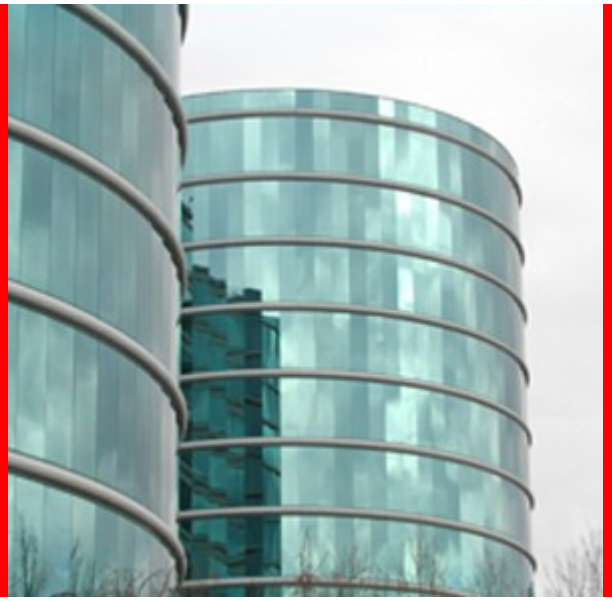


Hudson Security Architecture

ORACLE

ORACLE THIRTY YEARS 30

Winston Prakash

Hudson Security Architecture

Hudson provides a security mechanism which allows Hudson Administrators to control areas of access to users or group of users. The key definitions are:

Authentication - Determines the identity of a user or roles and represents the Authenticated mode of the user o roles.

Authorization/Permission - Determines what resources can be accessed or what actions can be executed by an authenticated user or process.

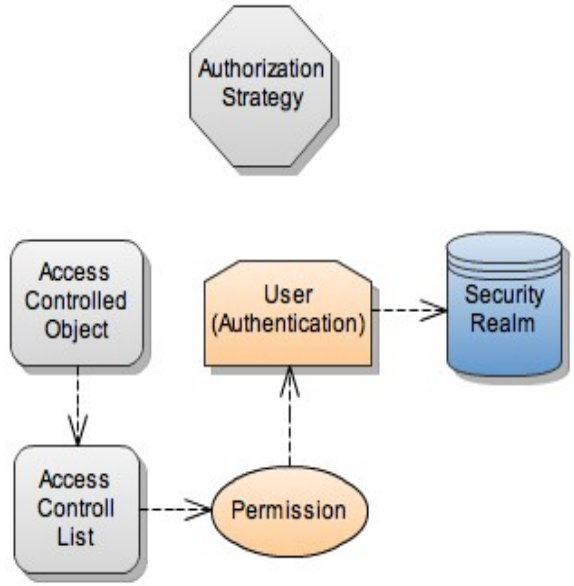
Role - Represents a set of functional responsibilities and specific permissions. Users and groups are assigned to roles to authorize these permissions.

AccessControlled – An object that has constraints defined by Permission object.

ACL – a list of Permissions attached to an **AccessControlled** Object.

SecurityRealm – represents the scope of the security data from which the users are authenticated.

AuthorizationStrategy – a strategy with which the Hudson security is enabled.



Authorization Strategy

Authorization strategy dictates how Hudson areas are granted access.

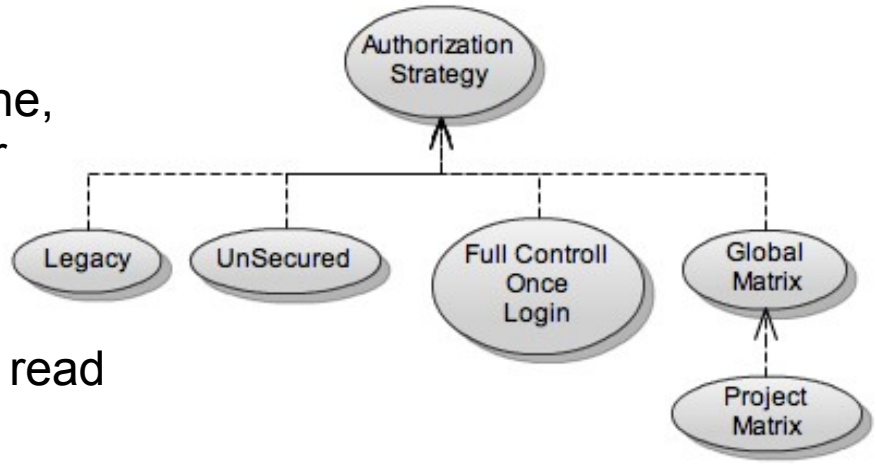
Unsecured – In this default mode, no security imposed, any one can do anything

