

# JEE Security Structure Part 1

Ari Ayvazyan

29.09.2014

# Contents

<b>1</b>	<b>JEE Security Structure Part 1</b>	<b>2</b>
1.1	Introduction to Security Architecture . . . . .	2
1.2	Authentication . . . . .	3
1.3	Authorization . . . . .	3
1.4	Deployment Descriptors . . . . .	4
1.4.1	web.xml . . . . .	4
1.4.2	(vendor-specific).xml . . . . .	7
1.5	Principals . . . . .	7
1.6	Credential . . . . .	7
1.7	Groups . . . . .	7
1.8	Roles . . . . .	7
1.9	Realms . . . . .	7
1.10	Implementation sample . . . . .	9
1.11	Frameworks . . . . .	10
1.11.1	Shiro . . . . .	10
1.11.2	Spring . . . . .	10
1.11.3	JAAS - Java Authentication and Authorization Service	10
1.12	Output escaping . . . . .	10
1.13	Whats to come in Part 2 (Adrian) . . . . .	10

# Chapter 1

## JEE Security Structure Part 1

### 1.1 Introduction to Security Architecture

Most web applications have a few things in common:

They need to figure out who is the user that is using the application and what is he allowed to do and see.

A typical application has more than one security layer, it may be protected by only being available from a specified network or VPN. In addition there usually is some kind of identity determination followed by a SQL user with permission to query only the required functions and data sets.

On top of this, there should be output escaping to ensure that a attacker, who is able to manipulate the output for users, is limited in the harm he is able to cause.

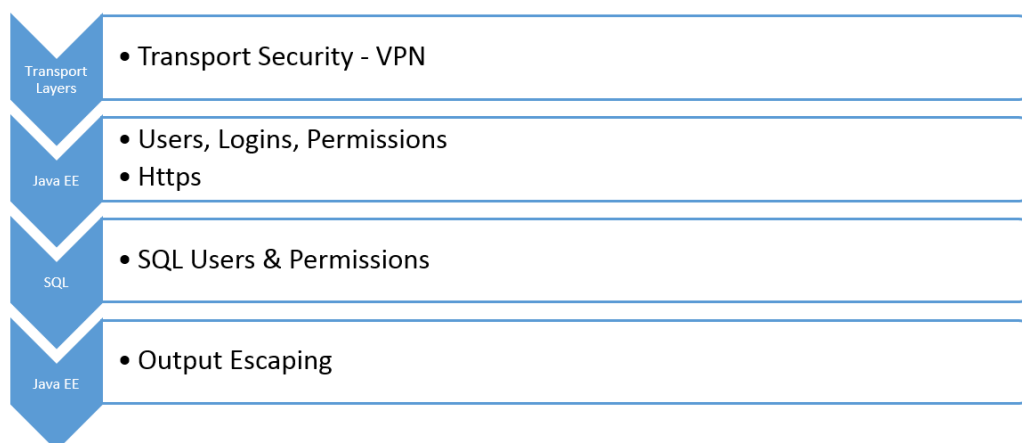


Figure 1.1: Security Layers in a common JEE application

According to Oracle[4], there are two ways to implement such access control functionality with Java EE:

1. Programmatic
2. Declarative (this includes Annotations and XML-Files)

While the programmatic implementation offers a wider range of customization, the declarative provides a well structured and easy to use approach.

## 1.2 Authentication

Authentication describes the identification process. This is mostly done by asking for a user-name & password or sending a Token/Hash.

## 1.3 Authorization

Authorization is what happens after you are authenticated. It deals with the question of what a authenticated person is allowed to do. Authorization may be applied to URLs or resources like Beans and Servlets.

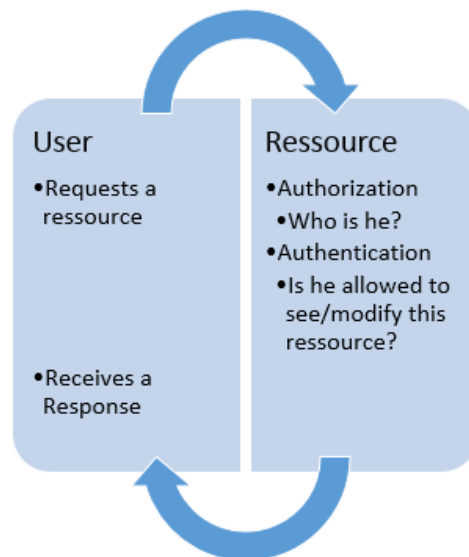


Figure 1.2: Authentication & Authorization

## 1.4 Deployment Descriptors

A Deployment Descriptor describes how a Java EE application should be deployed.

