Simplicity *itself*

Messages in Grails

Why?

Localisation (L10n)

Keep display text out of code

Based on

Java Resource Bundles

Java `Locale` *class*

Resource bundle

messages.properties:

```
hello.world=Hello World!
book.title.label=Book title
```

Message key

Default text for this key

Resource bundle

messages_fr.properties:

```
hello.world=Bonjour le Monde!
book.title.label=Titre de livre
```

Message key

French text for this key

Resource bundle

Can be parameterised:

```
greeting=Hello {0}!
```

File names

messages.properties	*Fallback text if no bundle matches requested locale*
messages_fr.properties	*Generic French text*
messages_fr_CA.properties	*French Canadian text*

<bundle name>_<lang>_<country>.properties

Locales consist of

Two-letter language code

en, fr, zh, es, de, etc.

Two-letter country code (optional)

GB, CA, US, PT, BR, etc.

Rules

Requested locale: es_CL

Matches (highest priority first):

messages_es_CL.properties

messages_es.properties

messages.properties

Rules

Requested locale: es

Matches (highest priority first):

messages_es.properties

messages.properties

How is the current locale determined?

In Grails

- `lang` URL/form parameter
  - stored in session
- *Accept-Language* HTTP header
- Default JVM locale (may come from platform locale)

For example

http://example.org/my-app/home?lang=fr

Fetch request locale

```
import org.springframework.web.servlet.support.
RequestContextUtils as RCU

RCU.getLocale(request)
```

How do I fetch messages?

GSP tag

In GSP

```
<g:message code="hello.world"/>
<g:message error="${book.errors.getFieldError('title')}"/>
```

In controller

```
g.message code: "hello.world"
```

Bean

```
import org.springframework.context.MessageSource

class MyService {
    MessageSource messageSource
    List greeting(String name, Locale locale) {
        return messageSource.getMessage(
                "greeting",
                [name] as Object[],
                locale)
    }
}
```

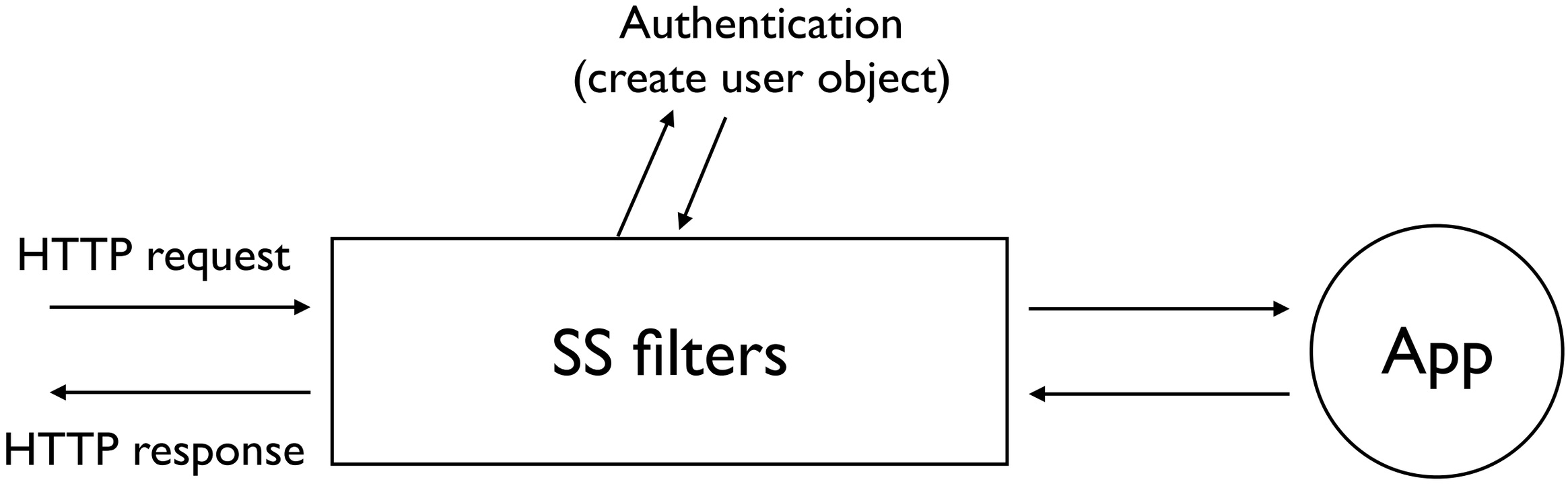
Grails resource bundles have a base name of *messages* and are stored in the *grails-app/i18n* directory.