Legacy – This strategy grants read access to every one, except a username “admin” who gets full administrator privilege.

Full Control Once Login – This strategy grants full Administrator privilege to any who logs in, rest all gets read only privilege.

Global Matrix – A matrix of privileges are retained by this strategy to grant access Privileged areas are granted access based on the strategy defined in this matrix.

Project Matrix – Each project can include its own subset of matrix based privileges.



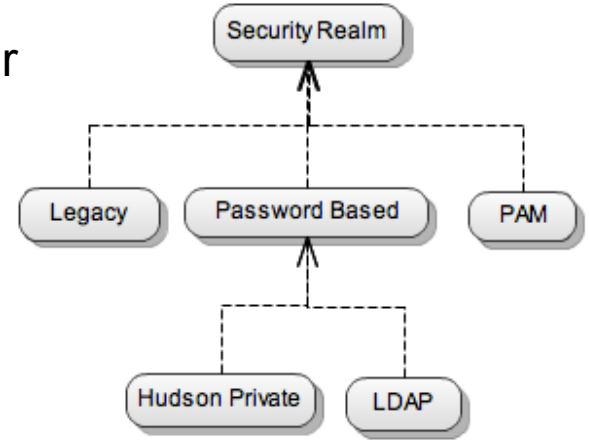
Security Realm

A Realm provides a collection of users and groups (user database) that are controlled by the same authentication policy.

Legacy – Uses Container based Authentication. Assumes the user is already authenticated by container.

Password Based – Uses Username/Password based authentication. Two implementations are

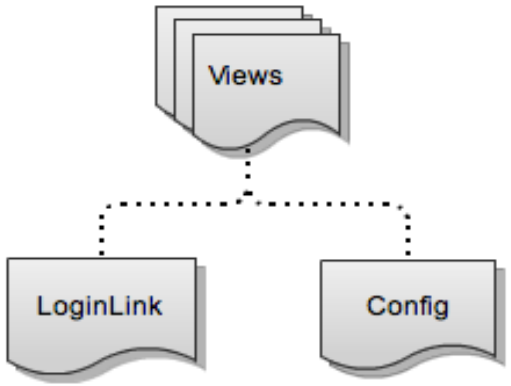
- Hudson Private – Uses internal database to store username & password
- LDAP based authentication



PAM – Uses Unix Pluggable Authentication Module for authentication.

Security Realm contributes to view via two files

- **LoginLink**.jelly – Adds the login/logout link to the dashboard
- **Config**.jelly – Provides UI to configure the page



Access Control List

When a user requests an operation on a Model object, if it is a access controlled, the Access Control List (ACL) is checked for an applicable entry to decide whether the requested operation is authorized.

Hudson uses a **Security identifier (Sid)** as a key to map the Permissions.

The Authorization Strategy adds Map Entry with a mapping between Sid and Permission to ACL.

