the HERAS^{AF}-PAP implementation. The administrator (either *BAdmin* or *TAdmin*) can create, manage, delete and replicate `Templates` visually.

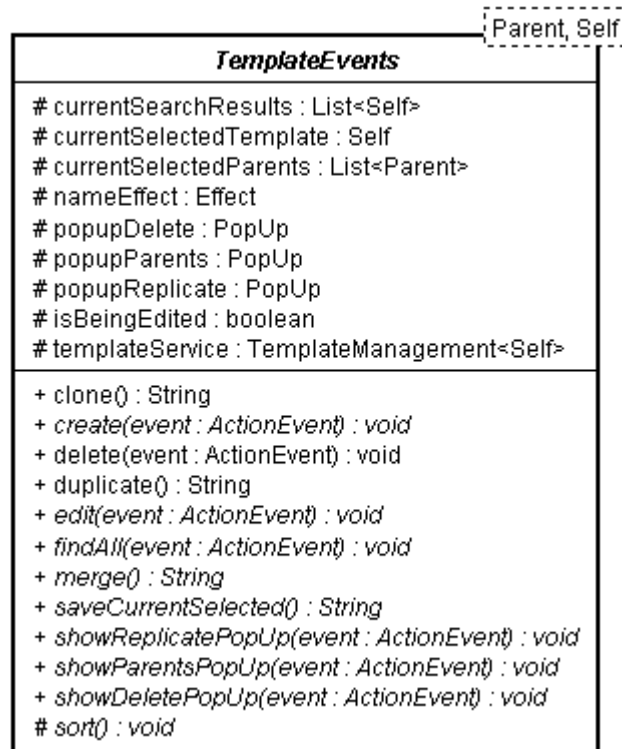
The module allows an *BAdmin* to deploy a created and instantiated `PolicyTemplates` and `PolicySetTemplates`.

19.1 Interfaces

19.1.1 Abstract class `TemplateEvents`

Description The abstract class `TemplateEvents` contains operations that have to be implemented in all template backing beans and defines certain behaviors itself. It extends the abstract class `SortableTable`, described in 19.1.2.

UML



Attribute	Description
currentSearchResults	A List of type <code><Self></code> . So every <code>xxxTemplateSessionController</code> holds a List for its <code>xxxTemplate</code> . Usually all existing <code>Templates</code> of



	<code>type Self</code> are stored in this list.
<code>current-Selected-Template</code>	Represent the <code>currentSelectedTemplate</code> . As every template backing bean (<code>xxxTemplateSessionController</code>) is responsible for its underlying <code>xxxTemplate</code> , the currently selected <code>Template</code> is stored in this variable.
<code>current-Selected-Parents</code>	In a <code>replicatePopUp</code> lists of parents are shown. They can be selected or deselected. This is handled by using this attribute.
<code>nameEffect</code>	The ICEfaces <code>Effect</code> that will appear in the web user interface when no name is entered while creating a template.
<code>popupDelete</code>	The delete <code>PopUp</code> . This <code>PopUp</code> is shown when the delete link is clicked on any overview page of a template in the web user interface.
<code>popupParents</code>	The parents <code>PopUp</code> . This <code>PopUp</code> is shown when the show-parents-link is clicked on any overview page of a template in the web user interface.
<code>popupReplicate</code>	The replication <code>PopUp</code> . This <code>PopUp</code> is shown when the <i>replicate</i> link is clicked on any overview page of a template in the web user interface. Additionally this <code>PopUp</code> is shown when the save button is clicked if a user modified (=edited) a previously created template that is already used by some parent templates, because editing would affect that parents too.
<code>isBeing-Edited</code>	Indicates if the current <code>Template</code> in a page is edited or not. If it is not edited, it has to be in a process of creation. This boolean is used to show different titles in the web page where a template can be created or edited. As of the current architecture, the web page used to edit a <code>Template</code> or to create a template is the same, thus we need to distinguish if we are editing or creating a template.
<code>template-Service</code>	Requests to the service layer and thus the database can be made using the <code>templateService</code> .
Operation	Description
<code>clone()</code>	Executes a cloning on the <code>currentSelectedTemplate</code> . This method is usually called via the <code>replicationPopUp</code> , when the <code>clone</code> -button is pressed.
<code>create()</code>	This method is executed every time a user clicks in the



	web front end on a “create <i>xxxTemplate</i> ” button.
<code>delete()</code>	Deletes the <i>Template</i> . Is usually executed when the delete link on a <i>Template</i> in the <code>popupDelete</code> is pressed. It deletes the <i>Template</i> from the database, executes a <code>findAll()</code> and closes the <code>popupDelete</code> .
<code>duplicate()</code>	Executes a duplication on the <code>currentSelectedTemplate</code> . It delegates the method call to the service layer, clears the <i>List</i> <code>currentSelectedParents</code> and finally executes a <code>findAll()</code> . This method is normally called via the <code>replicationPopUp</code> , when the <code>duplicate</code> -button is pressed.
<code>edit()</code>	When an <code>edit</code> link is clicked, this method is executed. Generally said this method refreshes the selected <i>Template</i> from the database and thus allows modification of the <i>Template</i> .
<code>findAll()</code>	Retrieves all corresponding <i>Templates</i> from the database.
<code>merge()</code>	This method is called when a user is editing a template that has already been used by parents and tries to save it. The <code>replicationPopUp</code> is shown. When the “Save” button is clicked, this method is called on the corresponding bean.
<code>saveCurrent-Selected()</code>	This method is called when a “Save <i>xxxTemplate</i> ” link is clicked. It saves the <code>currentSelectedTemplate</code> in the database.
<code>showDelete-PopUp()</code>	Shows the <code>deletePopUp</code> . Has to be implemented by every subclass of <code>TemplateEvent</code> .
<code>showParents-PopUp()</code>	Shows the <code>deleteParentsUp</code> . Has to be implemented by every subclass of <code>TemplateEvent</code> .
<code>show-Replicate-PopUp()</code>	Shows the <code>deleteReplicateUp</code> . Has to be implemented by every subclass of <code>TemplateEvent</code> .
<code>sort()</code>	Has to be implemented by every subclass. Sorts the <i>Collection</i> <code>currentSearchResults</code> .

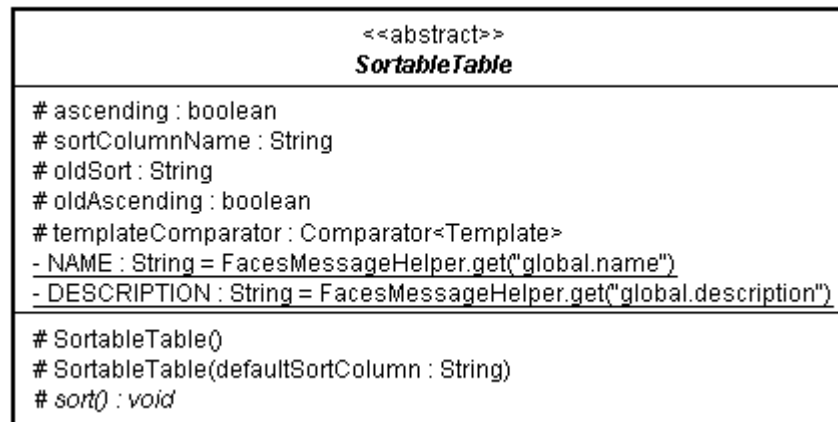
19.1.2 Abstract class `SortableTable`

Description

The abstract class `SortableTable` handles sorting of a table. In the web user interface, any table that lists templates can be sorted by its name or its description.



UML



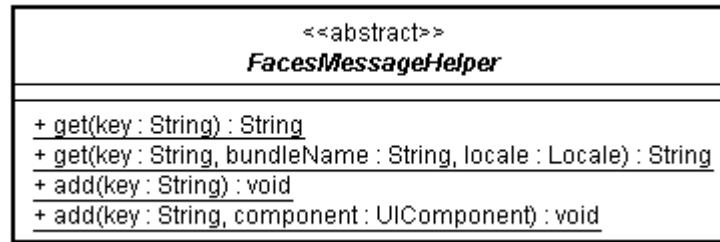
Attribute	Description
ascending	Indicates whether to sort ascending or not.
oldAscending	The ICEfaces <code>Effect</code> that will appear in the web user interface when no name is entered while creating a template.
oldSort	The old sorted column name.
sortColumnName	The current sorted column name.
templateComparator	A <code>Comparator</code> which sorts <code>Templates</code> by their name or description (depends on <code>sortColumnName</code>) ascending or descending (depends on <code>ascending</code>).
NAME	The name of the name column. Is set accordingly to the entry <code>global.name</code> in the language property file accessed via <code>FacesMessageHelper</code> .
DESCRIPTION	The name of the description column. Is set accordingly to the entry <code>global.description</code> in the language property file accessed via <code>FacesMessageHelper</code> .
Operation	Description
sort()	Has to be implemented by all subclasses.

19.1.3 Abstract class `FacesMessageHelper`

Description

The abstract class `FacesMessageHelper` enables access to the internationalized message properties file. Thus internationalized error handling can be achieved in the backing beans or internationalized error messages can be redirected to the web pages.

UML



Operation	Description
<code>get (String)</code>	Returns a localized key from the message bundle that is registered with the JSF <i>Application</i> .
<code>get (String, String, Locale)</code>	Returns a localized key from a given message bundle and a given <i>Locale</i> .
<code>add (String)</code>	Localizes a text and adds it as a global <i>FacesMessage</i> to the message queue of the actual <i>FacesContext</i> .
<code>add (String, UIComponent)</code>	Localizes a text and adds it as a <i>FacesMessage</i> to the specified <i>UIComponent</i> . The <i>Locale</i> is automatically determined. The resource bundle declared in the <code>faces-config.xml</code> is used.

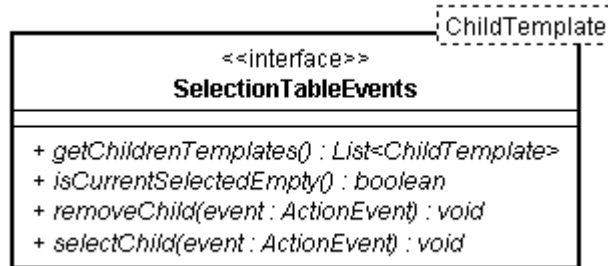
19.1.4 Interface `SelectionTableEvents`

Description

The interface `SelectionTableEvents` contains operations that have to be implemented in backing beans that only have one type of children template and one type of parent template. This applies to `ActionTemplate`, `ActionsTemplate`, `EnvironmentTemplate`, `EnvironnementsTemplate`, `ResourceTemplate`, `ResourcesTemplate`, `SubjectTempalte` and `SubjectsTemplate`. All other templates either have no children templates, like `ConditionTemplate` or `TargetMatchTemplate`, or have multiple parents templates of different type.

Summed up, only templates with n:n mapping to one type of parent template and n:n mapping to one type of children template should implement this interface.

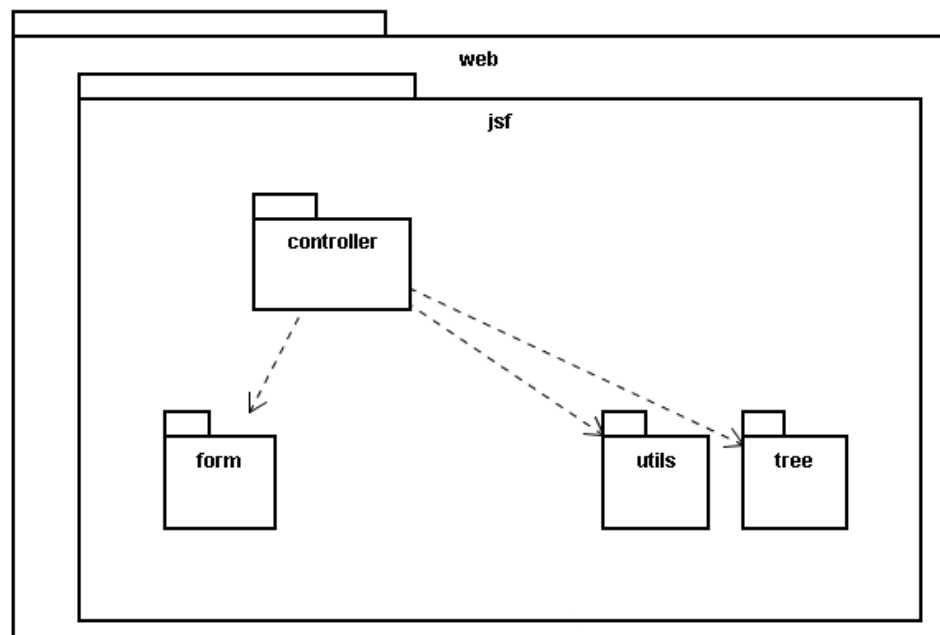
UML



Operation	Description
getChildren- Templates()	Returns a list of all existing children template of the current Template. Is displayed in the “Available xxxTemplate” table.
isCurrent- Selected- Empty()	Returns true if currently any children templates are listed in the “Selected xxxTemplate” table.
remove- Child()	Removes a child from the children list of the current selected template.
select- Child()	Adds a child to the children list of the current selected template.

19.2 Implementation

Package
Diagram



Description

jsf

The main web package. Contains all following packages:

controller

This package contains all backing beans.

form

This package contains forms for the business administrator.

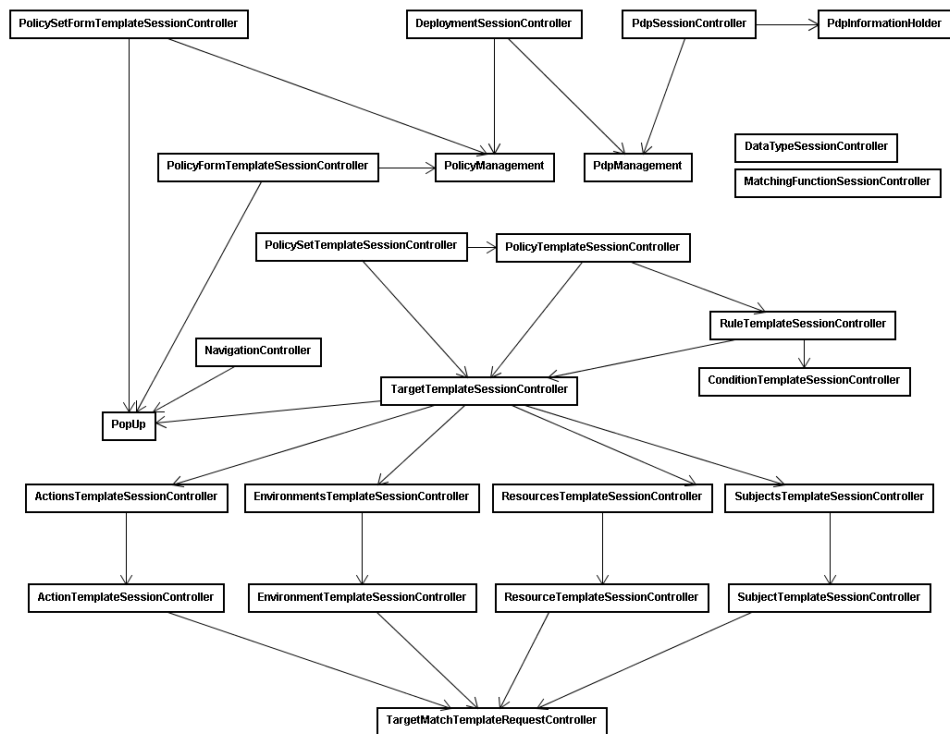
tree

This package helps to build a ICEfaces tree, for example the condition tree.

utils

This package contains utilities, like the `FacesMessageHelper` or the `PopUp`.

Class diagram



Super classes, interfaces and generics were removed due to overview reasons.

19.2.1 Facelets

Overview

This chapter explains the composition components created with facelets technology. First general explanations are given, then the actual implemented facelets tags are described.

Why use facelets?

Using the facelets technology helps avoiding repeated xhtml-code, especially when used in an environment where many elements use similar functions or



code. This is done by creating a facelets tagfile.

