Simplicity *itself*

Grails & Spring Security

Built in security

- Protection against:
  - SQL injection
  - Cross Site Scripting (XSS)
  - Cross Site Request Forgery (CSRF)
- These are in top ten OWASP web vulnerabilities

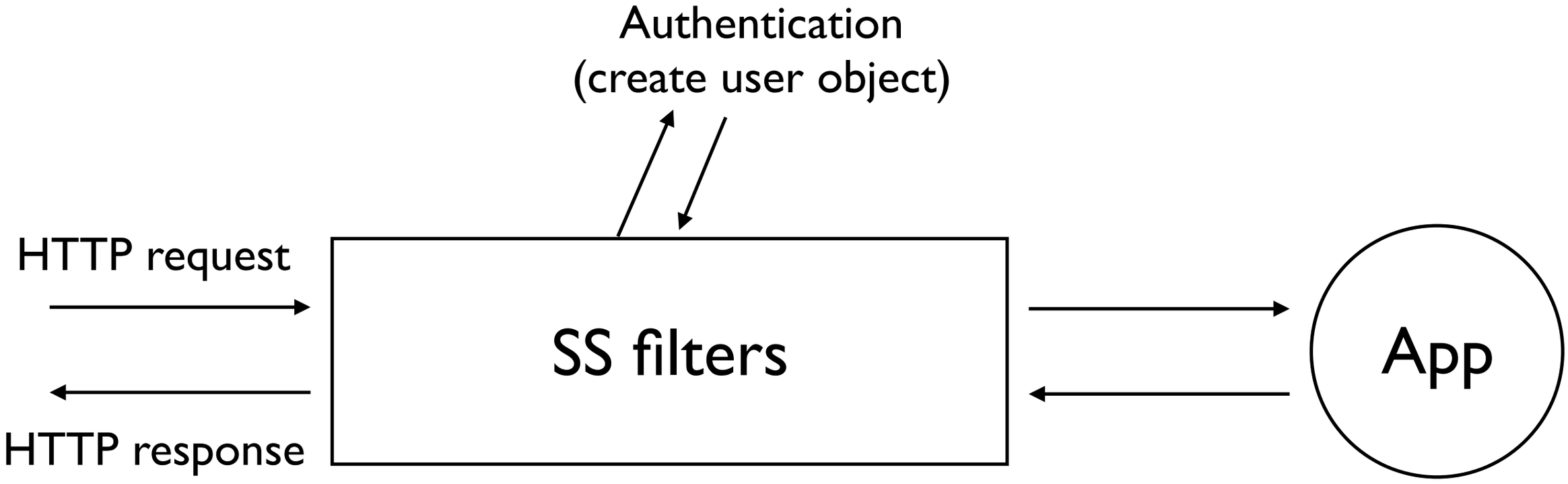
But no access control

Banner uses the Spring Security plugin for Grails

Spring Security plugin provides

- Authentication
  - Form-based, HTTP Basic authentication and more
- Access control
  - Roles (*authorities* in Spring Security core)
  - Access Control Lists (ACLs)

