Grails

- Links
  - Official Website: `http://grails.org/`
  - Latest Docs: `http://www.grails.org/doc/latest/`
  - Mailinglist: `http://grails.1312388.n4.nabble.com/Grails-user-f1312389.html`
  - Various Blogs and Google
- Books:
  - Beginning Groovy and Grails
  - Grails in Action
  - The Definitive Guide to Grails

Project: Twidder

Ingredients

- Netbeans IDE
- Grails 1.3.5
- Grails Plugins:
  - Spring Security Core Plugin (`http://grails.org/plugin/spring-security-core`)
  - Jasig CAS support for the Spring Security plugin (`http://grails.org/plugin/spring-security-cas`)
  - LDAP Plugin (`http://grails.org/plugin/ldap`)
    Not using (sic!) Spring Security LDAP Plugin

Build Basic Application

1. New Grails Project in Netbeans or `grails create-app` named Twidder
2. New domain class: Message

```
1  package twidder
2
3  class Message {
4    String content
5
6    Date time
7
8    static constraints = {
9      content(blank:false, size:0..140)
10   }
11
```

```
    String toString() {
13    "${content} at ${time}"
    }
15 }
```

3. Rightclick on domain class Message → Generate All or `grails generate-all`
   This generates controller, views and testclasses.

4. Setting `time` in controller:
   Add following in save action of MessageController:

```
1 def save = {
      def messageInstance = new Message(params)
3
      messageInstance.time = new Date()
5     [...]
  }
```

5. Delete time DropDown fields in view

6. Build "nicer"views:
   layouts/main.gsp:
   Replace grailsLogo with

```
  <div id="grailsLogo"><h2>Twidder</h2></div>
```

   index.gsp:

```
1 <html>
      <head>
3         <title>Twidder</title>
          <meta name="layout" content="main" />
5     </head>
      <body>
7       Welcome to Twidder. You can send 140 character message and see
            what others are writing!
        <ul>
9         <li> <g:link controller="message" action="create">Write
              Message</g:link>
          <li> <g:link controller="message" action="messages">Read own
              Message</g:link>
11        <li> Read from other Twidder users:
        </ul>
13    </body>
  </html>
```

7. Adding view: message/messages.gsp

```
<%@ page contentType="text/html;charset=UTF-8" %>
2
<html>
```

```
4   <head>
      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"
          >
6     <title>Own messages</title>
    </head>
8   <body>
      <h1>Own messages</h1>
10  </body>

12  <g:render template="/message/message" var="message" collection="${
        allMessages}" />
</html>
```

8. Add action to MessageController:

```
  def messages = {
2   def allMessages = Message.createCriteria().list(sort:'time', order
        :'desc') {}

4   return [allMessages : allMessages]
  }
```

9. Add view: message/_message.gsp:

```
1 <%@ page contentType="text/html;charset=UTF-8" %>

3 <p>${message.content} at ${message.time}</p>
```

Add Authentication using CAS and LDAP

1. Install Plugins:
   ```
   grails install-plugin spring-security-core
   grails install-plugin spring-security-cas
   grails install-plugin ldap
   ```

2. Mapping LDAP to Java attributes:
   src/groovy/ldap/LdapUser.groovy:

```
1 package ldap

3 import gldapo.schema.annotation.GldapoNamingAttribute
  import gldapo.schema.annotation.GldapoSynonymFor
5
  /**
7  * LDAP User Domain class
   * Represents the user from LDAP directory
9  *
   * @author Dominik Schuermann
11 */
  class LdapUser {
13
```

```
    /** "Primary Key" */
15  @GldapoNamingAttribute
    @GldapoSynonymFor("uid")
17  String username

19  String cn

21  @GldapoSynonymFor("sn")
    String lastname
23
    @GldapoSynonymFor("givenName")
25  String firstname

27  @GldapoSynonymFor("mail")
    String email
29
    @GldapoSynonymFor("ou")
31  String ldapRole

33 }
```

3. Generate default Spring-Security classes with included command: `grails s2-quickstart twidder SpringUser SpringRole`
   This generates default domain classes SpringUser and SpringRole in twidder package and corresponding controllers and views.