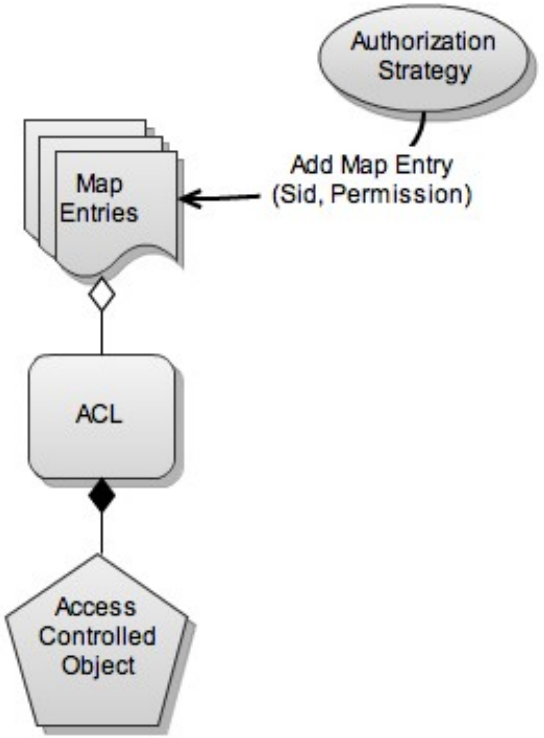
Authorization Strategy like **Matrix** Provides UI for users to add the Role based Permission Mapping.

Full Control Once LoggedIn Authorization Strategy adds the Entries as

```
(EveryOne, Administrator, true)
(Anonymous, Administrator, false)
(Anonymous, Read, true)
```

Legacy Authorization Strategy adds as

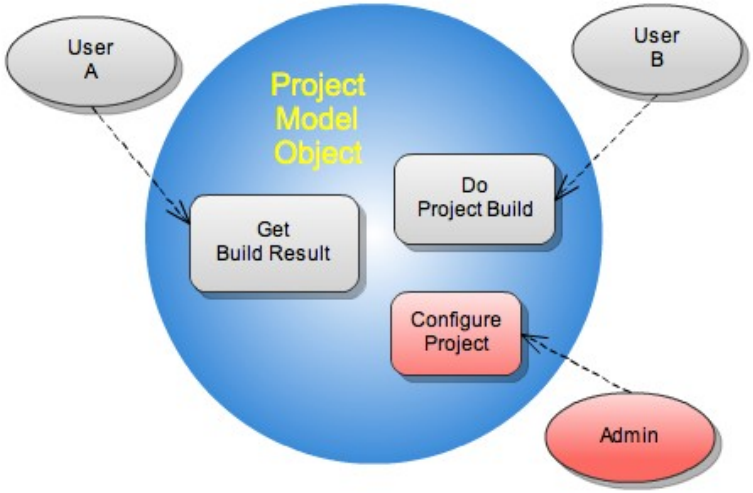
```
(Anonymous, Read, true)
("admin", Administrator, true)
```



Authorization

When Security is enabled, with out proper authorization, certain areas of hudson can not be accessed. The authorization is governed by authorization Strategy.

There are three entities that are governed by authorization



- Hudson Model Objects
  Some portion or all of the model Object can not be executed, with out proper authorization.

- Hudson View elements
  Some portion of the view will show only when the logged in user has appropriate access.

- Hudson Web pages
  In this case the page itself can not be viewed by unauthorized user
  Ex. hudson configuration page

Authorizing access to Hudson Objects

When Security is enabled, certain Operations on Hudson Model Objects and Extension Points are given access based on the authorization strategy.