Also facelets enables separation of layout and logic.

How to enable facelets templating

- Create a tag library file
- Declare the tag library in web.xml
- Import the tagfile using namespace and use it

A simple Example

Tag minimal library file (e.g. herasaf.taglib.xml) has to be created:

```
<facelet-taglib>
  <namespace>http://www.herasaf.org/jsf</namespace>
  <tag>
    <tag-name>backButton</tag-name>
    <source>../path/to/button_back.xhtml</source>
  </tag>
</facelet-taglib>
```

In order to use the tag library, it has to be registered in the web.xml:

web.xml

```
<context-param>
  <param-name>facelets.LIBRARIES</param-name>
  <param-value>
    ../path/to/herasaf.taglib.xml
  </param-value>
</context-param>
```

button_back.xhtml renders a link. That link is widely used in the web project. Instead of writing always the same code, a composition was created.

button_back.xhtml

```
<html xmlns="http://www.w3.org/1999/xhtml"
  xmlns:ui="http://java.sun.com/jsf/facelets"
  xmlns:c="http://java.sun.com/jstl/core"
  xmlns:ice="http://www.icesoft.com/icefaces/component">

  <ui:composition>
    <c:if test="{empty link_text}">
      <c:set var="link_text" value="back" />
    </c:if>
    <ice:form>
      <ice:commandLink styleClass="button"
        value="{link_text}" action="back" />
      <ui:insert />
    </ice:form>
  </ui:composition>
</html>
```



Now the tagfile can be used in a xhtml file. First the xml namespace *herasaf* is defined. Thus the custom tag library can be used.

```
<html>
  <f:view
    xmlns:f=http://java.sun.com/jsf/core
    xmlns:herasaf="http://www.herasaf.org/jsf">
    <head />
    <body>
      <herasaf:backButton />
    </body>
  </f:view>
</html>
```

The empty element ui:insert

An empty element `<ui:insert />` offers easy integration of further elements.

As an example, we can add more elements to the back-button mentioned above. Instead of just using

```
<herasaf:backButton />
```

following construct may be used

```
<herasaf:backButton>
  <ice:outputText value="Added text after button" />
  <ice:outputText value="Even more text" />
</herasaf:backButton>
```

Passing arguments

As the usage of facelets composition resembles using functions in a java program, parameters can be passed to these compositions too

For example, we can modify above example so the link displays another text by passing it.

```
<herasaf:backButton>
  button_text="New button text" />
```

This idea has been implemented for all repeatedly appearing elements in the PAP web user interface. Mostly the name of the backing bean has to be passed, in order to invoke backing bean functions dynamically.

Conclusion

By using above approaches, no duplicated code has to be written and the DRY-principle is respected. Of course this positively impacts the maintainability of the code. There is just one place to perform changes.

By extending the composition with JSTL-capabilities (like `if`, `if else`), distinction of cases is fulfilled. The developer can check if parameters have been passed and initialize them to default values or decides which part of code is rendered and thus displayed to the user.

Existing facelets templates

Following facelets templates exist.



“Extendable” means if there exists an empty `ui:insert`. In that case more elements can be added.

19.2.1.1 Tag `backButton`

Overview Renders the back button. Is used nearly on every existing page.

Location `backButton` is defined in `button_back.xhtml`.

<i>Details</i>	Tag name	<code>backButton</code>
	Parameters	None
	Extendable	Yes, the passed elements are inserted directly after the back-button link.
	Requirements	None

19.2.1.2 Tag `createButton`

Overview Renders a create button.

Location `createButton` is defined in `button_create.xhtml`.

<i>Details</i>	Tag name	<code>createButton</code>
	Mandatory parameters	<code>msgsCreate</code> the value of the link
		<code>backingBean</code> The backing bean.
	Optional Parameters	None
	Extendable	No
	Requirements	The backing bean must have the method <code>create(ActionEvent)</code> .

19.2.1.3 Tag `deletePopup`

Overview This popup is rendered when the delete link is pressed on a template in the *overview* page.

Location `deletePopup` is defined in `popup_delete.xhtml`.



<i>Details</i>	Tag name	deletePopup	
	Mandatory parameters	label	The label of the ICEfaces panel popup
		backingBean	The backing bean.
		title	The title of the popup.
		parentsText	The text explaining which parents still reference this element.
		question	The text asking if the user is sure about his action.
	Optional parameters	showSecondDegreeParents	<p>default:false</p> <p>If set to true, the <code>backingBean</code> needs to have the property <code>secondDegreeParents</code>. This possibility was introduced because there does not exist any <code>TargetTemplate</code> in the web user interface. A <code>TargetTemplate</code> is created implicitly and not visible for the user.</p> <p>For example the immediate parent of a <code>ActionsTemplate</code> is a <code>TargetTemplate</code>, but as it cannot be displayed, we need the parent of the <code>TargetTemplate</code>, for example a <code>PolicyTemplate</code>, to be listed. In this case we need this method.</p>
	Extendable	No	
	Requirements	The backing bean must have a property <code>popupDelete</code> of type <code>PopUp</code> , a property <code>currentSelectedParents</code> of Type <code>List<ParentsTemplateType></code> , and a method <code>delete(ActionEvent)</code> .	

19.2.1.4 Tag parentsPopup

Overview This popup is rendered when the “view parents” link is pressed on a template.

Location `parentsPopup` is defined in `popup_parents.xhtml`.

<i>Details</i>	Tag name	parentsPopup	
	Mandatory parameters	label	The label of the ICEfaces panel popup.
		backingBean	The backing bean.
	Optional parameters	showSecondDegreeParents	<p>default:false</p> <p>If set to true, the <code>backingBean</code> needs to</p>

	s	<p>have the property <code>secondDegreeParents</code>.</p> <p>This possibility was introduced because there does not exist any <code>TargetTemplate</code> in the web user interface. A <code>TargetTemplate</code> is created implicitly and not visible for the user.</p> <p>For example the immediate parent of a <code>ActionsTemplate</code> is a <code>TargetTemplate</code>, but as it cannot be displayed, we need the parent of the <code>TargetTemplate</code>, for example a <code>PolicyTemplate</code>, to be listed. Thus we need this method.</p>
Extendable	No	
Requirements	<p>The backing bean must have a property <code>popupParents</code> of type <code>PopUp</code>, a property <code>currentSelectedParents</code> or <code>secondDegreeParents</code> (depends if <code>showSecondDegreeParents</code> is true or false).</p>	

19.2.1.5 Tag replicationPopup

Overview This popup is rendered when the replicate link is pressed on a template or when a template with parents referring to it is edited.

Location replicationPopup is defined in `popup_replicate.xhtml`.

<i>Details</i>	Tag name	<code>replicationPopup</code>	
	Mandatory parameters	<code>label</code>	The label of the ICEfaces panel popup.
Optional parameters		<code>backingBean</code>	The backing bean.
		<code>showSaveButton</code>	Default: true If set to false, the save link is not displayed.
		<code>showParents</code>	Default: true If set to false, the parents are not displayed.
		<code>showSecondDegreeParents</code>	default:false If set to true, the <code>backingBean</code> needs to have the property <code>secondDegreeParents</code> . This possibility was introduced because there does not exist any

`TargetTemplate` in the web user interface. A `TargetTemplate` is created implicitly and not visible for the user.

For example the immediate parent of a `ActionsTemplate` is a `TargetTemplate`, but as it cannot be displayed, we need the parent of the `TargetTemplate`, for example a `PolicyTemplate`, to be listed. In this case we need this method.

Extendable	No
Requirements	The backing bean must have a property <code>popupReplicate</code> of type <code>PopUp</code> , a property <code>currentSelectedParents</code> of Type <code>List<ParentsTemplateType></code> , the methods <code>merge</code> , <code>clone</code> , <code>duplicate</code> , and <code>selectParentForReplication</code> .

19.2.1.6 Tag `templateCreateContent`

Overview This component renders the often used “selected/available”-tables. It uses the `<herasaf:templateTable>`

Location `templateCreateContent` is defined in `template_create_content.xhtml`.

<i>Details</i>	Tag name	<code>templateCreateContent</code>											
	Mandatory parameters	<table> <tr> <td><code>label</code></td> <td>The label of the ICEfaces panel pop up.</td> </tr> <tr> <td><code>backingBean</code></td> <td>The backing bean.</td> </tr> <tr> <td><code>msgsSelected</code></td> <td>The title above the table containing the selected templates.</td> </tr> <tr> <td><code>msgsNochildSelected</code></td> <td>The message indicating that no children have been selected yet.</td> </tr> <tr> <td><code>msgsAvailable</code></td> <td>The title above the table containing all children templates.</td> </tr> <tr> <td><code>backingBeanChild</code></td> <td>The backing bean of the current child. Every Template has children, except for <code>TargetMatchTemplate</code> and <code>ConditionTemplate</code>.</td> </tr> </table>	<code>label</code>	The label of the ICEfaces panel pop up.	<code>backingBean</code>	The backing bean.	<code>msgsSelected</code>	The title above the table containing the selected templates.	<code>msgsNochildSelected</code>	The message indicating that no children have been selected yet.	<code>msgsAvailable</code>	The title above the table containing all children templates.	<code>backingBeanChild</code>
<code>label</code>	The label of the ICEfaces panel pop up.												
<code>backingBean</code>	The backing bean.												
<code>msgsSelected</code>	The title above the table containing the selected templates.												
<code>msgsNochildSelected</code>	The message indicating that no children have been selected yet.												
<code>msgsAvailable</code>	The title above the table containing all children templates.												
<code>backingBeanChild</code>	The backing bean of the current child. Every Template has children, except for <code>TargetMatchTemplate</code> and <code>ConditionTemplate</code> .												
	Optional parameters	<table> <tr> <td><code>currentSearchResults</code></td> <td>Default: <code>currentSearchResults</code> Should point to a Collection of all available templates of the specific type, i.e. the <code>ActionsTemplate</code> backing bean should hold a property of type <code>List<ActionsTemplate></code>.</td> </tr> </table>	<code>currentSearchResults</code>	Default: <code>currentSearchResults</code> Should point to a Collection of all available templates of the specific type, i.e. the <code>ActionsTemplate</code> backing bean should hold a property of type <code>List<ActionsTemplate></code> .									
<code>currentSearchResults</code>	Default: <code>currentSearchResults</code> Should point to a Collection of all available templates of the specific type, i.e. the <code>ActionsTemplate</code> backing bean should hold a property of type <code>List<ActionsTemplate></code> .												

<code>currentSelectedEmpty</code>	Default: <code>isCurrentSelectedEmpty</code> Indicates if the actual Template has children added.
<code>childrenTemplates</code>	Default: <code>childrenTemplates</code> A property holding a List of children templates.
<code>removeChild</code>	Default: <code>removeChild</code> Pointing to the function that removes children templates from the current selected template.
<code>selectChild</code>	Default: <code>selectChild</code> Point to the function that adds children to the current selected template.
Extendable	No
Requirements	The backing bean must have a property <code>currentSearchResults</code> , the methods <code>isCurrentSelectedEmpty</code> , <code>removeChild(ActionEvent)</code> and <code>selectChild(ActionEvent)</code> .

19.2.1.7 Tag `templateCreateFooter`

Overview This composition is used on the create pages of `Subject(s)Template`, `Action(s)Template`, `Environment(s)Template` and `Resource(s)Template`. It includes a “back”, a “manage” and a “save” link.

Location `templateCreateFooter` is defined in `template_create_footer.xhtml`.

<i>Details</i>	Tag name	<code>templateCreateFooter</code>								
	Mandatory parameters	<table> <tr> <td><code>backingBean</code></td> <td>The backing bean.</td> </tr> <tr> <td><code>msgsSave</code></td> <td>The text on the save link.</td> </tr> <tr> <td><code>backingBeanChild</code></td> <td>The backing bean of the child.</td> </tr> <tr> <td><code>msgsManage</code></td> <td>The text on the manage link.</td> </tr> </table>	<code>backingBean</code>	The backing bean.	<code>msgsSave</code>	The text on the save link.	<code>backingBeanChild</code>	The backing bean of the child.	<code>msgsManage</code>	The text on the manage link.
<code>backingBean</code>	The backing bean.									
<code>msgsSave</code>	The text on the save link.									
<code>backingBeanChild</code>	The backing bean of the child.									
<code>msgsManage</code>	The text on the manage link.									
	Extendable	No								
	Requirements	The backing bean must have a method <code>saveCurrentSelected()</code> . The backing bean child must have the method <code>findAll(ActionEvent)</code> .								

19.2.1.8 Tag `templateCreateHeader`

Overview This composition is used by all Templates. Every Template needs to have a name and may have a description in the creation process. It can be expanded with further elements.

Location `templateCreateHeader` is defined in `template_create_header.xhtml`.

<i>Details</i>	Tag name	<code>templateCreateHeader</code>	
	Mandatory parameters	<code>label</code>	The label of all ICEfaces components inside the composition.
		<code>backingBean</code>	The backing bean.
	Extendable	Yes. Further elements are inserted below the name and description fields.	
Requirements	The backing bean needs to have a property <code>currentSelectedTemplate</code> .		

19.2.1.9 Tag `templateOverviewOperations`

Overview This composition represents the operations that can be applied to any Template. Currently these operations are “edit”, “delete”, “replicate” and “View parents”.

Location `templateOverviewOperations` is defined in `template_operations_overview.xhtml`.

<i>Details</i>	Tag name	<code>templateOverviewOperations</code>	
	Mandatory parameters	<code>backingBean</code>	The backing bean.
		Extendable	No
	Requirements	The backing bean needs to have the methods <code>edit(ActionEvent)</code> , <code>showDeletePopup(ActionEvent)</code> , <code>showReplicatePopup(ActionEvent)</code> and <code>showParentsPopup(ActionEvent)</code> .	

19.2.1.10 Tag `templateTable`

Overview This composition is used by all Templates in their corresponding overview page. It is used by all select/remove tables existing in the creation process too. Basically it renders a sortable table that contains the name, description and children of an underlying template type. The last column is called “operations” and left empty, thus it can be expanded with custom elements. In the current



implementation this composition is either expanded with a `templateOverviewOperations` (described above) or with a `remove-` or `select-` operation from within a `templateCreateContent` (described above).

Location `templateTable` is defined in `template_table.xhtml`.

<i>Details</i>	Tag name	<code>templateTable</code>	
	Mandatory parameters	<code>label</code>	The label.
		<code>backingBean</code>	The backing bean.
	Optional parameters	<code>currentSearchResults</code>	Default: <code>currentSearchResults</code>
<code>chResults</code>		Can be overwritten if another search method of the <code>backingBean</code> is to be used than default.	
Extendable	Yes	Elements are inserted into the “operations” column.	
Requirements	The backing bean needs to have the properties <code>sortColumnName</code> , <code>ascending</code> , <code>NAME</code> and <code>DESCRIPTION</code> (or alternatively extend the abstract class <code>SortableTable</code>).		

19.2.2 Web page structure

Overview This chapter covers the structure of the website for the business and the technical administrator.