Spring Security filter chain



The SS filter chain is wrapped by a standard Servlet filter

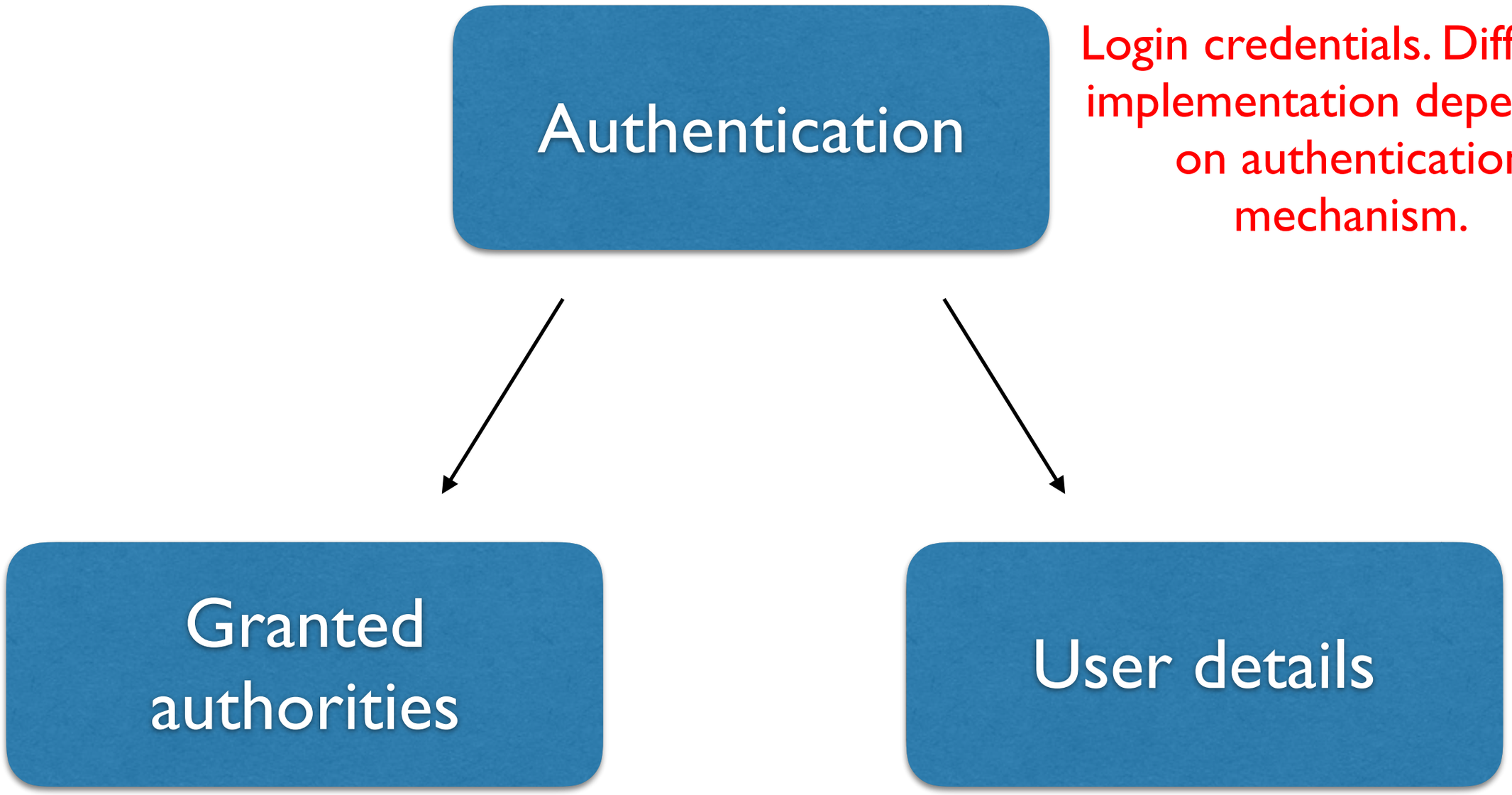
Example SS filters

- Logout
- Anonymous user
- Username/password authentication
- Remember me
- Attach security context to request thread
- Authority checks

Filters can trigger on different URLs:

Logout	→	/j_spring_security_logout
Authentication	→	/j_spring_security_check
Anonymous	→	/**
Authority check	→	/**

Other core objects



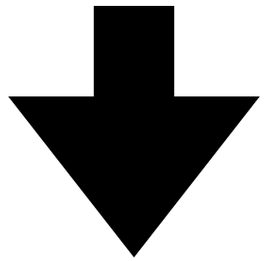
Login credentials. Different implementation depending on authentication mechanism.