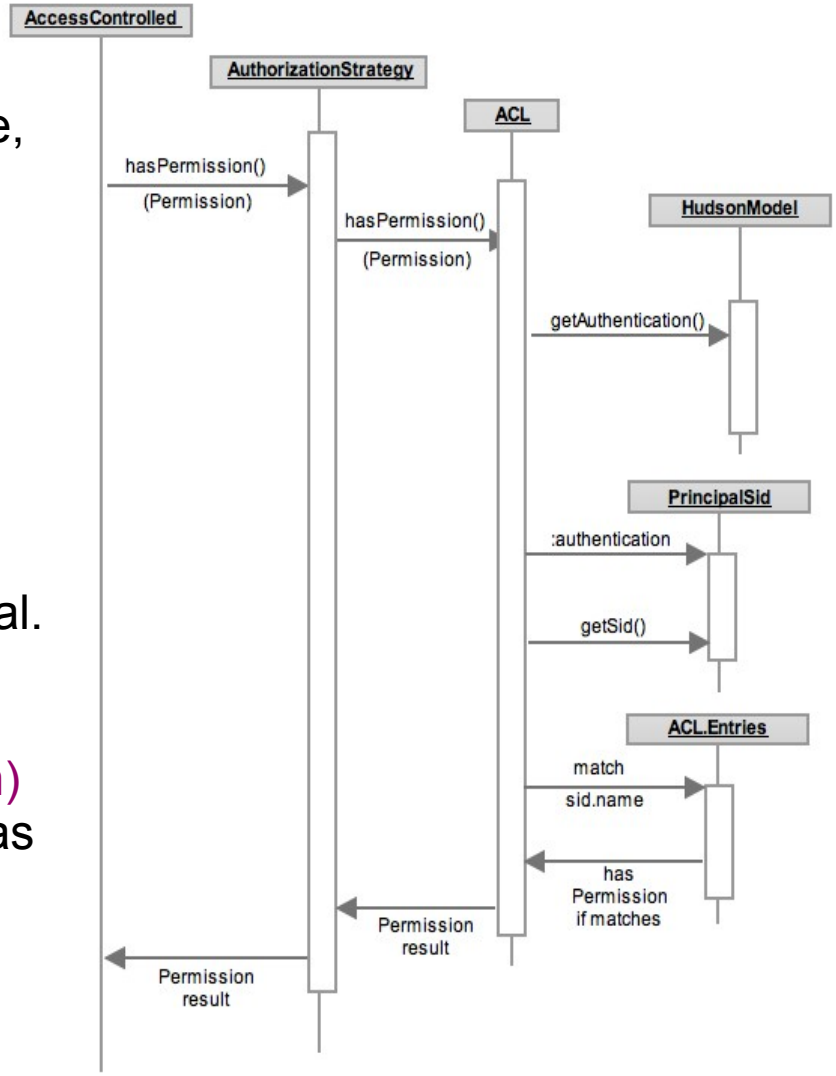
When a method or block of statement is executed, **hasPermission(Permission)** is invoked. For example, in the project Object

```
public void configure() thows
                    NoPrevilegeException{
    hasPermission(Hudson.ADMINISTRATOR)
    // Do the configuration
}
```

The permission is checked against the current principal.

`hasPermission()` method depends on AuthorizationStrategy. The entry map (sid, Permission) of the ACL object is iterated to see if the current sid has the requested Permission.

The sid itself is checked for authentication before checking for authorization.

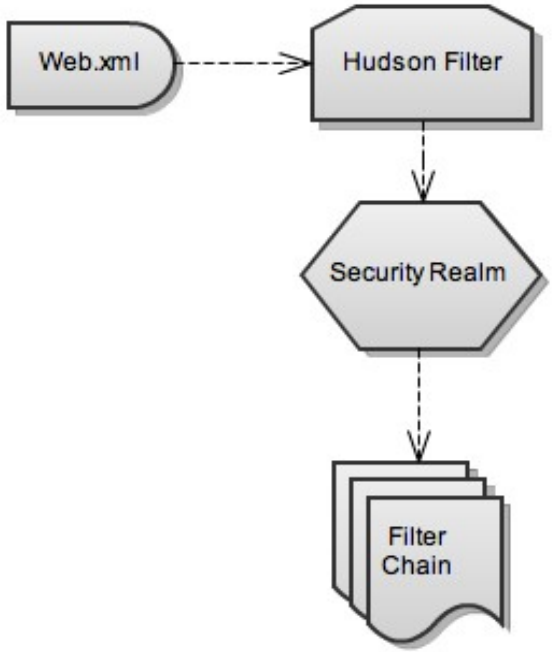


Authorizing access to Hudson Pages

When Security is enabled, Hudson web pages are given access based on the authorization strategy that is in effect. The Page filtering is done via Java EE Filter mechanism. A servlet Filter called **HudsonFilter** specified in the web.xml to do the job.

If a security realm is specified then get the Filter Chain from it. Invoke the filter chain with the specific Servlet Context.