19.2.2.1 Technical administrator

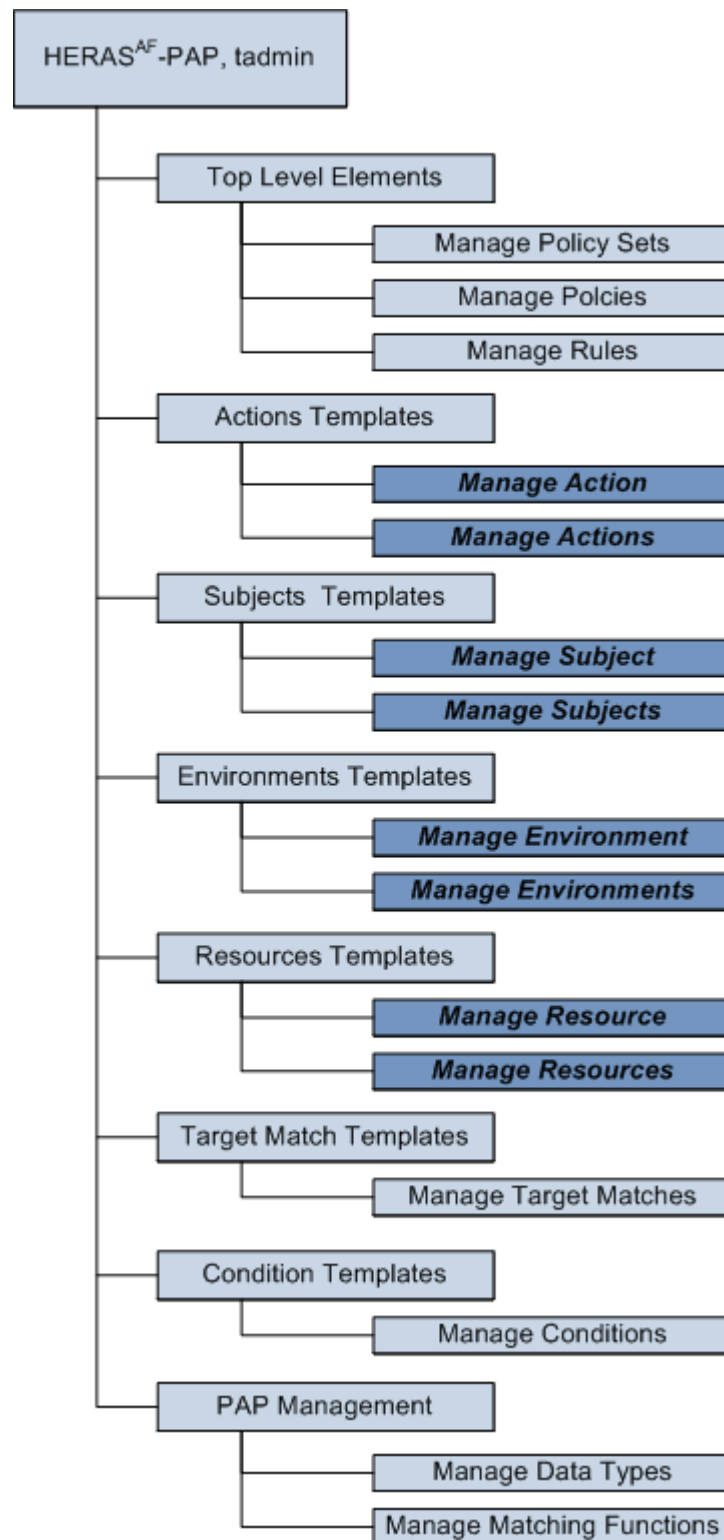
Overview This chapter covers the structure of the website for the technical administrator

Goals The structure of the web site focuses on:

- Hierarchical navigation structure
- Represents the structure of a XACML policy structure and thus navigation becomes more intuitive
- Quick access to often used elements



Sitemap



This picture visualizes the sitemap seen by the technical administrator. Links displayed on the start page after successful login are shown boldly and cursively.

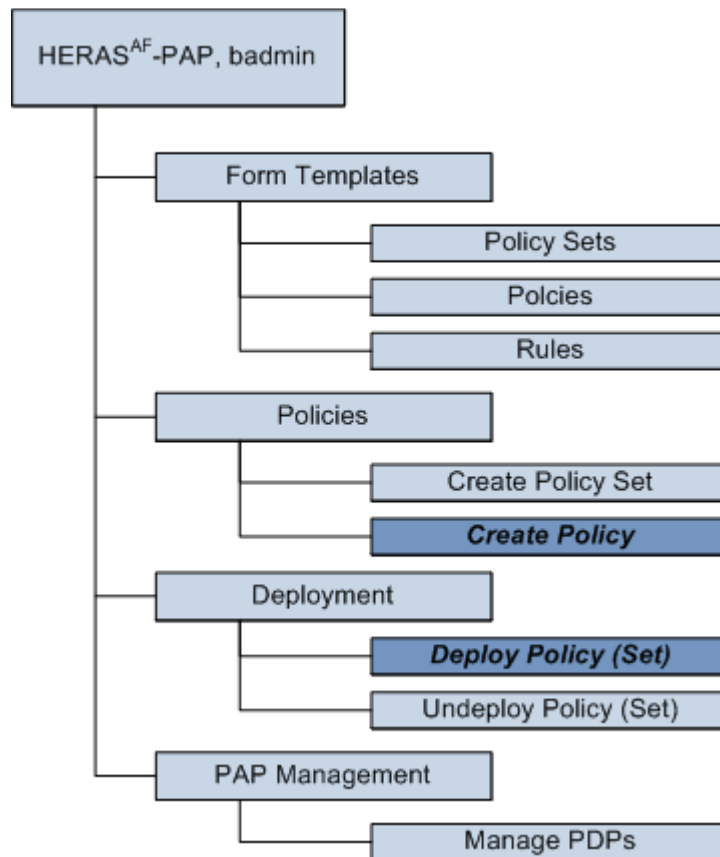
19.2.2.2 Business administrator

Overview This chapter covers the structure of the website for the business administrator.

Goals The structure of the web site focuses on:

- Hierarchical navigation structure
- Quick access to often used elements

Sitemap



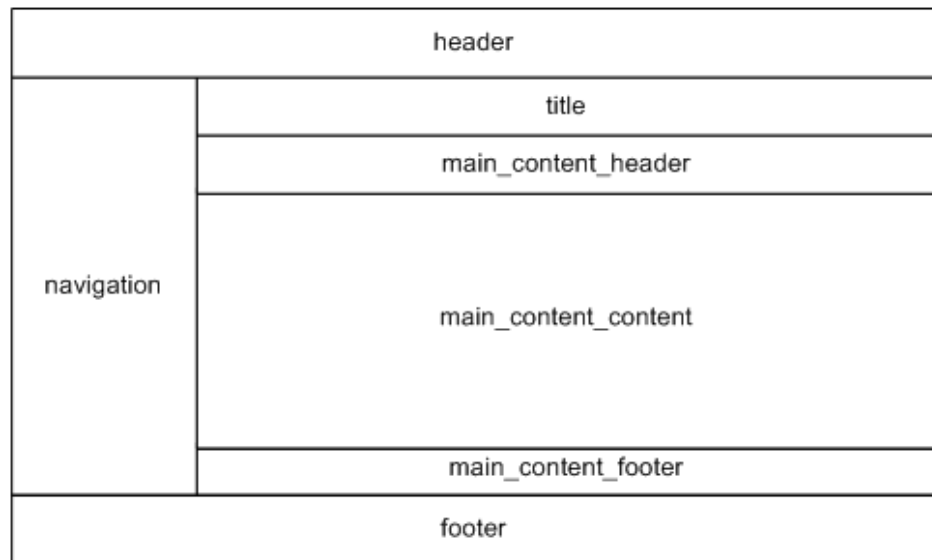
This picture visualizes the sitemap presented to the business administrator. Links displayed on the start page after successful login are shown boldly and cursively.

19.2.3 Xhtml Templates

Overview All web pages are designed by using composition and decoration of different web page templates

In this chapter, a template is not meant as a `Template` of the PAP domain, but is rather meant as a layout model.

*Illustration of
the web layout*



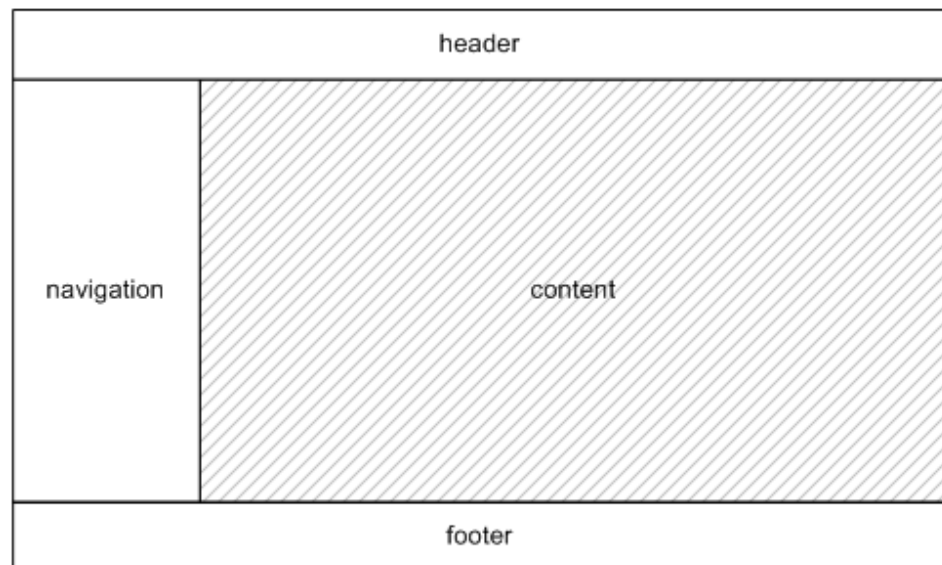
A global view at the layout and its composition.

19.2.3.1 master_layout.xhtml

Overview

The master_layout.xhtml defines the layout of the PAP. It is separated into multiple components. The component *header*, *navigation* and *footer* are static and remain at the same position on every page. The component *content* is dynamic and is used for different contents.

*Illustration of
master_layout.x
html*

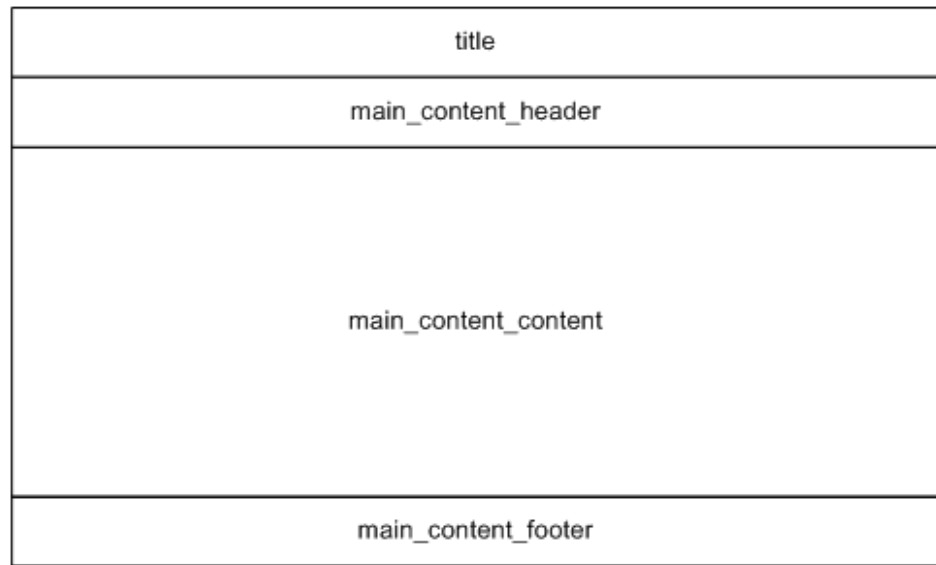


19.2.3.2 main_content_layout.xhtml

Overview

The *main_content_layout.xhtml* defines the layout of the content. It represents the *master_layout.xhtml content* component. Every page decorates this layout using its own elements.

Illustration of *main_content_l ayout.xhtml*



19.2.4 Web Flows

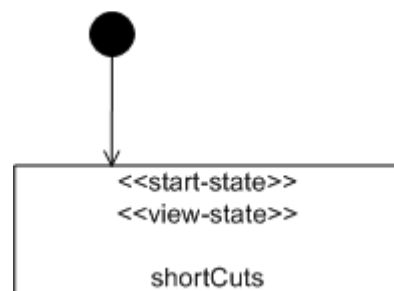
Overview

This chapter covers all Spring Web Flows of the HERAS^{AF} PAP web project. Using Spring Web Flow simplifies the handling of web application page flow.

19.2.4.1 Main flow

Overview

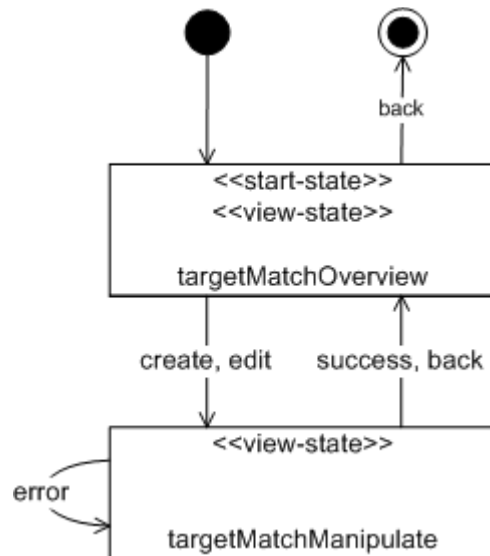
The main flow is the initial flow, it just displays the start page (after a successful login).



19.2.4.2 Target Match Template flow

Overview The *Target Match Template* flow is responsible for listing, creating or editing `TargetMatchTemplate`s.

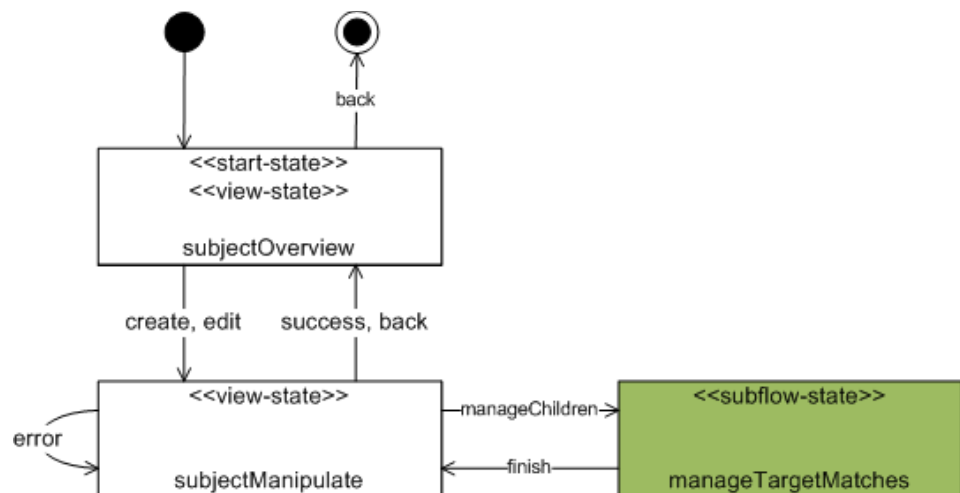
Target match flow diagram



19.2.4.3 Subject Template flow

Overview The *Subject Template* flow is responsible for listing, creating or editing `SubjectTemplate`s.

Subject flow diagram

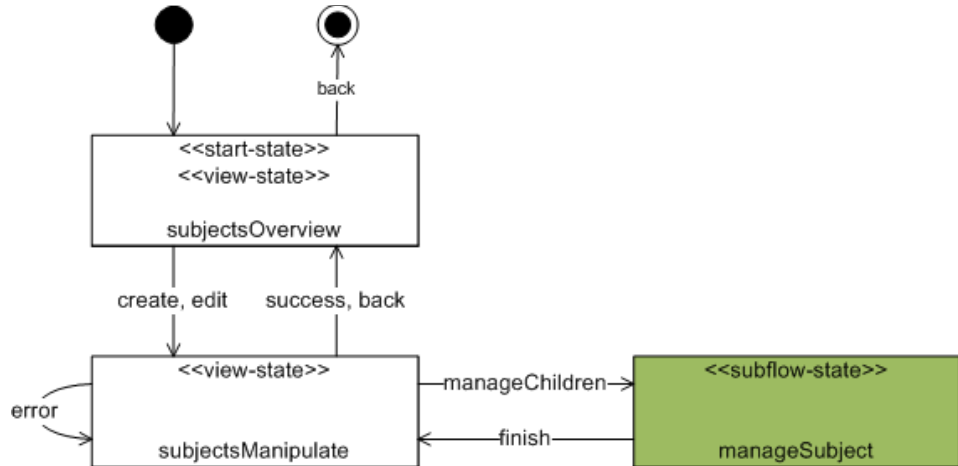


19.2.4.4 Subjects Template flow

Overview The *Subjects Template* flow is responsible for listing, creating or editing

SubjectsTemplates.