Internationalisation

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

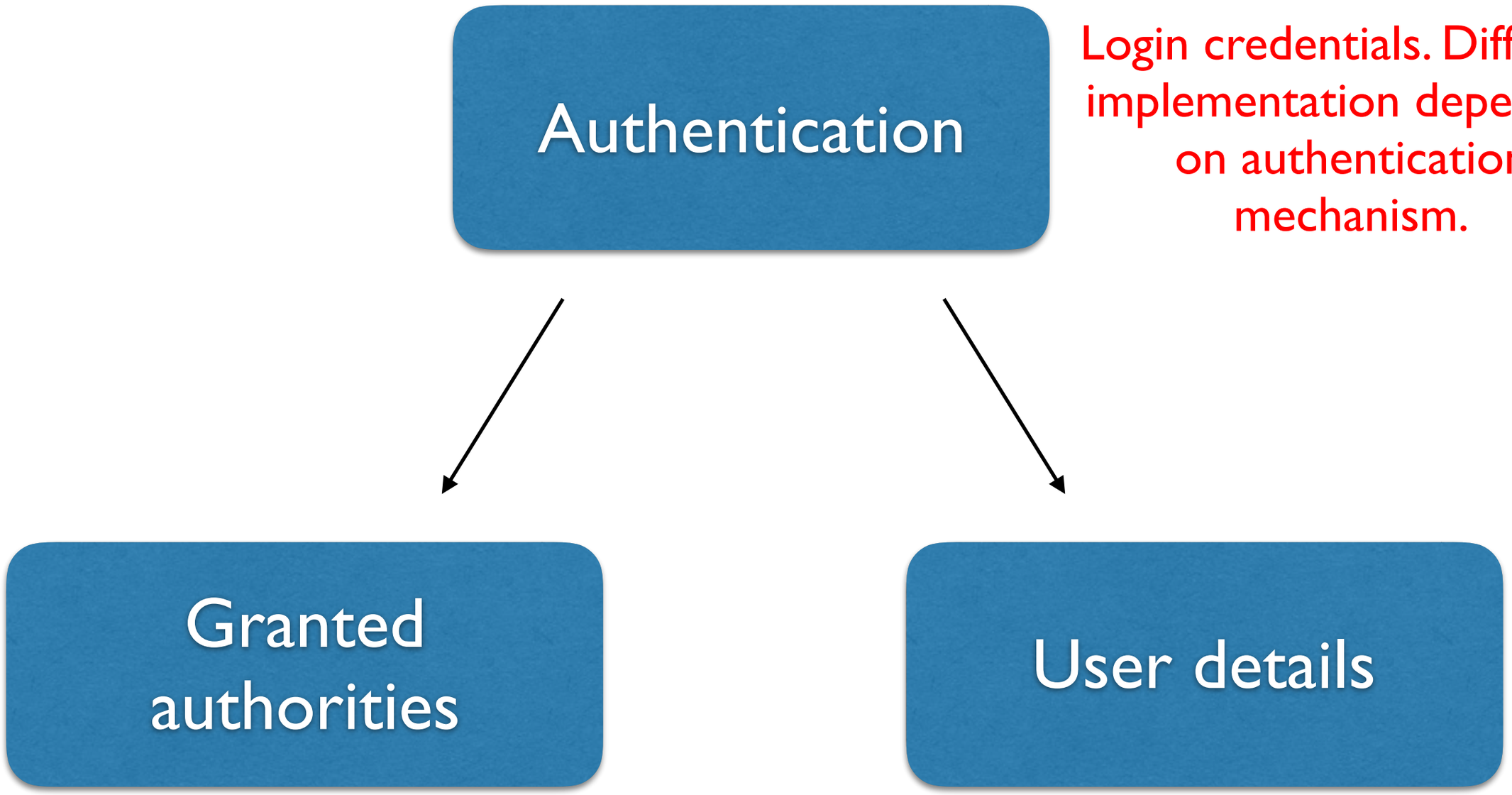
Example SS filters

- 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



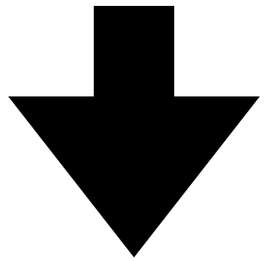
Login credentials. Different implementation depending on authentication mechanism.

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!