They contain information about security constraints, accessibility and resource references.

Deployment Descriptors are XML-Files that are by default located in the /WEB-INF/ directory.

The following deployment descriptors may be found here:

- web.xml
- <vendor-specific>.xml (E.g. when using Glassfish: glassfish-web.xml)

### 1.4.1 web.xml

The Web.xml file stores apart from usual deployment information like servlet mappings also security related information about:


- Protected Resources
- Security Roles
- Authentication methods

The following XML snippets are located within the <web-app></web-app> tag.

## Protected Resources

It is possible to limit access to resources by defining a security constraint on the URL or by securing the resource itself.

E.g. to protect the /primes/ URL with all its subdirectories we would have to use the following code:



### Securing a URL

```
<security-constraint>
  <web-resource-collection>
    <web-resource-name>primes
    </web-resource-name>
    <!--Include /primes/ including all following subfolders-->
    <url-pattern>/primes/*</url-pattern>
    <!-- This would result in a security leak because
         there are more http-methods than GET and POST-->
    <!-- by defining no http-method at all,
         everything will be blocked-->
    <http-method>GET</http-method>
    <http-method>POST</http-method>
  </web-resource-collection>

  <auth-constraint>
    <role-name>view_role</role-name>
  </auth-constraint>
</security-constraint>
```

In result, only a user with the role view\_role is allowed to access the defined resources.

## Security Roles

A Security Role was used in the last subsection "Protected Resources", we declared that only users with a `view_role` are allowed to view the restricted URL.

A Security Role is an abstract layer in front of the container, it defines an identifier which we can use for constraints. This identifier is then used by the container to specify its meaning by telling who is part of this Security Role.



### Defining a Security Role

```
<security-role>
  <description>This role has view access</description>
  <role-name>view_role</role-name>
</security-role>
```

## Authentication Methods

There are several kinds of Authentication Methods with different security behaviors.

BASIC Authentication opens a login prompt when a user tries to access the secured URL, it is simple to implement but insecure. BASIC Authentication sends the user's credentials unencrypted to the server.



### Defining BASIC Authentication

```
<login-config>
  <auth-method>BASIC</auth-method>
  <realm-name>Java EE Login</realm-name>
</login-config>
```

### 1.4.2 (vendor-specific).xml

Most containers use in addition to the web.xml a vendor specific XML file that is usually located in the same directory with web.xml.

E.g. Glassfish calls this file "glassfish-web.xml" while Tomcat has named it "context.xml".

The following settings can be configured there:

- User – Role mapping
- Group – Role mapping
- Other container specific configuration.

## 1.5 Principals

A Principal is a identity that can be authenticated.

E.g. a Unique user name

## 1.6 Credential

A Credential is defined as information that is used to authenticate a Principal.

E.g. a Password

## 1.7 Groups

Groups and Principals can be mapped to Roles.

Groups are defined in the vendor-specific.xml

## 1.8 Roles

Permissions are granted to Roles.

Roles are defined in the web.xml file



## 1.9 Realms

aka Security policy domain

Provides information about principals, their Groups and their credentials

May be a Database, File structure, connection. . .

In other words:

It contains user information

E.g. Username, Password & Permissions

## 1.10 Implementation sample

<https://github.com/aayvazyan-tgm/JavaEESecurityExample>

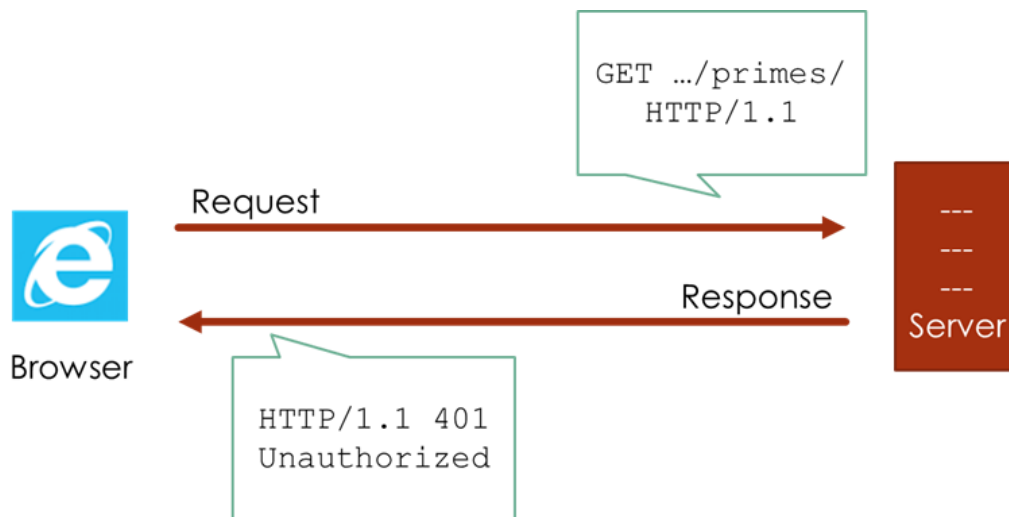


Figure 1.3: The user tries to access a resource without authentication

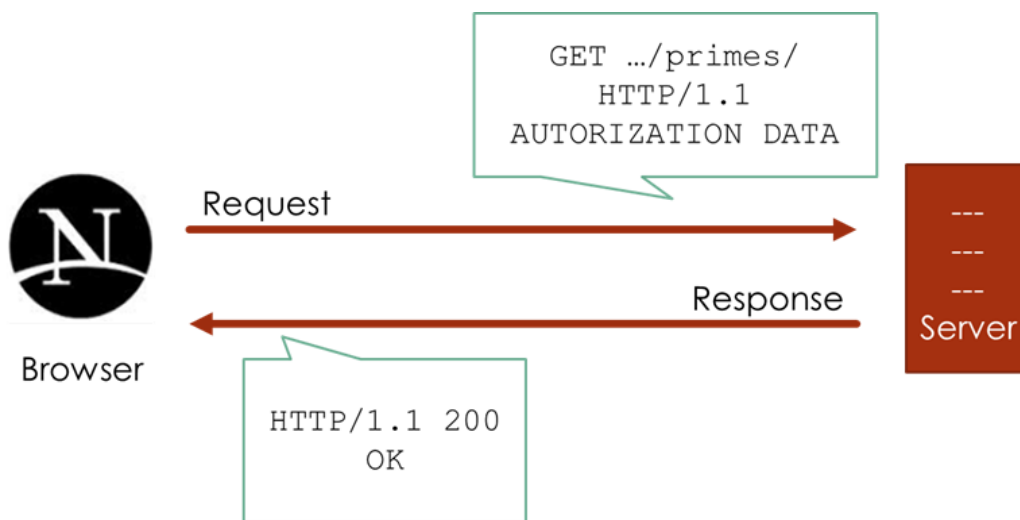


Figure 1.4: The user sends authentication data with his request

## **1.11 Frameworks**

### **1.11.1 Shiro**

Offers: Authentication, Authorization, Cryptography  
Simple to use

Advantages/Disadvantages  
Implementation Sample

### **1.11.2 Spring**

Offers: Authentication, Authorization, Cryptography  
Very structured

Advantages/Disadvantages

### **1.11.3 JAAS - Java Authentication and Authorization Service**

Offers: Authentication, Authorization, Cryptography  
Included in Java SE since Java 1.4 (javax.security.auth)

Advantages/Disadvantages

## **1.12 Output escaping**

Escape user input to prevent injections.

Escape the output to add a extra layer of security.  
Use a Framework to do so!

## **1.13 Whats to come in Part 2 (Adrian)**

- Working with Digital Certificates

- Securing Application Clients
- Security with Enterprise Beans
- Further Framework Information

# Bibliography

- [1] JavaOne 2014: The Anatomy of a Secure Web Application Using Java,  
Shawn McKinney & John Field, September 29, 2014  
San Francisco
- [2] Java Security: Sicherheitslücken identifizieren und vermeiden,  
Marc Schönefeld, 1. edition 2011  
Publisher: Hüthig Jehle Rehm GmbH, Heidelberg.  
ISBN/ISSN 978-3-8266-9105-8
- [3] Enterprise Java Security: Building Secure J2EE Applications,  
Marco Pistoia, Nataraj Nagaratnam, Larry Koved, Anthony Nadalin,  
1. edition 2004  
Publisher: Addison-Wesley Professional.  
ISBN/ISSN: ISBN 0-321-11889-8
- [4] Official JavaEE Documentation, Oracle,  
29.09.2014 <http://docs.oracle.com/javaee/7/tutorial/partsecurity.htm#GIJRP>  
Java EE 6,  
Dirk Weil, 1. edition 2012  
Publisher: entwickler.press  
ISBN 978-3-86802-077-9
- [5] Java EE 6 Cookbook for Securing, Tuning, and Extending Enterprise  
Applications,  
Mick Knutson,  
1. edition June 2012  
Publisher: Addison-Wesley Professional.  
ISBN/ISSN: ISBN 9781849683166