Subjects flow diagram

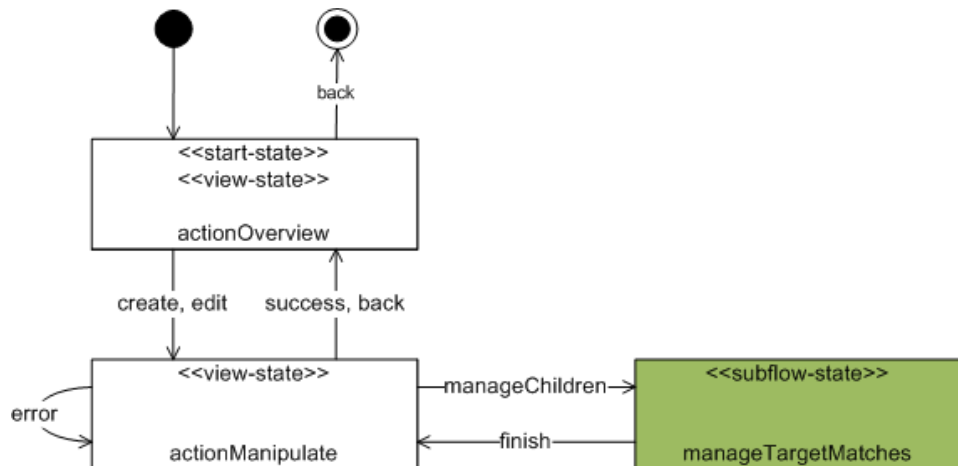


19.2.4.5 Action Template flow

Overview

The *Action Template flow* is responsible for listing, creating or editing `ActionTemplates`.

Action flow diagram

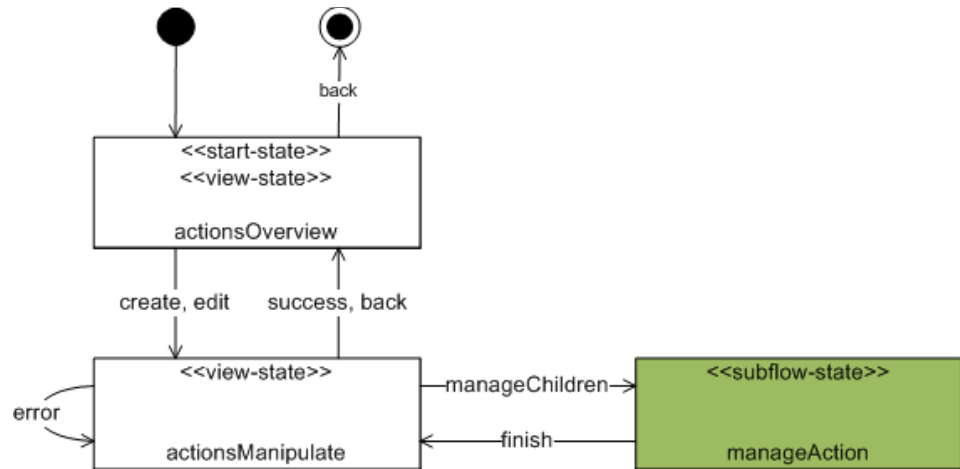


19.2.4.6 Actions Template flow

Overview

The *Actions Template flow* is responsible for listing, creating or editing `ActionsTemplates`.

Actions flow diagram

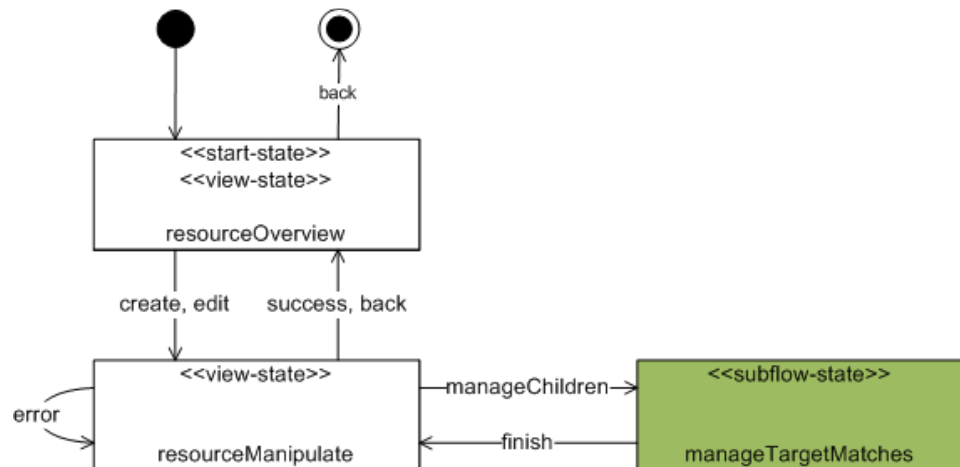


19.2.4.7 Resource Template flow

Overview

The *Resource Template* flow is responsible for listing, creating or editing `ResourceTemplateS`.

Resource flow diagram

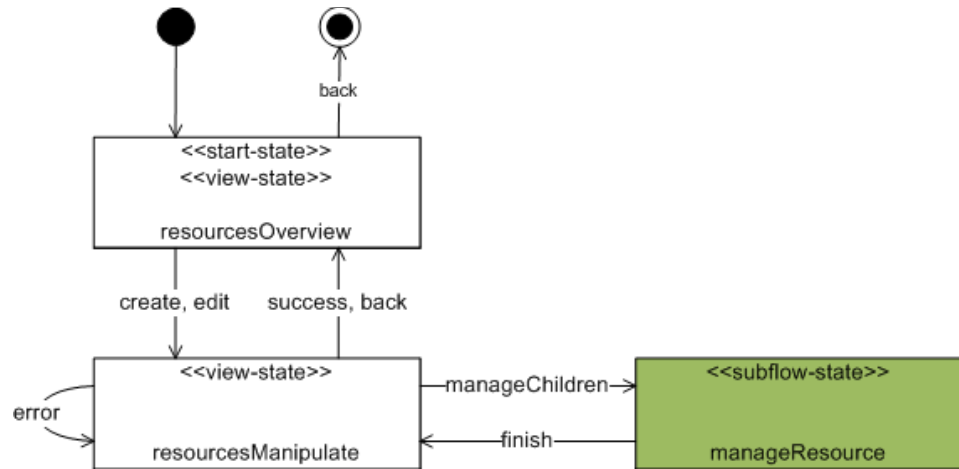


19.2.4.8 Resources Template flow

Overview

The *Resources Template* flow is responsible for listing, creating or editing `ResourcesTemplateS`.

Resources flow diagram

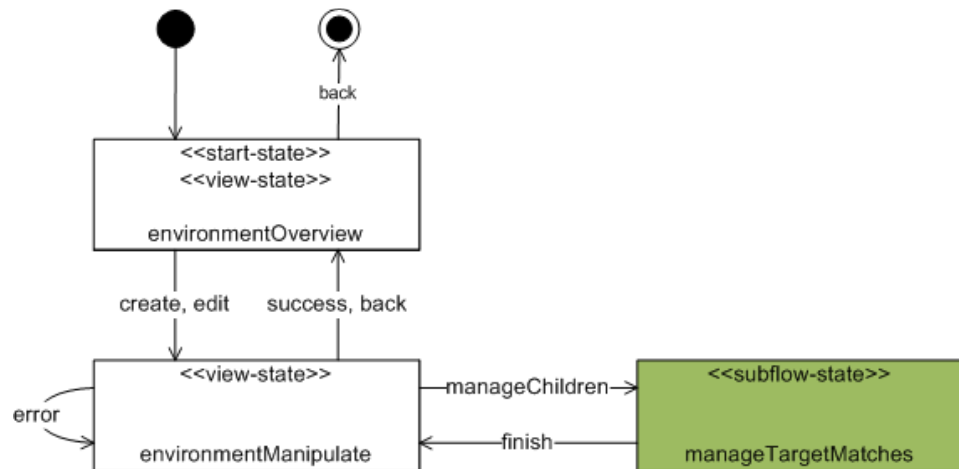


19.2.4.9 Environment Template flow

Overview

The *Environment Template* flow is responsible for listing, creating or editing `EnvironmentTemplateS`.

Environment flow diagram

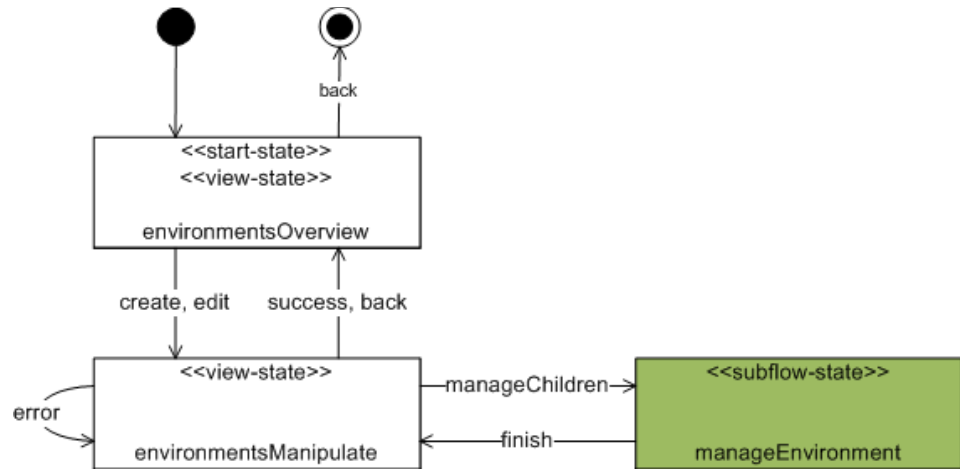


19.2.4.10 Environments Template flow

Overview

The *Environments Template* flow is responsible for listing, creating or editing `EnvironmentsTemplateS`.

Environments
flow diagram

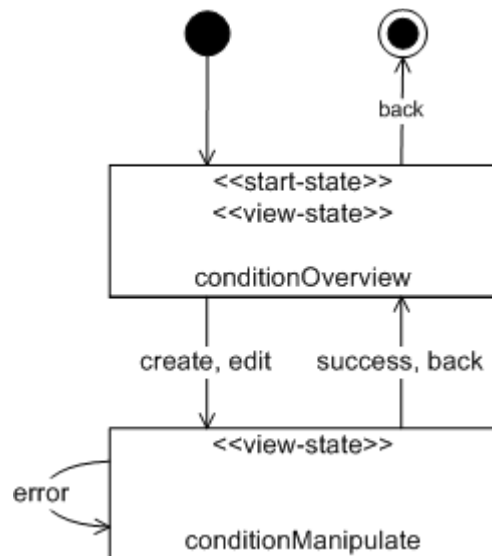


19.2.4.11 Condition Template flow

Overview

The *Condition Template flow* is responsible for listing, creating or editing `ConditionTemplates`.

Condition flow
diagram

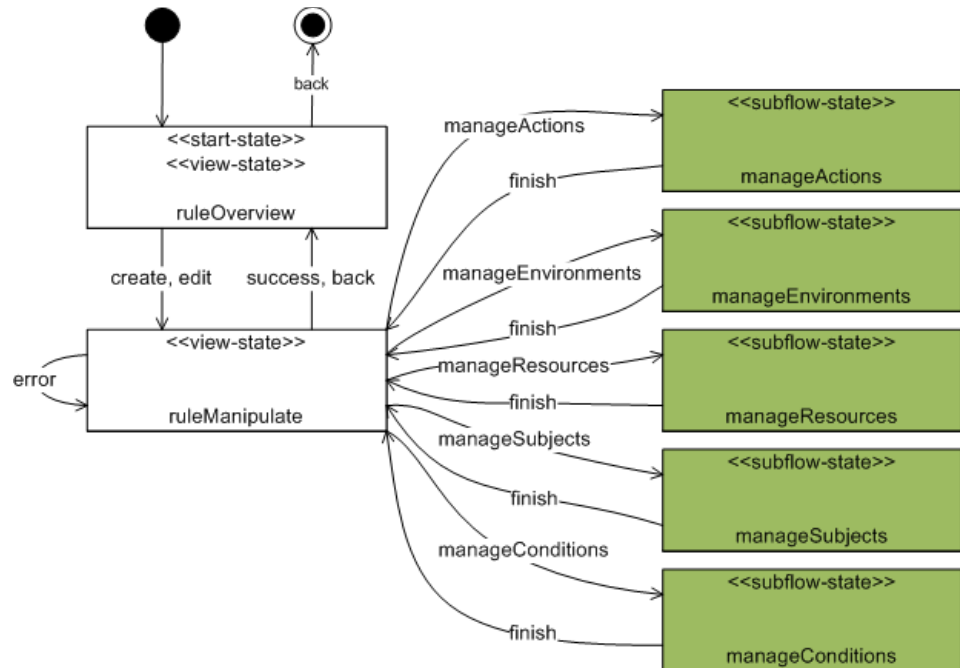


19.2.4.12 Rule Template flow

Overview

The *Rule Template flow* is responsible for listing, creating or editing `RuleTemplates`.

Rule flow diagram

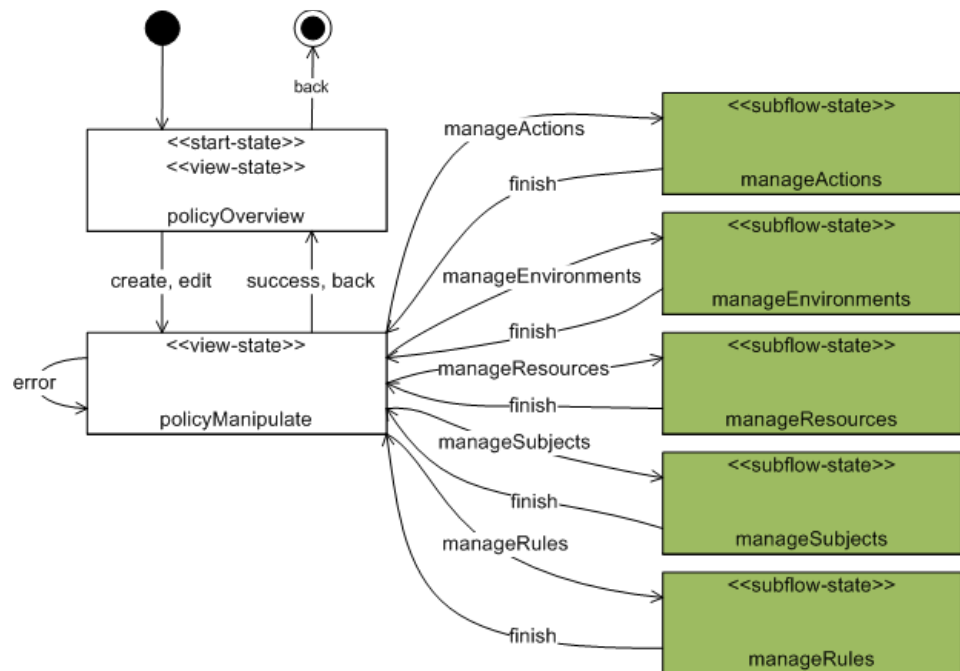


19.2.4.13 Policy Template flow

Overview

The *Policy Template flow* is responsible for listing, creating or editing PolicyTemplates.