4. Configure plugins in Config.groovy:

```
1  /*
    * LDAP Plugin
3   */
   ldap {
5    directories {
       tubs {
7        url = "ldap://ldapk5.tu-bs.de:389"
         base = "ou=people,dc=tu-bs,dc=de"
9        userDn = ""
         password = ""
11       searchControls {
           countLimit = 40
13         timeLimit = 600
           searchScope = "subtree"
15       }
       }
17   }
     schemas = [ ldap.LdapUser ]
19 }

21 /*
    * Spring Security Configuration
23  *
    * http://burtbeckwith.github.com/grails-spring-security-core/docs/
        manual/index.html
```

```
25  */
    grails {
27    plugins {
        springsecurity {
29        // Classes:
          userLookup.userDomainClassName = 'twidder.SpringUser'
31        userLookup.authorityJoinClassName = 'twidder.
              SpringUserSpringRole'
          authority.className = 'twidder.SpringRole'
33
          // Spring Security CAS Plugin
35        // http://burtbeckwith.github.com/grails-spring-security-cas/
              docs/manual/index.html
          cas {
37          serverUrlPrefix = "https://cas.tu-braunschweig.de"
            loginUri = "/login"
39          proxyReceptorUrl = "/secure/receptor"
            key = "twidder"
41          sendRenew = false
            artifactParameter = 'ticket'
43          serviceParameter = 'service'
            filterProcessesUrl = '/j_spring_cas_security_check'
45          useSingleSignout = true
            // CAS URLs
47          proxyCallbackUrl = "${grails.serverURL}/secure/receptor"
            serviceUrl = "${grails.serverURL}/j_spring_cas_security_check"
49        }
          logout.afterLogoutUrl = "https://cas.tu-braunschweig.de/logout?
              url=${grails.serverURL}/"
51      }
      }
53  }
```

5. Create own User domain class inherited from SpringUser:
   User.groovy:

```
1   package twidder
    import ldap.LdapUser
3
    /**
5    * User domain class
     *
7    * Extends SpringUser class from Spring Security Plugin
     * extended by LDAP operations and project attributes
9    *
     * @author Dominik Schuermann
11   */
    class User extends SpringUser {
13
      static hasMany  = [messages:Message]
15
      /** User Real Name */
17    String firstname
```

```
    String lastname
19
    /** eMail */
21  String email

23
    static constraints = {
25    firstname(blank: false)
      lastname(blank: false)
27    email(nullable: false)
    }
29
    String toString() {
31    "$firstname $lastname"
    }
33
    /**
35   * creates new User by mapping the LDAPO class LdapUser
     *
37   * @param String username the username for the creating user
     * @return returns the new user object
39   */
    static createUserFromLdap(String username) {
41    // search for the user in LDAP directory
      def ldapUser = LdapUser.find(directory: "tubs", filter: "(uid=${
          username})")
43
      // create the user
45    def newUser = new User(
        username     : ldapUser.username,
47      firstname    : ldapUser.firstname,
        lastname     : ldapUser.lastname,
49      email        : ldapUser.email,
        password     : "usingcas", // not used because we use cas for
            authentication
51      enabled      : true
      ).save(flush: true, failOnError: true)
53
      // give default role to new user
55    def roleUser = SpringRole.findByAuthority('ROLE_USER')
      SpringUserSpringRole.create(newUser, roleUser, true)
57
      return newUser
59  }

61 }
```

6. Add role on Bootstrap init:

```
1   def init = { servletContext ->
      def userRole = twidder.SpringRole.findByAuthority('ROLE_USER') ?:
          new twidder.SpringRole(authority : "ROLE_USER").save(
          failOnError: true)
3   }
```