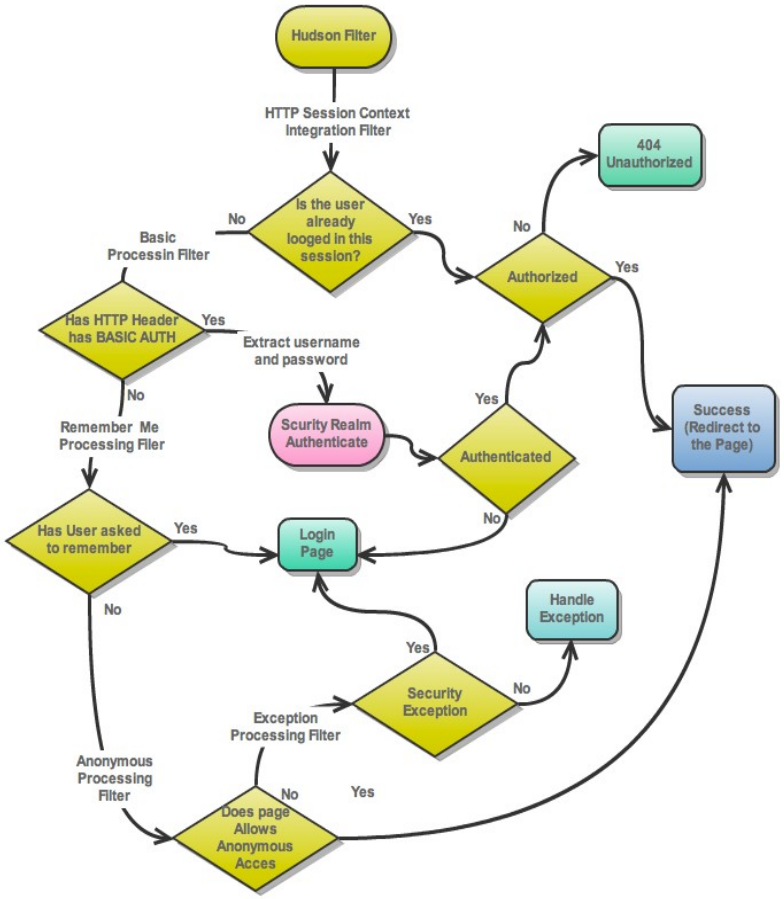
If Security Realm is not set, then simply redirect to the Filter chain supplied by the container itself.