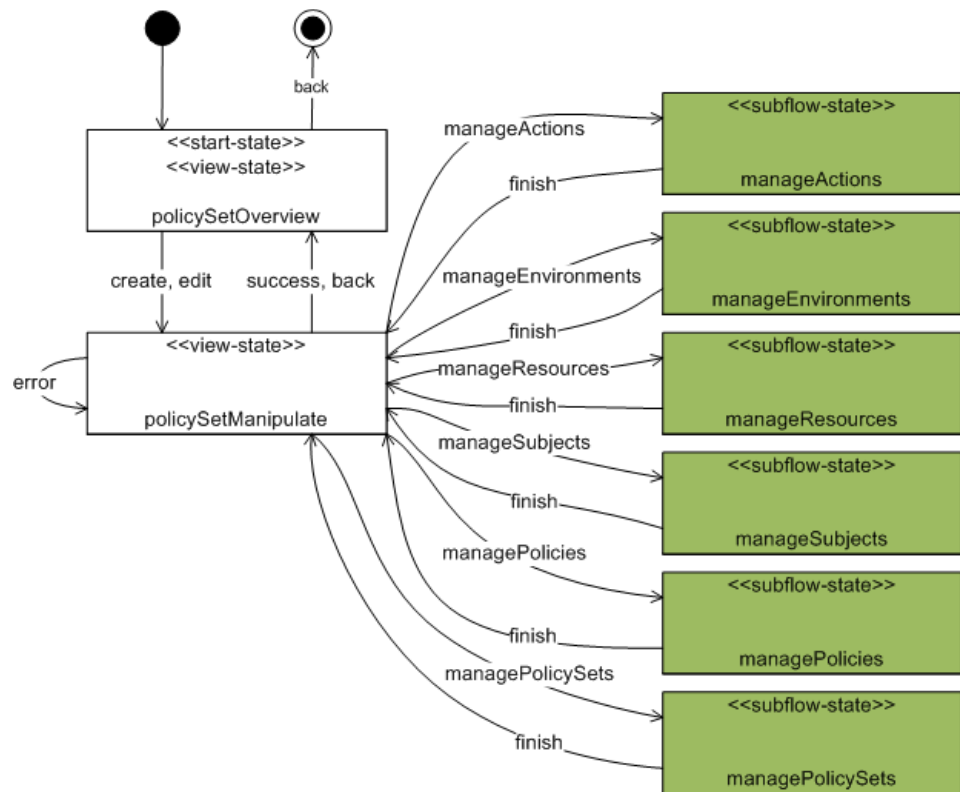
Policy flow diagram



19.2.4.14 Policy Set Template flow

Overview The *Policy Set Template flow* is responsible for listing, creating or editing `PolicySetTemplates`.

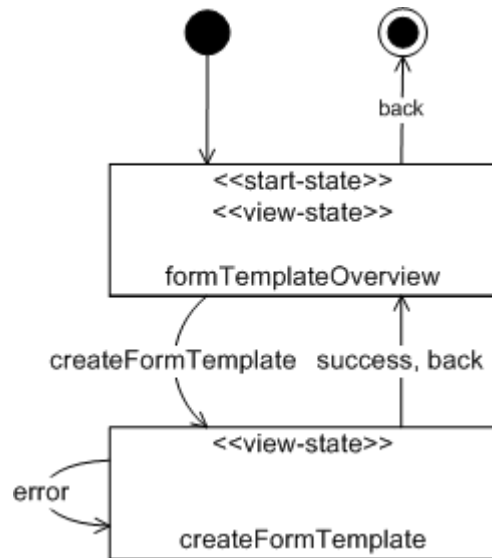
Policy set flow diagram



19.2.4.15 Policy Form Template flow

Overview The *Policy Form Template flow* is responsible for instantiating `PolicyTemplates`.

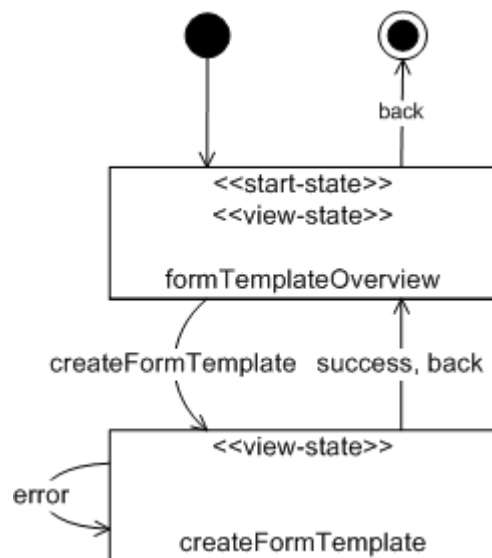
Policy form flow diagram



19.2.4.16 Policy Set Form Template flow

Overview The *Policy Set Form Template flow* is responsible for instantiating *PolicySetTemplates*.

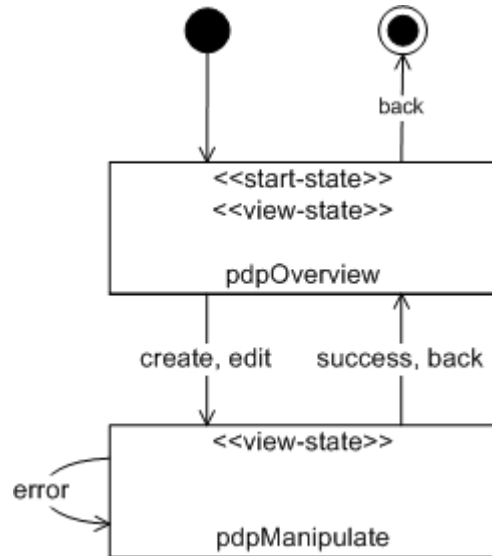
Policy set form flow diagram



19.2.4.17 PDP Management flow

Overview The *Pdp Management flow* is responsible for listing, creating and editing *PDPs*.

*Pdp
management
flow diagram*

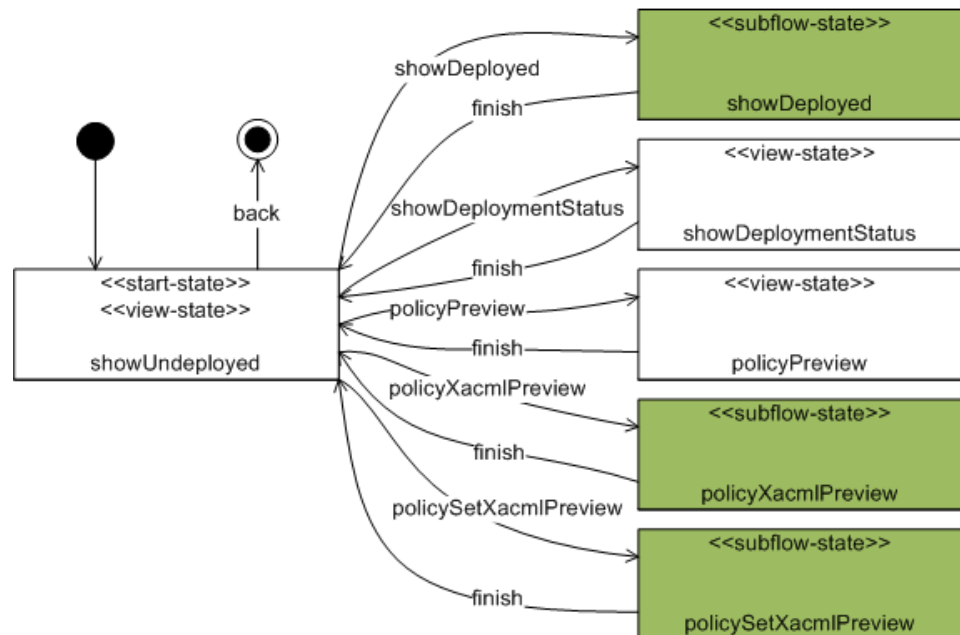


19.2.4.18 Deployment flow

Overview

The *Deployment flow* is responsible for deploying instantiated *PolicyTemplates* or *PolicySetTemplates*.

*Deployment
flow diagram*

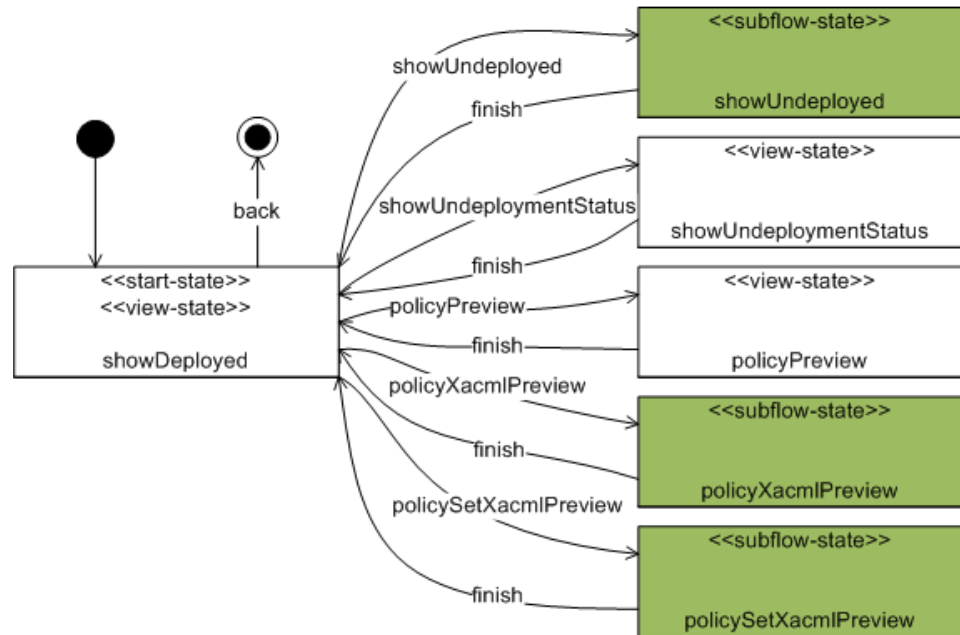


19.2.4.19 Undeployment flow

Overview

The *Undeployment flow* is responsible for undeploying (=revoking) instantiated *PolicyTemplates* or *PolicySetTemplates*.

Undeployment
flow diagram

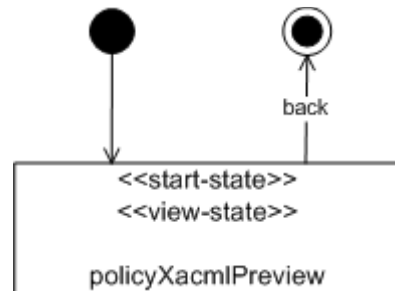


19.2.4.20 Policy XACML preview flow

Overview

The *Policy XACML preview flow* is responsible for displaying a XACML preview of a *PolicyTemplate*.

Undeployment
flow diagram

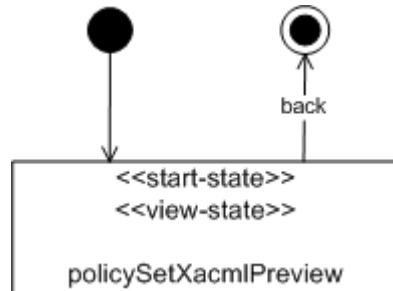


19.2.4.21 PolicySet XACML preview flow

Overview

The *Policy XACML preview flow* is responsible for displaying a XACML preview of a *PolicySetTemplate*.

Undeployment
flow diagram

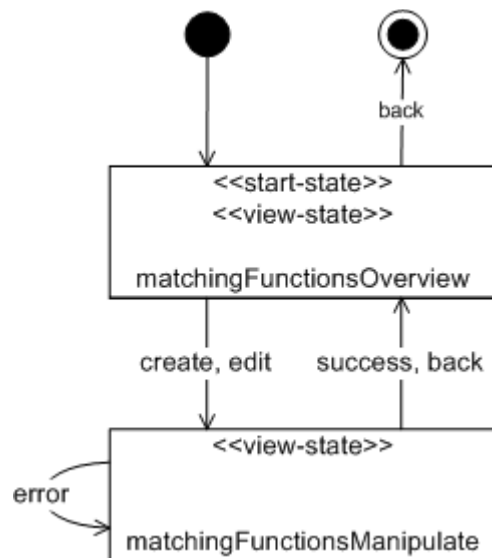


19.2.4.22 Matching Functions flow

Overview

The *Matching Functions* flow is responsible for listing, creating or editing MatchingFunctions.

Matching
function flow
diagram

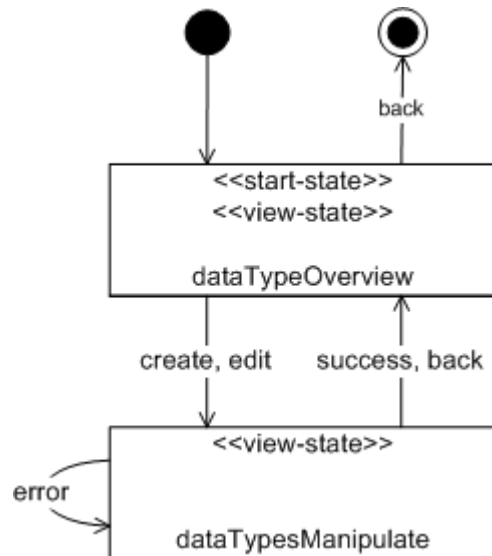


19.2.4.23 Data type flow

Overview

The *Data Type* flow is responsible for listing, creating or editing DataTypes.

Data type flow
diagram



19.2.5 Internationalization

<i>Overview</i>	This chapter contains the description of how internationalization was integrated in the web project and how it can be extended
<i>Message bundle</i>	In the package <code>org.herasaf.pap.jsf.locale</code> a properties file called <code>messages.properties</code> exist. In this file any text that is displayed on any page in the web project is located.
<i>Default language</i>	The default language is English (ISO country code <code>en</code>). Refer to [ISO3166] for a complete list of country codes.
<i>Add another language</i>	<p>In order to add another language, the correct properties file has to be created (<code>messages_[insert ISO country code here].properties</code>, listed in the <code>faces-config.xml</code> and ALL entries of the file <code>messages.properties</code> have to be translated.</p> <p>Example:</p> <p>We want to add support for Swedish (ISO country code <code>se</code>) to the HERAS^{AF} PAP web project, create a file named <code>messages_messages_se.properties</code>, copy it to <code>org.herasaf.pap.jsf.locale</code> and add a new entry to <code>faces-config.xml</code> (<code><supported-locale>de</supported-locale></code>).</p>

19.3 Design decisions

<i>Overview</i>	This chapter provides information about design decisions we took regarding the <code>herasaf-pap-icefaces-web</code> module.
<i>New role model</i>	To pay more attention to separation of business concerns, the exact usage of



what user interface elements may be accessed by the *BAdmin* or the *TAdmin* are defined.

In general, the *technical admin* (*TAdmin*) is allowed to create any template.

The *business admin* (*BAdmin*) is allowed to use *Actions_Templates*, *Environments_Templates*, *Resources_Templates* and *Subects_Templates*. He is allowed to create *PolicyTemplates*, *PolicySetTemplates* and *RuleTemplates*.

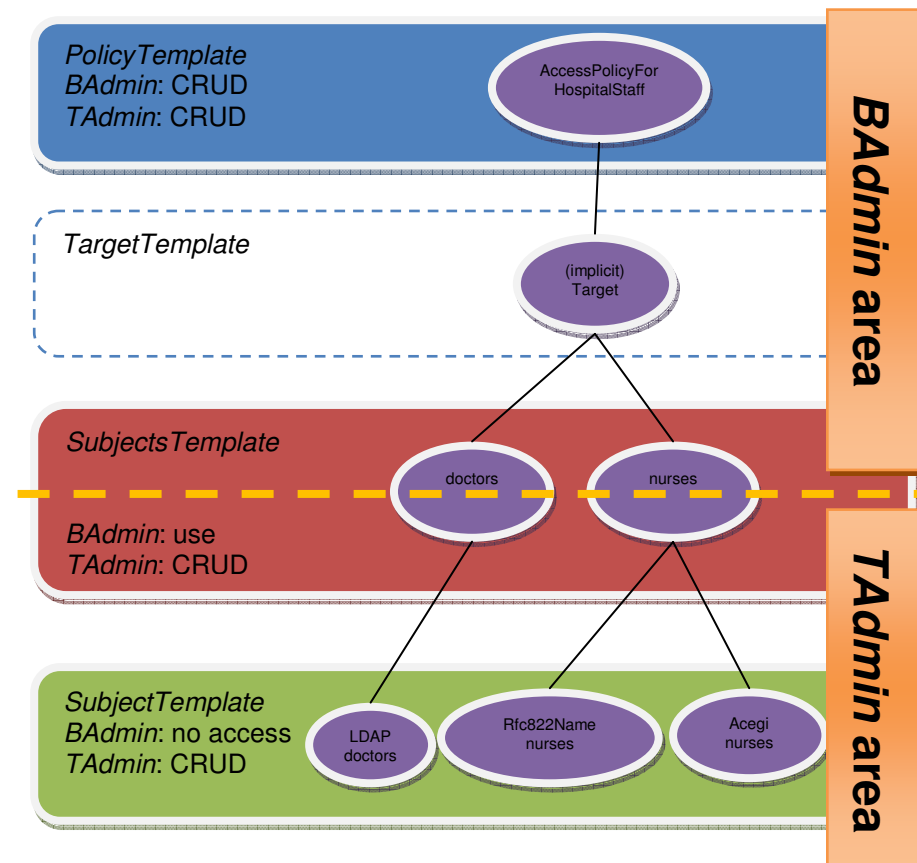


Illustration 19-1: Visualization of the role model in the case of *Subjects_Template*. The dashed orange line marks the demarcation between the *BAdmin* and *TAdmin* area. This illustration used *Subjects_Template* and *SubjectTemplate* but this behavior applies to all *GoFs*.