Roles/permissions that authenticated user has

Arbitrary user info provided by the auth implementation. E.g. full name & email address

Getting Started

```
grails s2-quickstart org.example User Role
```



- Creates `User` and `Role` domain classes
  - provide authentication & access control
- Creates login & logout controllers

Getting Started

- Add any extra properties to User you want
  - e.g. full name, email address, etc.
- Decorate or replace the login page
- Lots of config options
  - access control rules
  - login form URL

The bare minimum

Add access rules for URLs

- via annotations on controllers/actions
- via static URL rules
- via URL rules stored in the database (`Requestmap`)

Annotations

```
import grails.plugin.springsecurity.annotation.Secured

@Secured(['IS_AUTHENTICATED_ANONYMOUSLY'])
class MyController {

    @Secured(['ROLE_USER'])
    def secret() {
       ...
    }

    @Secured(['IS_AUTHENTICATED_FULLY'])
    def home() {
        ...
    }
}
```

URL rules (Config.groovy)

```
g.p.ss.securityConfigType = "InterceptUrlMap"
g.p.ss.interceptUrlMap = [
    '/':                ["permitAll"],
    '/assets/**':       ["permitAll"],
    '/login/**':        ["permitAll"],
    '/logout/**':       ["permitAll"],
    '/secret/**':       ["hasRole('ROLE_USER')"],
    '/**':              ["isAuthenticated()"]
]
```

(see plugin guide for dynamic rules)

Rules

Constant	Expression	Description
IS_AUTHENTICATED_ANONMOUSLY	permitAll	Anyone has access
IS_AUTHENTICATED_REMEMBERED	isAuthenticated()	Client must be authenticated or have a 'remember me' cookie
IS_AUTHENTICATED_FULLY	isFullyAuthenticated()	Client must be authenticated in this session
ROLE_USER	hasRole('ROLE_USER')	Custom role (must be added to database)

Other common config options

```
g.p.ss.userLookup.userDomainClassName = "org.example.User"
g.p.ss.userLookup.authorityJoinClassName =
        "org.example.UserRole"
g.p.ss.authority.className = "org.example.Role"

g.p.ss.userLookup.usernamePropertyName = "loginId"
g.p.ss.userLookup.passwordPropertyName = "passwordHash"

g.p.ss.auth.loginFormUrl = "/login/form"
g.p.ss.successHandler.defaultTargetUrl = "/"
```

Is the user authenticated?

via a service method:

```
springSecurityService.isLoggedIn()
```

via a GSP tag:

```
<sec:ifLoggedIn>
    ...
</sec:ifLoggedIn>
```

Who's logged in?

via a service method:

```
springSecurityService.currentUser
springSecurityService.authentication
springSecurityService.principal
```

Domain instance

Token

UserDetails object

via a GSP tag:

```
<sec:username/>
<sec:loggedInUserInfo field="fullName"/>
```

Has access?

via a utility method:

```
SpringSecurityUtils.ifAllGranted("ROLE_USER")
SpringSecurityUtils.ifAnyGranted("ROLE_USER")
SpringSecurityUtils.ifNotGranted("ROLE_USER")
```

via a GSP tag:

```
<sec:ifAnyGranted roles="ROLE_USER">
    ...
</sec:ifAnyGranted>
<sec:access expression="hasRole('ROLE_USER')">
    ...
</sec:access>
```

Banner...

...does things differently

Banner overview

- Security in *banner_core* plugin
- Custom authentication provider
- Custom authentication token
- Custom access voter
- No domain classes

Seems to be based on Oracle Forms security

Core configuration

```
formControllerMap = [
        'commonmenu':               ['GUAGMNU'],
        'medicalinformation':       ['GOAMEDI'],
        'informationtexteditor': ['GUAINFO'],
        ...
]
```

```
g.p.ss.securityConfigType = "InterceptUrlMap"
g.p.ss.interceptUrlMap = [
    '/':                ["permitAll"],
    ...
    '/**':              ["ROLE_DETERMINED_DYNAMICALLY"]
]
```

Good luck!