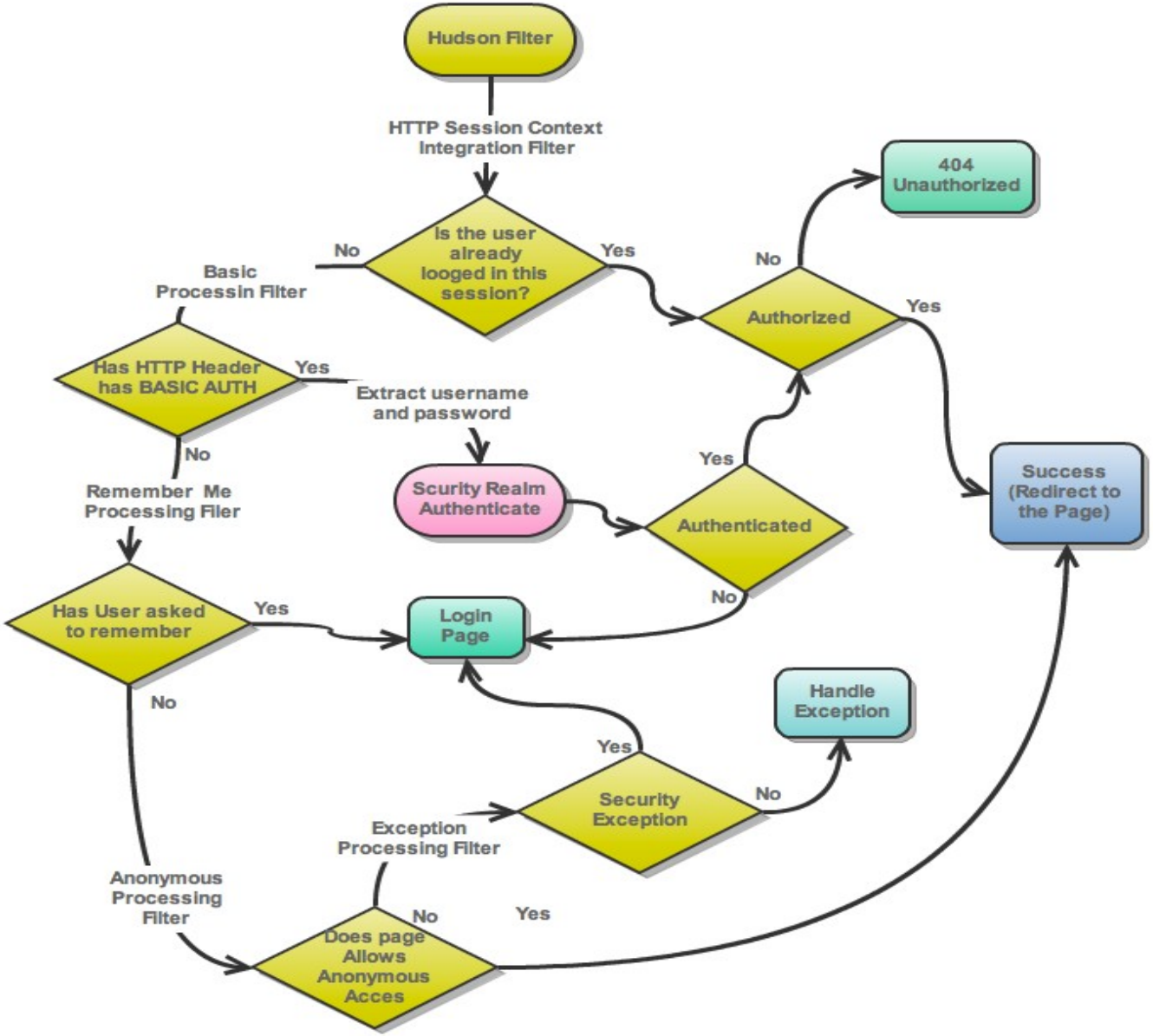
Filter Chain

The Web pages are filtered in the following order in the chain of filters.

- **HTTP Session Context Integration Filter** – Responsible for handling the authentication with in a Session.
- **Basic processing Filter** – Handles the HTTP BASIC Authentication, sent via HTTP header (especially command line request)
- **Authentication processing Filter** – Form based authentication happens in this filter
- **RememberMe Processing Filter** – Send the remembered user name and password to the login page
- **Anonymous Processing Filter** – This filter checks if the page allows anonymous access based on Authorization Strategy.
- **Exception Translation Filter** – Handles if exception happens during Authentication and Authorization (non security related)
- **Unwrap Security Exception Filter** – Sends the user to appropriate login page if security related exception happened.