Note: The *TargetTemplate* layer between the *PolicyTemplate* and *Subjects_Template* layer is implicitly created and therefore visualized in a dashed rectangle.

BAdmin area
and *TAdmin area*

These two areas are the typical working areas the two role models cover. For example, it is common for a *BAdmin* to create *PolicyTemplates* by using *Actions_Templates*, *Environments_Templates*, *Resources_Templates* and *Subects_Templates* but he has no possibility to create, let's say a

`SubjectTemplate` because this has to be provided by the *TAdmin*.

Since the *TAdmin* knows a lot about technical details of XACML but most likely only little about the business, this separation makes a lot of sense.

Nevertheless, if a *TAdmin* is aware of the business processes going on, he absolutely has the ability to create fully-fledged policies that can be deployed without further manipulation by the *BAdmin*.

*Use
functionality
provided by
ICEfaces*

By using ICEfaces [*Icefaces*] in conjunction with JSF [*JSF*], there is an excellent property named `renderedOnUserRole` for all components provided by ICEfaces.

As the name suggests, this property makes it possible to control if a component gets rendered depending on the user role. The tight integration of Acegi security [*Acegi*], Spring [*Spring*] and ICEfaces [*Icefaces*] allows for simply using a defined user role within Acegi security as a value for this property:

Example: `renderedOnUserRole`

```
<ice:panelGroup
  renderedOnUserRole="ROLE_TECHNICAL_ADMIN">
```

Having mentioned this, we can have web user interface components and compositions rendered, which are defined only once, for a single as well as multiple user roles. For example, the *TAdmin* naturally does have more possibilities to interact with the underlying system than the *BAdmin*. He is the holder of all privileges, comparable to a system administrator.

However, the *BAdmin* shall only be able to see a subset in addition to some *BAdmin*-specific user interface components, such as deployment-related artifacts. This is where `renderedOnUserRole` kicks in: we only let those components and compositions be rendered that correspond to the particular role. Therefore, from a technical point of view, we can use exactly the same components for both roles but the web user interface of the *BAdmin* does look different from the one the *TAdmin* has access to although the same set of components are used.

20 herasaf-pap-xacml-transformer

Overview This chapter contains the description of the transformer interfaces and the implementation.

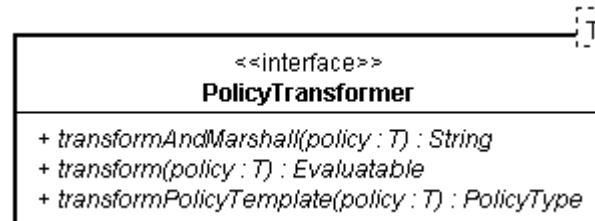
20.1 Interfaces

Overview This chapter contains the description of the transformer interfaces.

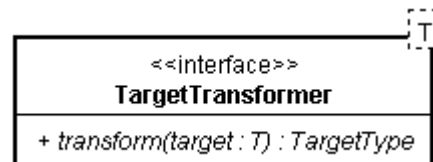
Policy Set Transformer



Policy Transformer



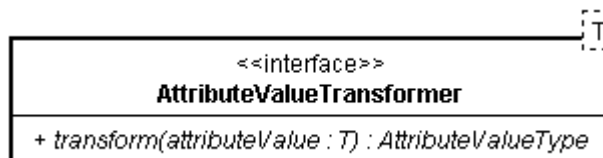
Target Transformer



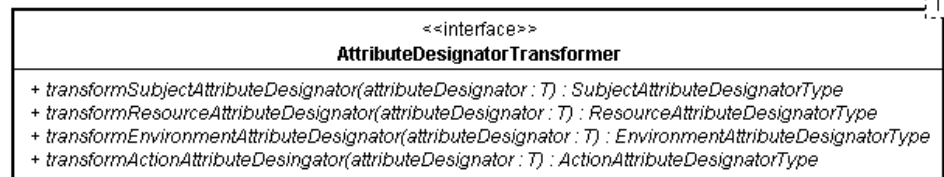
Expression Transformer



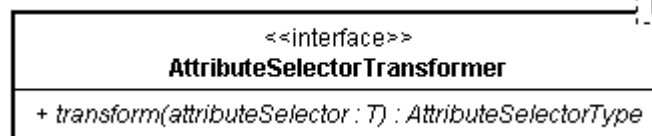
Attribute Value Transformer



Attribute Designator Transformer



Attribute Selector Transformer



Remarks

These interfaces have no dependencies to any other package in the actual HERAS^{AF} PAP implementation. Only dependencies to HERAS^{AF} XACML (package `org.herasaf.xacml.core`) exist.

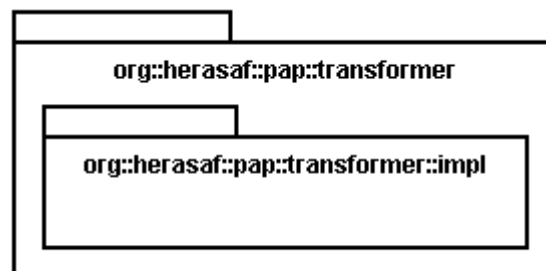
Therefore, the above interfaces can be implemented if there is need for a new transformer, for example to convert from any other classes or even a custom PAP implementation into XACML *Evaluatables*.

20.2 Implementation

Overview

This chapter focuses on the implementation and classes of the transformer.

Package diagram



Package description

org.herasaf.pap.transformer

The *transformer* package contains all interfaces, as listed in 20.1 (page 109, **herasaf-pap-xacml-transformer - Interfaces**).

org.herasaf.pap.transformer.impl

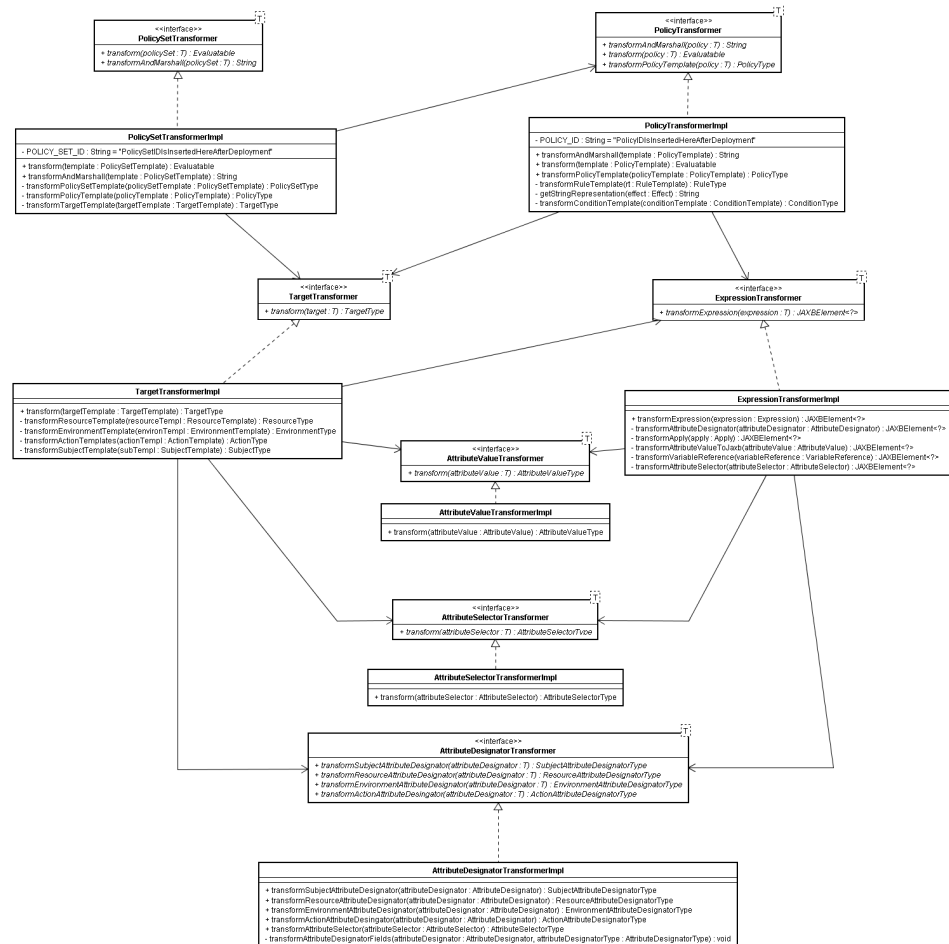
The package *transformer.impl* is the actual implementation of the transformer. It is used to transform *PolicyTemplates* or *PolicySetTemplates* to their corresponding XACML representation as *Evaluatables*.

20.2.1 Package org.herasaf.pap.transformer.impl

Overview

The package `org.herasaf.pap.transformer.impl` contains all the classes to convert an existing `PolicyTemplate` or `PolicySetTemplate` from the HERAS^{AF} template model into valid HERAS^{AF} XACML 2.0 Evaluatables.

Class diagram



20.2.2 Sequence Diagrams

Overview

As an example of a transformation a sequence diagram is supplied.

Precondition

A `PolicyTemplate` has been created (for example by using the HERAS^{AF} PAP web project).

In order to understand what is going to be transformed in the sequence diagram



below, the resulting policy is provided:

resulting policy

```

<?xml version="1.0" encoding="UTF-8"?>
<Policy
  xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

  xsi:schemaLocation="urn:oasis:names:tc:xacml:2.0:policy:sch
ema:os
  http://docs.oasis-open.org/xacml/access_control-xacml-
2.0-policy-schema-os.xsd"
  PolicyId="urn:oasis:names:tc:example:ExamplePolicy1"
  RuleCombiningAlgId="identifier:rule-combining-
algorithm:deny-overrides">
  <Target />
  <Rule
    RuleId="urn:oasis:names:tc:xacml:2.0:example:ExampleRule1"
    Effect="Permit">
    <Target>
      <Subjects>
        <Subject>
          <SubjectMatch
            MatchId="urn:oasis:names:tc:xacml:1.0:function:rfc822Name-
match">
              <AttributeValue
                DataType="http://www.w3.org/2001/XMLSchema#string">med.exam
ple.com</AttributeValue>
              <SubjectAttributeDesignator
                AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-
id" DataType="urn:oasis:names:tc:xacml:1.0:data-
type:rfc822Name" />
            </SubjectMatch>
          </Subject>
        </Subjects>
      </Target>
    </Rule>
  </Policy>

```

Corresponding
sequence
diagram



20.3 Testing

- Overview* This chapter explains how the `herasaf-pap-xacml-transformer` module was tested.
- How the transformer was tested?* The transformer was tested by wiring up a `PolicyTemplate` or `PolicySetTemplate` by hand, using a given XACML file (.xml). This template was then transformed into the HERAS^{AF} XACML 2.0 model. The result was compared to the original xml XACML policy file, to assure the transformer worked correctly.
- As referenced instances of templates are saved in `java.util.Sets` inside templates and these Sets store their elements unordered, the elements were not in the same order when the set was filled and when it was iterated over. Thus after transforming a template, the resulting element ordering is variable and not predictable.
- As a consequence, we needed a Java library that can compare two xml files without bothering about the element ordering.
- ExamXML* ExamXML [ExamXML] offers comparison between two xml files. Additionally, there are analysis tools for comparing differences in two files.
- Why not use XMLUnit?* XMLUnit [XMLUnit] offers the same benefits as ExamXML but is widely known and accepted in the Java community.
- However, XMLUnit is not able to successfully compare two xml files that have changed element ordering. As this should have no impact on the outcome of a xml file comparison, XMLUnit complains about the files being unequal, if the differentiating elements exceed a deepness of one.

An example of xml comparison

For a better understanding, an example of the problem above is given below.

As a matter of fact, the following two xml files are identical:

File one

```
<a>
  <b>
    <firstElement />
  </b>
  <b>
    <secondElement />
  </b>
</a>
```

File two

```
<a>
  <b>
    <secondElement />
  </b>
  <b>
    <firstElement />
  </b>
</a>
```

Testing the above files for equality, both XMLUnit and ExamXML claimed those were identical.

But what if there are more elements before a child element?

File one

```
<a>
  <b>
    <c>
      <firstElement />
    </c>
  </b>
  <b>
    <c>
      <secondElement />
    </c>
  </b>
</a>
```

File two

```
<a>
  <b>
    <c>
      <secondElement />
    </c>
  </b>
  <b>
    <c>
      <firstElement />
    </c>
  </b>
</a>
```

Testing the above two files for equality, ExamXML claimed those were identical, whereas XMLUnit claimed them to be unequal.

Summed up, by using XMLUnit, it was not possible to achieve an “areEqual” result comparing similar xml files as those listed above. ExamXML offers this possibility. This was the reason ExamXML was chosen as our xml comparison library.

20.4 Design decisions

Overview

This chapter provides information about design decisions we took regarding the `herasaf-pap-xacml-transformer` module.

Use of a generic interface

We introduced generic interfaces for the module instead of no interface at all or a interfaces without generics.

Discussion of our solution



By providing generic interface (located in `org.herasaf.pap.transformer`) different implementations of the interface can easily be switched through Spring's dependency injection.

Secondly, this interface can also be used with other external models as the generic type arguments specify the external model to be used. This is especially important if other models than the HERAS^{AF} template model is used.

21 herasaf-pap-deployment

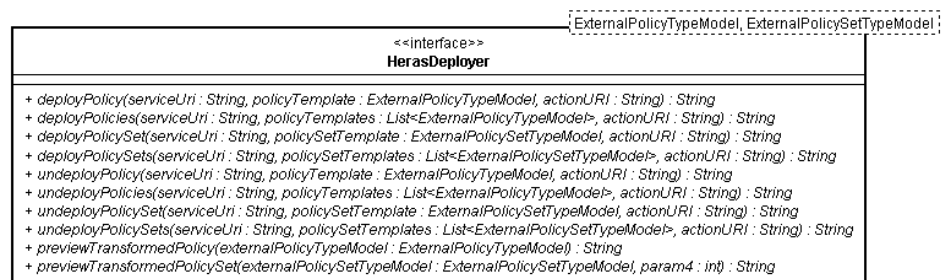
Overview This chapter contains information about

21.1 Interfaces

21.1.1 Interface HerasDeployer

Description This generic interface provides all methods needed for deployment of policies and policysets. It ensures the compatibility with other models than the HERAS^{AF} template model. Additionally, different implementations of the interface can easily be switched through Springs dependency injection.