7. Add user on authentication to own database (extending Spring-Security, based on various tutorials):
   Services/MyUserDetailsService.groovy:

```
import twidder.User
import org.springframework.security.core.authority.
    GrantedAuthorityImpl
import org.springframework.security.core.userdetails.UserDetails
import org.springframework.security.core.userdetails.
    UserDetailsService

import org.codehaus.groovy.grails.plugins.springsecurity.
    GrailsUserDetailsService
import org.codehaus.groovy.grails.plugins.springsecurity.GrailsUser
import org.springframework.security.core.userdetails.
    UsernameNotFoundException

/**
 * Service that is used instead of GormUserDetailsService from Spring
     Security Plugin
 * It overrides in conf/spring/resources.groovy the Spring class
 *
 * It is used in conjunction with the Ldap Plugin to create Users from
     Ldap when they are not
 * in the local database
 *
 * @author Dominik Schuermann
 */
class MyUserDetailsService implements GrailsUserDetailsService {

  UserDetails loadUserByUsername(String username, boolean loadRoles)
      throws UsernameNotFoundException {
    return loadUserByUsername(username)
  }

  UserDetails loadUserByUsername(String username) throws
      UsernameNotFoundException {
    User.withTransaction { status ->

      User user = User.findByUsername(username)
      // if the user does not exist
      if (!user) {
        // create user from LDAP
        user = User.createUserFromLdap(username)

        if (!user) throw new UsernameNotFoundException('User not in
            LDAP', username)
      }

      def authorities = user.authorities.collect { new
          GrantedAuthorityImpl(it.authority) }
```

```
        return new GrailsUser(user.username, user.password, user.enabled
            ,
39          !user.accountExpired, !user.passwordExpired,
            !user.accountLocked, authorities, user.id)
41      }
    }
43  }
```

8. Overwrite bean in Config/spring/resources.groovy:

```
1  beans = {
     // using other Details Service for Spring Security to get user from
         ldap
3    userDetailsService(MyUserDetailsService)
   }
```

9. Secure create action in MessageController:

```
     import grails.plugins.springsecurity.Secured
2    [...]
     @Secured(['ROLE_USER'])
4    def create = {
       [...]
6    }
```

Mapping Messages to Users

1. Adding belongsTo to User in Message domain class

```
   class Message {
2    static belongsTo = [User]

4    User author
     [...]
6  }
```

2. On message save action, add to current user object:

```
   class MessageController {
2
     // Inject Acegi Authenticate Service, used in save
4    def springSecurityService

6    [...]

8    def save = {
       def messageInstance = new Message(params)
10
       messageInstance.time = new Date()
12
       // author is set to the corresponding user
```

```
14      def currentUser = User.get(springSecurityService.principal.id)
        messageInstance.author = currentUser
16      [...]
```

3. New action in MessageController to show all users with link to their twidder messages:

```
  def showUsers = {
2   return [users: User.getAll()]
  }
```

4. New view: message/showUsers.gsp

```
1 <%@ page contentType="text/html;charset=UTF-8" %>

3 <ul>
    <g:each var="user" in="${users}">
5     <li><g:link controller="message" action="messagesOfUser" id="${
          user.id}">${user?.encodeAsHTML()}</g:link></li>
    </g:each>
7 </ul>
```

5. Include this in index.gsp:

```
1 [...]
        <ul>
3         <li><g:link controller="message" action="create">Write
              Message</g:link>
          <li><g:link controller="message" action="messages">Read own
              Message</g:link>
5         <li>Read from other Twidder users:
            <g:include controller="message" action="showUsers" />
7       </ul>
  [...]
```

6. Create messagesOfUser action in MessageController:

```
  def messagesOfUser = {
2   def user = User.get(params.id)

4   def userMessages = Message.createCriteria().list(sort:'time',
        order:'desc') {
      eq('author', user)
6   }

8
    def userName = "${user.firstname} ${user.lastname}"
10
    return [userMessages: userMessages,
12    userName: userName]
  }
```

7. Create corresponding view:

```
<%@ page contentType="text/html;charset=UTF-8" %>

<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <title>${userName}</title>
  </head>
  <body>
    <h1>${userName}</h1>
  </body>

  <g:render template="/message/message" var="message" collection="${userMessages}" />
</html>
```