Filter Chain invocation



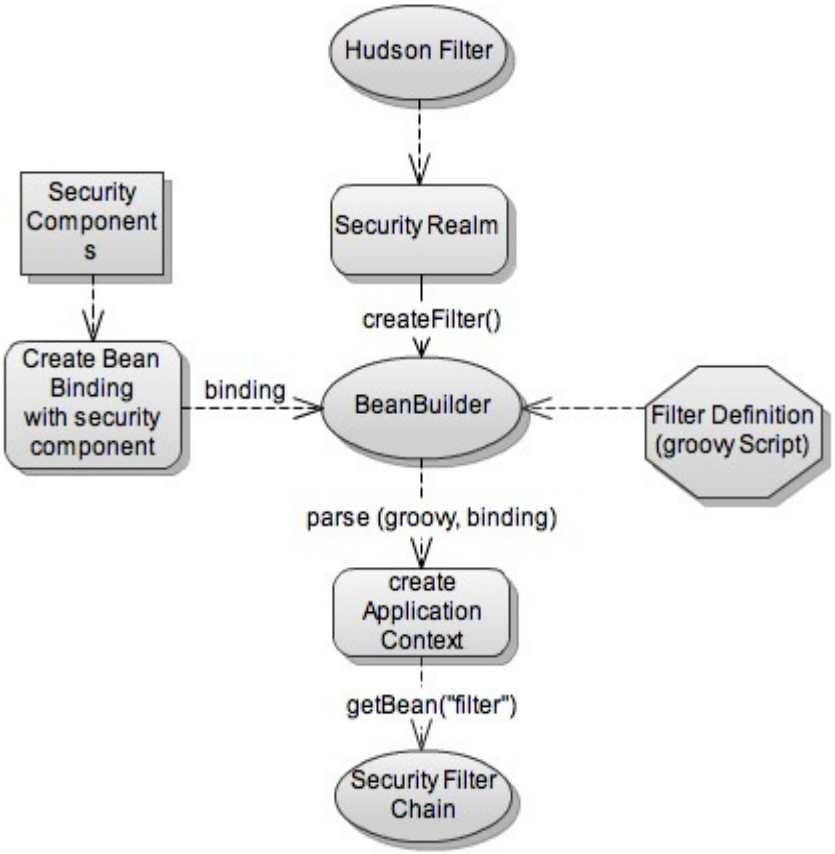
Specifying Security Filters

The filter chain is obtained from Security Realm. It is up to the security Realm to create the filter chain. Hudson provides a easy way to create hudson using Spring security architecture on the fly.

Hudson defines a tuple with which the Filter chain provider can inject some of the security components such as

- Authentication manager
- User Details
- RememberMe service
- Failure URL
- Filter Process URL
- Target URL

The definitions of security filter itself is defined in a groovy script. Spring Bean Builder service is used to create the corresponding Filters beans.



Filter Chain Groovy Script

```
filter(ChainedServletFilter) {
    filters = [

        bean(AuthenticationProcessingFilter2) {
            authenticationManager = securityComponents.manager
            rememberMeServices = securityComponents.rememberMe
            authenticationFailureUrl = "/loginError"
            defaultTargetUrl = "/"
            filterProcessesUrl = "/j_acegi_security_check"
        },
        bean(RememberMeProcessingFilter) {
            rememberMeServices = securityComponents.rememberMe
            authenticationManager = securityComponents.manager
        },
    ]
}
```

Authentication Filter Processing

When the page is accessed by User, the Filter.doFilter() method first checks if Authentication and Authorization is required. This is done via Authorization Strategy Object.

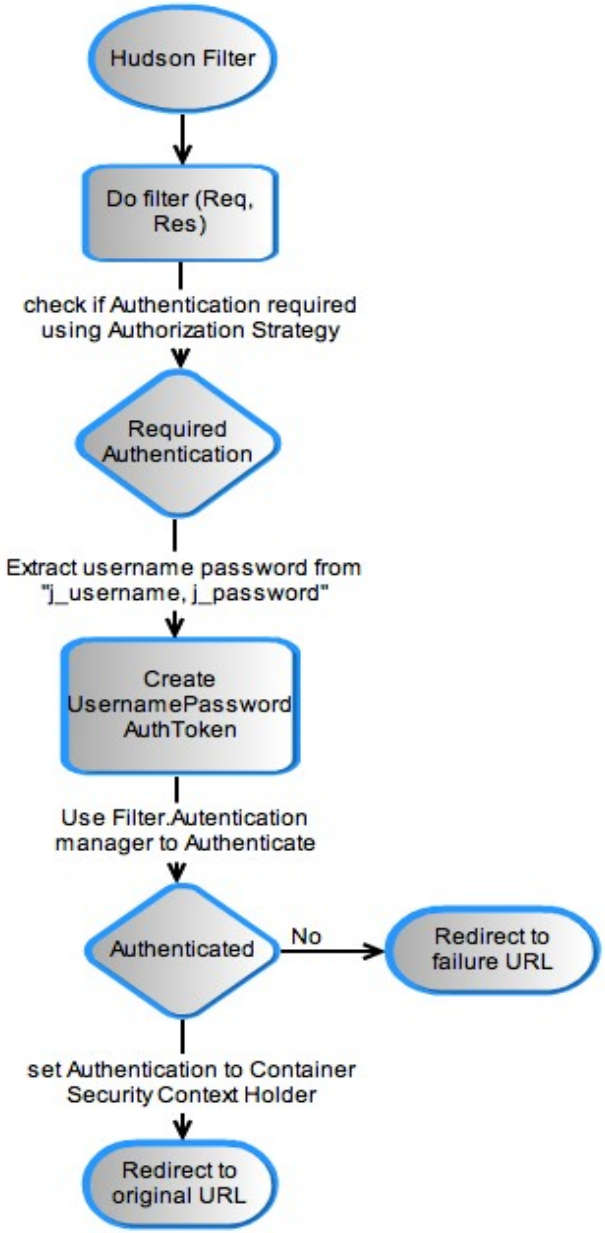
If required, the username and password is extracted from **j_username** and **j_password** Servlet Context Parameters.