UML



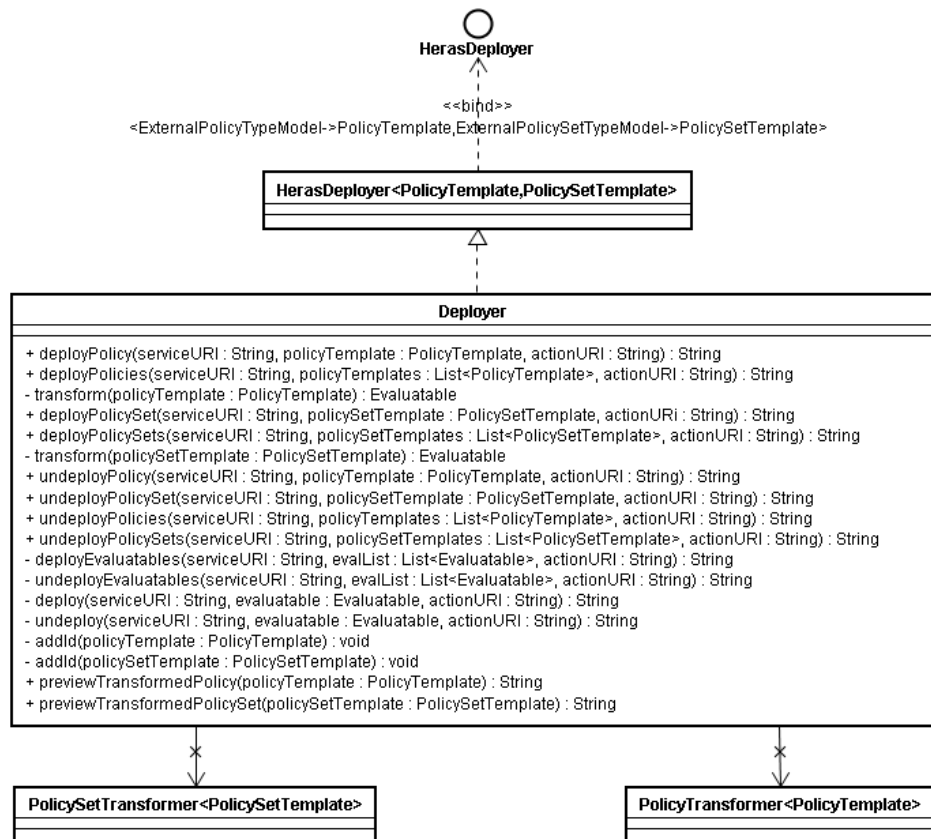
Operation	Description
deployPolicy()	Deploys the specified ExternalPolicyTypeModel to the specified serviceUri and actionUri.
deployPolicySet()	Deploys the specified ExternalPolicySetTypeModel to the specified serviceUri and actionUri.
deployPolicies()	Deploys the list of the specified ExternalPolicyTypeModels to the specified serviceUri and actionUri
deployPolicySets()	Deploys the list of the specified ExternalPolicySetTypeModels to the specified serviceUri and actionUri.
undeployPolicy()	Undeploys the specified ExternalPolicyTypeModel.
undeployPolicySet()	Undeploys the specified ExternalPolicySetTypeModel.
undeployPolicies()	Undeploys the list of the specified ExternalPolicyTypeModels.
undeployPolicySets()	Undeploys the list of the specified ExternalPolicySetTypeModels

previewTransformedPolicy()	Returns the string representation of the transformed ExternalPolicyTypeModel. If this method is not needed, just return null or "".
previewTransformedPolicySet()	Returns the string representation of the transformed ExternalPolicySetTypeModel If this method is not needed, just return null or "".

21.2 Implementation

Overview This chapter contains information about how we realized the HERAS^{AF}-pap-deployment module.

Class diagram



Packages **org.herasaf.pap.deployment**
Contains the generic interface `HerasDeployer` that needs to be fulfilled by deploying entities.

org.herasaf.pap.deployment.impl
Contains an implementation of the `HerasDeployer` interface. Other

implementations may be added.

21.3 Testing

Overview The deployment module was tested in the integrationtests together with the ws-client, the transformer and ws-policy module.

How the deployment module was tested? We tested the deployment module using a separate web project and integrated the ws-policy module as a dependency. Additionally the PDP has been integrated.

There were two sorts of tests:

- Testing the deployment and undeployment of `Evaluatables`
- Testing the validate method on the server-side

The deployment and undeployment of `Evaluatables` was carried out with single `Evaluatables` as well as multiple `Evaluatables` per SOAP-message.

21.4 Design decision

Overview This chapter provides information about the design decisions for the deployment module.

Use of a generic interface We introduced a generic interface for the `HerasPdpDeployer` instead of no interface at all or a normal interface.

Discussion of our solution

By providing the generic interface `HerasPdpDeployer` for a deploying entity, we kill two birds with one stone. Firstly, through the usage of an interface, different implementations of the interface can easily be switched through Spring's dependency injection.

Secondly, this interface can also be used with other external models as the generic type arguments specify the external model to be used. This is especially important if other models than the HERAS^{AF} template model shall be deployed.

22 herasaf-pap-ws-client

Overview This chapter contains information about how we realized the herasaf-pap-ws-client module.

22.1 Interfaces

22.1.1 Interface HerasPdpWsClient

Description The `HerasWsClient` interface provides the interface for a client to communicate with a server-side entity.

UML



Operation	Description
<code>sendXACMLPolicyStatement()</code>	Method that gets called from the various <code>deploy()</code> and <code>undeploy()</code> methods of this interface. This is the method that invokes the <code>marshallAndSend()</code> method provided by the super class <code>WebServiceGatewaySupport</code> provided by Spring. This is where the connection is made to the server-side and the SOAP message is being sent.
<code>deploy()</code>	This is an overloaded method for deploying <code>Evaluatables</code> from a client to the server, e.g. a PDP.
<code>undeploy()</code>	This is an overloaded method for undeploying <code>Evaluatables</code> on a server, e.g. a PDP

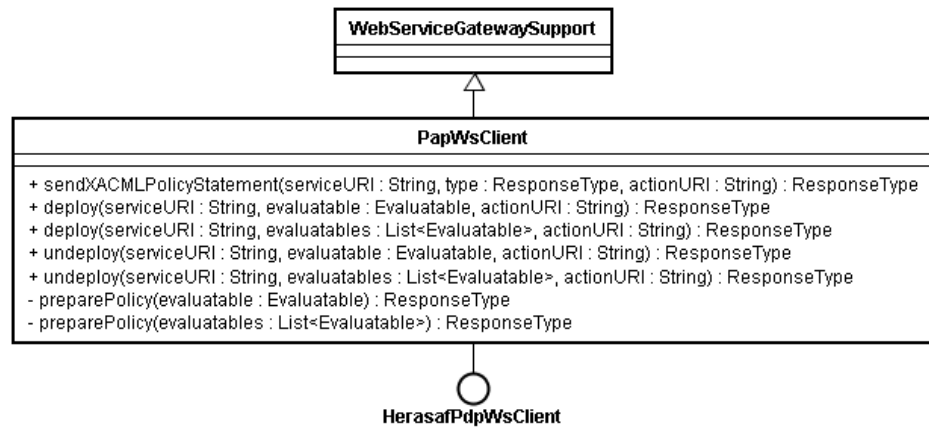
Remark The interface method `sendXACMLPolicyStatement()` could also be taken out of the interface and be made private in the implementation. This method was included so we could test with manipulated `ResponseTypes` as this would not have been possible otherwise.

For a future release, this method should to be refactored out of the interface.

22.2 Implementation

Overview This chapter contains information about how we realized the implementation HERASAF-pap-ws-client module.

Class diagram



Packages

org.herasaf.pap.ws.client

Contains the interface for the web service client in HERAS^{AF}.

org.herasaf.pap.ws.client.impl

This package contains the implementation of the `HerasafPdpWsClient`.

Remarks

Naming of this interface has been the reason for a discussion. We decided to put 'Pdp' into the name since this client, which implements the `HerasafPdpWsClient` interface, is intended to communicate with a PDP only and not with other endpoint instances as well.

Nevertheless, this is a minor refactoring to do if this should change for future releases of HERAS^{AF}.

22.3 Testing

How we tested this module

Testing this module was done, as with the HERAS^{AF} deployment module, through the integration tests mentioned in 21.3, page 119 *Testing*.

Remark

We thought about using a mock implementation of the PDP for testing purposes, as this was done within the HERAS^{AF} XACML core as well as the HERAS^{AF} PAP module, but discarded this possibility out of the following reasons:

- The main objective of testing deployment and undeployment of policies is whether they are saved correctly into the database on the PDP and if they can be deleted from this database again.
- The client does not contain logic that is worthy of testing. As developers say, the client is a 'dumb' entity.

22.4 Design decisions

<i>Overview</i>	<p>This chapter shall give insight on the thoughts we had about certain design aspects of the HERAS^{AF}-pap-ws-client module.</p>
<i>Usage of interceptors for setting HTTP headers</i>	<p>For standard-conformance, certain HTTP headers were recommended by OASIS to be used when SOAP over HTTP in conjunction with SAML is in place.</p> <p>Discussion of solution</p> <hr/> <p>By using the common HERAS^{AF} SAML module, we could take advantage of already implemented custom interceptors that set those headers automatically before the SOAP message is sent.</p> <p>This was chosen since the Spring web service framework recommends the use of interceptors as this enhances flexibility and configurability since various interceptors can be injected using Spring.</p> <p>Additionally, through using dependency injection, the low coupling of different components is another positive aspect of this solution.</p>
<i>Assertions and ResponseTypes</i>	<p>Again, to be conform regarding SAML, <code>Assertions</code> and <code>ResponseTypes</code> require some properties set and initialized with the correct value.</p> <p>Discussion of solution</p> <hr/> <p>Factory classes have been implemented in the HERAS^{AF} SAML module to be used when creating <code>Assertions</code> or <code>ResponseTypes</code>.</p> <p>This ensures that both elements are standard conform for every module using these factory classes.</p> <p>If there are new releases of SAML with new definition of fields or values for those fields (e.g. version 3.0 instead of 2.0), changes have to be in the HERAS^{AF} SAML module only without changing other code or modules. This increases maintainability.</p>
<i>Protocol for deploying policies</i>	<p>There is not yet a standard way about how to deploy a policy created on the PAP to a PDP, only vice versa. We therefore needed to implement a small protocol for this issue to be resolved.</p> <p>Discussion of solution</p> <hr/> <p>We decided to use <code>ResponseType</code> to convey <code>Evaluatables</code> over the wire. [SAMLProfile, page 35]:</p> <p style="padding-left: 40px;">“It [<code>ResponseType</code>] MAY be used to convey or store XACML policies for other purposes”</p> <p>We also needed to indicate to the client whether deployment and undeployment of <code>Evaluatables</code> was successful or not. There are no SAML status codes explicitly defined for this reason so we used</p> <ul style="list-style-type: none"> • <code>urn:oasis:names:tc:SAML:2.0:status:Success</code> and • <code>urn:oasis:names:tc:SAML:2.0:status:Responder</code> <p>to indicate successful and unsuccessful deployment or undeployment respectively.</p>



[SAMLProfile, page 40] defines these status codes as follows:

- `urn:oasis:names:tc:SAML:2.0:status:Success:`
The request succeeded. Additional information MAY be returned in the `<StatusMessage>` and/or `<StatusDetail>` elements.
- `urn:oasis:names:tc:SAML:2.0:status:Responder:`
The request could not be performed due to an error on the part of the SAML responder or SAML authority.

`ResponseType` was the only element suitable for our needs due to the definition given above. Eventually, we wanted to stay standard conform at all costs so there was not really an alternative.

The status codes were chosen are rather general. Since this thesis is based on [SAMLProfile] and this is a working draft only, changes are to be expected.

23 herasaf-pdp-ws-policy

Overview This chapter contains information about how we realized the HERASAF-pdp-ws-policy module.

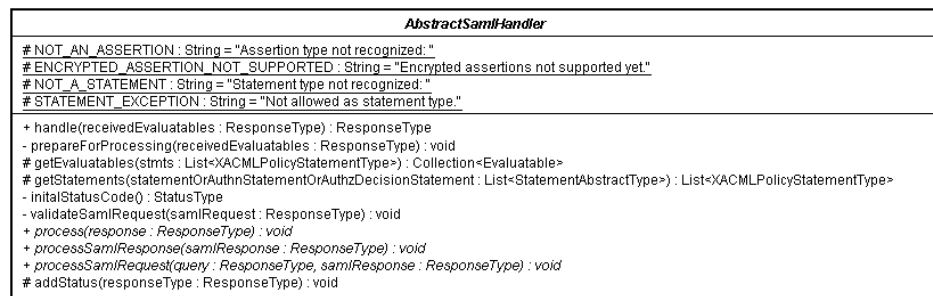
23.1 Interfaces

23.1.1 Abstract class AbstractSamlHandler

Description The `AbstractSamlHandler` is the template method class that provides the hook methods to be implemented by sub classes.

In addition, the requests are validated for standard conformance. As of now, this validation includes checking if the version of the request is equal "2.0".

UML



Operation	Description
<code>handle()</code>	Template main method that defines the chain of operations to be called in the correct order.
<code>process()</code>	Hook operation to be implemented by subclasses. Handles the main operations, e.g. whether deployment or undeployment shall take place.
<code>prepareForProcessing()</code>	Method for extracting <code>Evaluatables</code> out of the <code>ResponseType</code> . Since an <code>Evaluatable</code> is sent to the PDP regardless whether deployment or undeployment is to be carried out, extracting these <code>Evaluatables</code> has to be done anyways.
<code>processSamlRequest()</code>	Hook operation of the template method <code>handle()</code> of class <code>AbstractSamlHandler</code> . Can be used to post-process a response, e.g. for signing.
<code>processSamlResponse()</code>	Hook operation of the template method



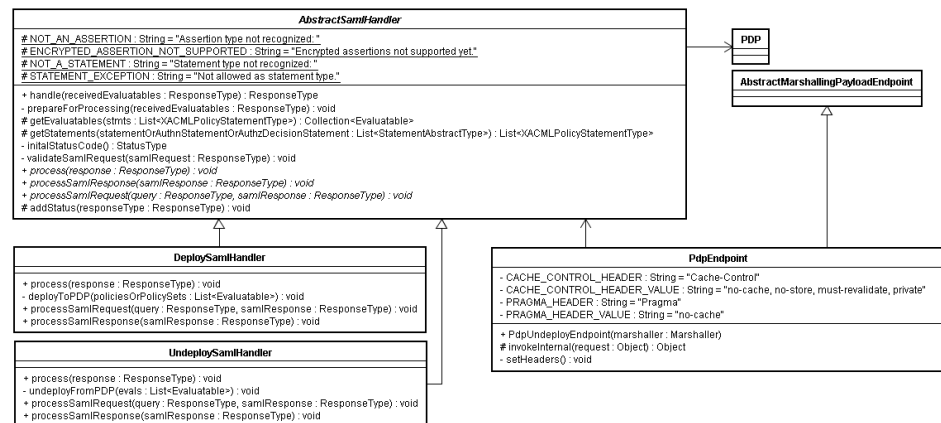
<code>Response()</code>	<code>handle()</code> of class <code>AbstractSamlHandler</code> . Can be used to pre-process a response, e.g. for signing.
<code>validateSamlRequest()</code>	Validates the specified SAML request and throws a <code>SamlException</code> if request didn't pass validation. For example: The version of the SAML request is not 2.0, thus the request can't be handled because of a version mismatch.

23.2 Implementation

Overview

This chapter contains information about how we realized the implementation of the HERAS^{AF}-pdp-ws-policy module.

Class diagram



Packages

org.herasaf.pdp.ws.policy.endpoints

Contains implementations for the deploy and undeploy endpoints. These two endpoints use the class `AbstractSamlHandler` from the package `org.herasaf.pdp.ws.policy.saml` and therefore have a dependency to this package.

org.herasaf.pdp.ws.policy.saml

Contains abstract as well as concrete implementations for handling incoming SAML messages. This is realized using the template method pattern.

23.3 Testing

How we tested this module

Testing this module was done, as with the HERAS^{AF} deployment module, through the integration tests mentioned in 21.3, page 119 *Testing*.



Remarks

Testing this module could have been partly done using mock objects, e.g. a `PDPMock` class. We then would have been able to test simple functionality like validating request.

This was not done due to time constraints we were facing towards the end of this thesis.

Here is how a mock test environment could be set up:

1. Create a test class including the desired tests like testing the validation of SAML requests.
2. Create a `PDPMock` class and create a field in the test class of type `PDPMock`.
3. Inject the `DeploySamlHandler` and `UndeploySamlHandler` as implementations for the `AbstractSamlHandler` field in the test class.
4. Create a non-standard conform `ResponseType` (e.g. set version to 1.0) in the test method and invoke the `handle()` method on the `AbstractSamlHandler`.
5. Check if the correct exception has been thrown.

23.4 Design decisions

Overview

This chapter gives insight on the design decision we were faced with when developing the HERAS^{AF} `pdp-ws-policy` module.

Endpoint-Mappings

Since the Spring web services framework has been used, we only needed to evaluate different solutions for doing the endpoint mapping regarding this framework.

There are three possibilities to do the endpoint mapping:

- Mapping based on the request URI: depending on the URI of the request, the message is routed to the correct endpoint.
- HTTP header `SOAPAction`: do the mapping based on the `SOAPAction` field in a SOAP HTTP request.
- WS-addressing: do the mapping based on WS-addressing values in the SOAP header.

Discussion of solution

We decided to use WS-addressing because we are not bound by any transport-layer protocol. Secondly, we chose it for its simplicity at the implementation level and that we are standard conform.

URI mapping is very similar to WS-addressing although works on the transport layer and thus is dependent on this layer. This is the reason why this solution was rejected.

Using the `SOAPAction` in the SOAP HTTP request was rejected because it is recommended by OASIS in their SAML bindings [SAMLBindings] to use <http://www.oasis-open.org/committees/security> as the value for the `SOAPAction` in a SOAP HTTP request. Additionally, using the `SOAPAction` header field is SOAP v1.1 specific and cannot be used with SOAP

v1.2. This was another strong fact against this solution.

*Usage of
HERAS^{AF}
SAML module*

The commonly HERAS^{AF} SAML module was used to ensure standard conformance over all modules within HERAS^{AF} that need SAML functionality in any way.

Discussion of solution

By using a common module that is used by various modules within HERAS^{AF}, it is ensured that all modules use exactly the same configuration properties regarding SOAP and SAML messages.

This also increases maintainability as changes have only to be made in one module.

*Template
method pattern*

In `AbstractSamlHandler`, we implemented the template method pattern for processing SAML requests.

Discussion of solution

Note the flexibility this solution leaves a developer with: through the template pattern, only the hook methods need to be implemented by sub classes. In conjunction with the two provided methods `processSamlRequest()` and `processSamlResponse()`, this forms a powerful mechanism but still is configurable regarding future changes of the SAML standard.

If used together with Spring's dependency injection, the behavior of the endpoint can be altered by just adjusting the context file.

Part IV: Deployment

1 Overview

Overview To fully build the HERAS^{AF} PAP implementation an automated building environment is auxiliary.

2 Apache Maven 2

Introduction Maven 2 [Maven] is used.
 Refer to online help [Maven] for guidance or installation instructions.

2.1 Global Maven Configuration

Introduction This chapter covers the global maven configuration.
 Starting a maven build in the module herasaf-pap-icefaces will build all PAP relevant modules, including the web project.

Standard build Go to the herasaf-pap-icefaces directory in the command console to perform a maven full build of all PAP relevant projects and execute following command:

```
mvn clean install
```

pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project>
  <modelVersion>4.0.0</modelVersion>
  <groupId>org.herasaf.pap.icefaces</groupId>
  <artifactId>herasaf-pap-icefaces</artifactId>
  <packaging>pom</packaging>
  <version>0.9.0</version>
  <name>HERAS-AF ICEfaces Policy Administration Point
  (root)</name>

  <pluginRepositories>
    <pluginRepository>
      <id>apache.snapshots</id>
      <url>
        http://people.apache.org/repo/m2-snapshot-
        repository
      </url>
    </pluginRepository>
  </pluginRepositories>

  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-source-plugin</artifactId>
        <executions>
```

1

2

3



```

    <execution>
      <id>attach-sources</id>
      <phase>verify</phase>
      <goals>
        <goal>jar</goal>
      </goals>
    </execution>
  </executions>
</plugin>
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-surefire-plugin</artifactId>
  <version>2.4.2-SNAPSHOT</version>
  <configuration>
    <skip>>false</skip>
  </configuration>
</plugin>
<plugin>
  <artifactId>maven-javadoc-plugin</artifactId>
  <executions>
    <execution>
      <phase>package</phase>
      <goals>
        <goal>jar</goal>
      </goals>
    </execution>
  </executions>
  <configuration>
    <aggregate>>true</aggregate>
    <quiet>>true</quiet>
  </configuration>
</plugin>
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-compiler-plugin</artifactId>
  <configuration>
    <source>1.5</source>
    <target>1.5</target>
  </configuration>
</plugin>
</plugins>
</build>

<modules>
  <module>../herasaf-pap-icefaces-dao</module>
  <module>../herasaf-pap-icefaces-domain</module>
  <module>../herasaf-pap-icefaces-service</module>
  <module>../herasaf-pap-icefaces-web</module>
  <module>../herasaf-pap-deployment</module>
  <module>../herasaf-pap-ws-client</module>
  <module>../herasaf-pap-xacml-transformer</module>
</modules>
</project>

```

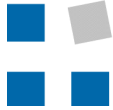
4

5

6

7

Legend



- 1 Definition of pom model version, project groupId, base name of the artifact being generated by this project, the packaging of the output and the actual version of the product.
- 2 Additional plugins repository location is added. This is needed for finding the maven surefire plugin from apache.
- 3 The source plugin creates a jar archive of the source files of the current project.
- 4 The maven surefire plugin is used during the test phase of the maven build lifecycle to execute the unit tests of the application.
- 5 The javadoc plugin uses the Javadoc tool to generate javadocs for the project.
- 6 Configuration so the Java compiler allows JDK 5.0 sources.
- 7 Additional modules that are built when this project is built. As this project is empty (except of this pom.xml file), all PAP modules are built using just one command by using the “mvn clean install “ command inside the herasaf-pap-icefaces directory.

2.2 Maven Configuration for further modules

Introduction

This chapter covers the general configuration for modules included in the parent project file (see previous chapter)

General pom.xml for herasaf-pap-icefaces

```
<parent>
  <groupId>org.herasaf.pap.icefaces</groupId>
  <artifactId>herasaf-pap-icefaces</artifactId>
  <version>0.9.0</version>
</parent>
```



Legend

- 1 Every successor pom.xml has to include its parent as shown.

Scoping tests

```
<dependency>
  <groupId>org.testng</groupId>
  <artifactId>testng</artifactId>
  <version>5.7</version>
  <classifier>jdk15</classifier>
  <scope>test</scope>
</dependency>
```



Legend

- 1 Of course a lot of TestNG tests have been written. But in a productive system neither test classes need to be provided, nor have any external test-related libraries to be included. Thus the <scope> element has been used inside of pom files to exclude test-specific libraries from the final java archive

Further

Analysis of duplicated dependencies has been made for every project.

configuration

Duplicated dependency entries can happen easily, because used components may have dependencies themselves.

In order to avoid this problem, dependencies have been excluded from dependent projects and separately defined. The analysis can effortlessly be achieved by using the Q4Eclipse maven plugin for eclipse [Q4Eclipse].

Appendix A: Issues, Extensions and Refactorings

1 Overview

Introduction This appendix contains open issues, extension points and recommended refactorings of the HERAS^{AF} PAP.

2 Open Issues

Overview This chapter contains open issues of the HERAS^{AF} PAP.

Persistence Since `FETCHTYPE.EAGER` is used in the HERAS^{AF} PAP domain, the execution time of database queries grows with the amount of `Templates` that are already persisted.

Web user interface `Evaluatables` are sent to all PDPs that were configured in the web user interface. It is unknown what policies are deployed to what PDPs.

When editing a `RuleTemplate`, `PolicyTemplate` or `PolicySetTemplate` in the web user interface, already selected children templates are listed. In the “available xxx templates” list, all available children `Templates` are listed, including the ones already selected.

The problem relies in the possibility to add an already selected template again to the list. It neither has affect on the logic in the domain (the child is only added once), but is only a visual problem that may distract the user.

If we create a new `RuleTemplate`, `PolicyTemplate` or `PolicySetTemplate`, this problem will not apply.

Microsoft Internet Explorer 7.0 is not recommended to be used to access the HERAS^{AF} PAP web project, as it cannot render ICEfaces popup modal dialogues. No interaction is possible at all if a popup shows up and alignment of elements generally differs. It is advised to access the web page with the Mozilla Firefox browser for full RIA experience.

Fast clicking in the web user interface may result in an error. For example if we create a new policy set, add a policy and click the “save” button quickly, it can be reproduced.

When creating policies, no `AttributeValues` can be set in a `ConditionTemplate` by the BAdmin. This should be implemented as soon as possible. However the `AttributeValue` of a `ConditionTemplate` can be set directly in the `ConditionTemplate` and is not ignored, it is just a displaying problem.

Creation of Policy Sets (in the meaning of instantiating) can be executed but no business informations can be entered. Therefore the business information has to



be entered directly when creating `TargetMatchTemplates` or `ConditionTemplates`.

When creating (in the meaning of instantiation) a Policy or Policy Set and the save-button is clicked, a popup is displayed asking if the user is sure about saving this policy (set). If clicked on 'yes', the popup window disappears but leaving the screen obscured. No interactions can be made in the browser unless hitting 'F5' for refreshing the web browser.

Previewing a policy set is not supported yet, although it can be previewed as XACML representation.

When deploying to an invalid PDP entry, a stack trace is shown. Exception handling should be implemented properly.

The german `messages_de.properties` file has been omitted due to timing issues.

Internationalization has been introduced in all backing beans, although no overall check has been made at the end. There may still exist some hardcoded strings, but they should not affect the views but is used for internal usage.

Concurrency

Concurrency should be taken care of in the web user interface. No concurrent actions have been tested yet, but theoretically a BAdmin and a TAdmin may work simultaneously on the same PAP.

Example of failing actions: While the TAdmin is editing a `TargetMatchTemplate`, the BAdmin clones a `PolicyTemplate` that uses the same `TargetMatchTemplate`. What is the result? This topic has not been paid attention to yet.

3 Extension Points

Overview

This chapter contains the possible extension points to this implementation. Listed are capabilities which could make the work with the HERAS^{AF} PAP more comfortable.

Persistence

Changing `FETCHTYPE.EAGER` to `FETCHTYPE.LAZY` should improve database query performance distinctively.

Auditing

Auditing enables an administrative entity to detect and track changes made on the PAP. This could include tracking what policies or policy sets have been created, manipulated or deleted in addition to logging all these manipulations for later inspection.

Workflow

A workflow model should be implemented clearly defining the steps involved in

<i>abstraction</i>	different processes that can be carried out using HERAS ^{AF} PAP. For example a policy needs to be reviewed by a user with special rights before it can be deployed to a PDP.
<i>Personalized PAP web user interface</i>	A personalized web UI on the PAP enables different users to work on the PAP and make their changes. User management functionality shall be provided as well. The personalized web UI goes hand in hand with auditing as it will be possible to make a clear statement about who at what time and what change has been made on the PAP.
<i>Security model</i>	As of now, there is no security implemented on any level of HERAS ^{AF} . A security model defines the aspects as well as requirements for a module in HERAS ^{AF} to be fulfilled. This includes secured communication channels between different HERAS ^{AF} modules, e.g. between the PAP and PDP. Not only communication channels but also access to the PAP needs better security barriers, for example by providing a personalized web UI in addition to the roles already defined.

4 Refactorings

<i>Overview</i>	This chapter contains refactorings which should be done to improve performance or design of the HERAS ^{AF} PAP implementation.
<i>Random number generator</i>	At the moment <code>java.lang.Math.random()</code> is used to generate the <code>policyId</code> or <code>policySetId</code> . It should be considered to use a better solution to generate random numbers.
<i>Web service client</i>	Make <code>sendXACMLPolicyStatement()</code> method private, as it should not be accessed directly but only be delegated to from its own <code>deploy()</code> and <code>undeploy()</code> methods.
<i>Layout of "Manage Data Types"</i>	The layout of this page breaks ranks with the overall layout.
<i>XACML preview</i>	When creating as BAdmin an XACML preview in the web user interface, the displayed XML is not rendered and displayed colored respectively not formatted. This should be fixed.
<i>Creation of Policies</i>	When a BAdmin is creating a Policy (in the meaning of instantiation), the preview sentence is currently hardcoded in German. This should be refactored to <code>messages.properties</code> files and correctly concatenated.
<i>Up-to-date versions of used libraries</i>	The versions used of Spring Web Flow and ICEfaces are not actual. Upgrading to newer versions is suggested.

Appendix B: General

5 Bibliography

5.1 HERAS^{AF} documents and theses

[NZ08PAPDev] **HERAS^{AF}: PAP.** Developer's Guide, Bachelor thesis – Rapperswil, HSR, 2008.
P. Neyer and C. Zellweger

[NZ08PAP] **HERAS^{AF}: Policy Deployment Module.** Bachelor thesis – Rapperswil, HSR, 2008.
P. Neyer and C. Zellweger

[DOH07DA] **HERAS^{AF}: XACML Implementierung.** Diploma thesis - Rapperswil : HSR, 2007.
F. Huonder, S. Oberholzer and S. Dolski

[DOH07DADev] **HERAS^{AF}: XACML Developer's Guide.** Diploma thesis - Rapperswil : HSR, 2007.
F. Huonder, S. Oberholzer and S. Dolski

[DOH07SA] **HERAS^{AF}: PDP Web Service Endpoint.** Term thesis - Rapperswil. HSR, 2007.
F. Huonder, S. Oberholzer and S. Dolski

[EGG06] **HERAS^{AF} - Holistic Enterprise-Ready Application Security Architecture Framework, Manageable policy-based access control for J2EE.**
Diploma thesis – Rapperswil. HSR, 2006
R. Eggenschwiler

[GRAF06] **Distributed Access Control Policies – Enterprise Ready.** Diploma thesis – Rapperswil. HSR, 2006
Y. Graf

5.2 Specifications and standards

[SAML] OASIS
Security Assertion Markup Language v2.0
<http://docs.oasis-open.org/security/saml/v2.0/saml-2.0-os.zip>

[SAMLBindings] OASIS

Bindings for the OASIS Security Assertion Markup Language (SAML) V2.0

<http://docs.oasis-open.org/security/saml/v2.0/saml-bindings-2.0-os.pdf>

[SAMLProfile]

OASIS

SAML 2.0 Profile of XACML, Verions 2, working draft 5

<http://www.oasis-open.org/committees/download.php/24681/xacml-profile-saml2.0-v2-spec-wd-5-en.pdf>

[XacmlSpec]

OASIS

eXtensible Access Control Markup Language TC v2.0 (XACML)

<http://www.oasis-open.org/committees/download.php/10578>

5.3 Web resources

[InfoExpert]

<http://web.cs.wpi.edu/~gpollice/cs4233-a05/CourseNotes/maps/class4/InformationExpert.html>

[OASIS]

http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml

[SunXacml]

<http://sunxacml.sourceforge.net/>

5.4 Frameworks and Software

[Acegi]

<http://www.acegisecurity.org/>

[EclEmma]

<http://www.eclEmma.org/index.html>

[ExamXML]

<http://www.a7soft.com/>

[Facelets]

<https://facelets.dev.java.net/>

[ICEfaces]

<http://www.icefaces.org/>

[JAXB]

<http://java.sun.com/developer/technicalArticles/WebServices/jaxb/>

[JSF]

<http://java.sun.com/javaee/javaserverfaces/>

[Maven]

<http://maven.apache.org/>

<i>[Spring]</i>	http://www.springframework.org/
<i>[SpringWebFlow]</i>	http://www.springframework.org/webflow
<i>[SpringWS]</i>	http://static.springframework.org/spring-ws/site/index.html
<i>[TestNG]</i>	http://testng.org/doc/
<i>[wikiJSF]</i>	http://en.wikipedia.org/wiki/JavaServer_Faces
<i>[XMLUnit]</i>	http://xmlunit.sourceforge.net/
<i>[Q4Eclipse]</i>	http://code.google.com/p/q4e/
<i>[ISO3166]</i>	ISO 3166 country code lists http://www.iso.org/iso/country_codes/iso_3166_code_lists.htm