An **UserNamePawwordAuthToken** is created from username and password and send to the Authentication manager specified in the Filter for Authentication.

If Authentication , the Authentication Object is set to the **Container Security Holder** for the container filter chain to know that the Authentication is successful

Then the the Servlet Context is redirected to the original URL.

If Authentication fails then Servlet Context is redirected to the Failure page, usually the Login page.



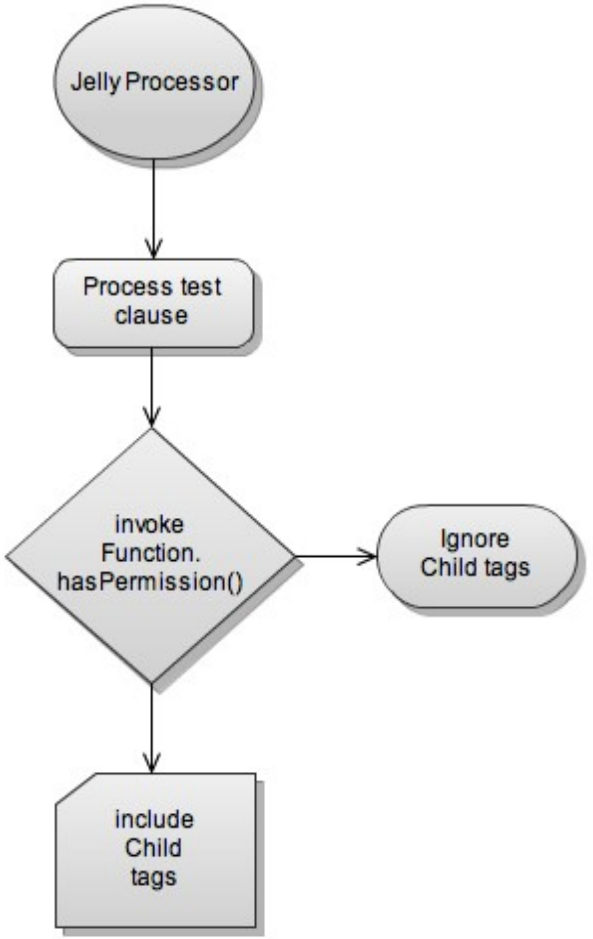
View Element Authorization

The view provided by model Objects or Extension Points can render portion of the view based on certain authorization. Ex. Project Dashboard would display the Configure Project only if the user is authorized to configure the project.

The elements of the jelly code that need to be access protected need to be surrounded with a test jelly clause

```
<j:if test="${h.hasPermission(it, permission)}">
    <d:invokeBody/>
</j:if>
```

If the current principal has given authorization, then the child elements will be included in the rendering of page.



Changing Authorization Strategy and Security Realm

Authorization Strategy and Security Realm are Extension Points. So multiple entities of each can exist in the Hudson Platform. Hudson deployer has option to choose one of each via Hudson Configuration Page.

The **AuthorizationStrategy** object (defaults to Unauthorized) is held by the Hudson Model Object. When user selects different Authorization Strategy, selected one replaces the earlier one.

The **SecurityRealm** object is held by the HudsonFilter. When Hudson model process the submitted config page, it sets the newly selected Security Realm

When a new Authorization Strategy set, the ACL will be set with the new authorization (sid, Permission) map entries.

When new SecurityRealm is set, previous Authentication will be no longer valid. User need to authenticate with proper security credentials again.