University of Southampton

IBBRE Technology

A technical case study

David Fowler, Gary Wills, Lester Gilbert
dwf@ecs.soton.ac.uk
School of Electronics and Computer Science
University of Southampton

October 11, 2010

IBBRE Technical Case Study

Contents

1 Introduction 4
1.1 Purpose of Document 4
1.2 Scope 4
1.3 Layout of the document 4
2 Introduction to LifeGuide and IBBRE 4
3 Technologies 5
3.1 Software development technologies 5
3.1.1 Grails (Groovy and Rails) 5
3.1.2 Programming IDE(s) 6
3.1.3 MySQL 7
3.1.4 Software version control (Subversion) 9
3.1.5 Grails plugins 10
3.1.6 Web technologies 11
3.1.7 Tools for testing 12
3.2 Additional web-based features 13
3.2.1 Forums 13
3.3 Project monitoring technologies 14
3.3.1 Informal feedback 14
3.3.2 Bug tracking software 14
4 Tutorial section 16
4.1 Common operations with Grails 16
4.1.1 Installation 16
4.1.2 Upgrading Grails 16
4.1.3 Running Grails applications on your local computer 16
4.1.4 Testing applications 16
4.1.5 Creating new domain classes, controllers, etc 17
4.2 Common operations with SVN 17
4.2.1 Installing the Subclipse SVN plugin to Eclipse 18
4.2.2 Importing the IBBRE code 18
4.3 Project deployment 18

4.3.1 Testing the code 19
4.3.2 Creating the WAR file 19
4.3.3 Copying the WAR file 20
4.3.4 Restarting the server (may be optional) 20
4.4 Setting up a discussion forum for an intervention 21
4.5 Modifying the production database schema 22
5 Commonly used Grails features 25
5.1 Access Control 25
5.1.1 A possible alternative – Shiro 25
5.2 Bootstrapping 28
5.3 Grails Testing 29
6 Recommendations 31
6.1 Testing 31
6.2 Code refactoring 31
6.3 Encourage/enforce the use of bug tracking software 31
6.4 Making the source code more widely available 31
6.5 Database migrations 31
7 Glossary 32
8 References and Further Reading 32

1 Introduction

1.1 Purpose of Document

This document forms one of the deliverables for the IBBRE project, namely: “a technical case study that outlines the technologies which have been adapted or integrated and the resources involved” (Wills, 2009)

It also forms part of an “end-to-end development tutorial” as recommended by the SSI report (Crouch, 2010). However, it deals only with the LifeGuide/IBBRE server, not the LifeGuide Authoring Tool. The tutorial is aimed at new developers who may continue working with the IBBRE code to add new features.

1.2 Scope

This document does not cover the Authoring Tool, as it is not strictly part of IBBRE. However, a similar tutorial approach may also be useful for future Authoring Tool developers.

1.3 Layout of the document

In Section 2, we introduce LifeGuide and IBBRE, giving some definitions of terms. In Section 3, we describe the technologies used in the project. Section 0 gives a tutorial explaining common tasks, lessons learned, and possible future directions. The two sections (3 and 0) are only loosely connected, and can be read independently. Throughout the document are standalone “lessons learned” boxes, which contain some hard-won knowledge that would have saved much time and effort. One key conclusion (backed up by experience, if not actual data) is that time spent on testing, documenting, refactoring code and making code more robust against errors will save much more time later in (a) tracking down bugs and (b) gaining sufficient understanding of code that other developers have written, in order to extend it. Such activities should be insisted on, and built into project plans.

2 Introduction to LifeGuide and IBBRE

The IBBRE website (also known as the LifeGuide Community website) allows behavioural scientists to upload, experiment with, and discuss *behavioural interventions.*

Behavioural interventions (BIs) – packages of advice and support for behaviour change – are arguably the most important methodology and technology employed by behavioural scientists for understanding and changing behaviour. Internet-Based BIs (IBBIs) are beginning to play a crucial role in the delivery of BIs, as they can be completed anytime and anywhere. IBBIs can be made available over the web to most of the population for little more than the cost of development. Interactive IBBIs can provide information and advice specifically ‘tailored’ to address the particular situation, concerns, beliefs, and preferences of the individual, by adapting their path through the information space.

The aims of the project were to develop a VRE to enable behavioural scientists creating IBBIs to collaborate in sharing and reviewing components of internet-delivered interventions, and to analyse if the VRE could be flexibly used to support collaborations within and outside the university.

3 Technologies

The technologies used in IBBRE can be classified as (a) those used for developing the server software and website, and (b) those used for project monitoring. The category (a) technologies are used exclusively by developers in ECS, whereas the category (b) technologies are used jointly by the project managers and the developers.

3.1 Software development technologies

Under the category of software development technologies, we include all those used for creating, extending, testing and monitoring the software.

3.1.1 Grails (Groovy and Rails)

The majority of the IBBRE software is written using Grails[1], which combines the Groovy scripting language with the Rails framework. The Groovy language is based on Java, but includes many syntactic shortcuts. These shortcuts can simplify coding a lot, but can also contain pitfalls for the unwary. For example, type checking is much less strict, and errors that would be spotted at compile time with Java may only be revealed at run time with Groovy.

The Rails framework is a way of organising projects using the Model View Controller pattern. Files are organised as models (describing data), controllers (programs that specify how the data is manipulated), and views (how the users can see and interact with the data).

For an example, interventions in IBBRE are modelled using the *domain class* `Intervention`, which is stored in the Groovy file `Intervention.groovy`. The *controller* is `InterventionController`, (in the file `InterventionController.groovy`). There are several *views*, one for each display or interaction, for example, the view for editing an intervention is in the file `edit.gsp` which is stored in the folder `views/Intervention`.

For long term storage of data, Grails uses GORM (Grails Object Relation Mapping). GORM maps domain classes to tables in a database, and handles the storage and retrieval of instance data. It aims to be transparent, so that the developer can think in terms of classes and objects without knowing about how the tables are actually arranged. (However, this ideal scenario is not always realisable, and the developer sometimes needs to delve into the inner workings of the database, and to have some idea of how the tables relate to the logical data objects.)

There are many resources available on the web dealing with Grails, in particular the reference documentation[2]. The book "Grails in Action" (Smith & Ledbrook, 2009) is also highly recommended.

[1] Grails: http://www.grails.org/
[2] The Grails Framework - Reference Documentation http://grails.org/doc/latest/

Lesson learned: defensive programming

As it is not advisable to restart the LifeGuide server outside the weekly "at risk" period (see Section 4.3 for more on this), it is very important to minimise the frequency of errors. Also, by taking a few minutes to add safeguards and validation to the code will save much more time later in tracking down bugs.

As a case in point, the following lines of code were used for sending SMS (text) messages to users:

```
def acc = ApplicationHolder.application.getClassForName
("InterventionSmsAccount").findByIntervention(message.intervention)

MessageServiceProvider provider = acc?.provider?.newProviderInstance();
boolean sent = provider.sendMessage(message)
return sent
```

The problem here is that if `provider` is null, the `provider.sendMessage(message)` invocation will (and did) throw an exception. By simply adding a check, this could have been avoided:

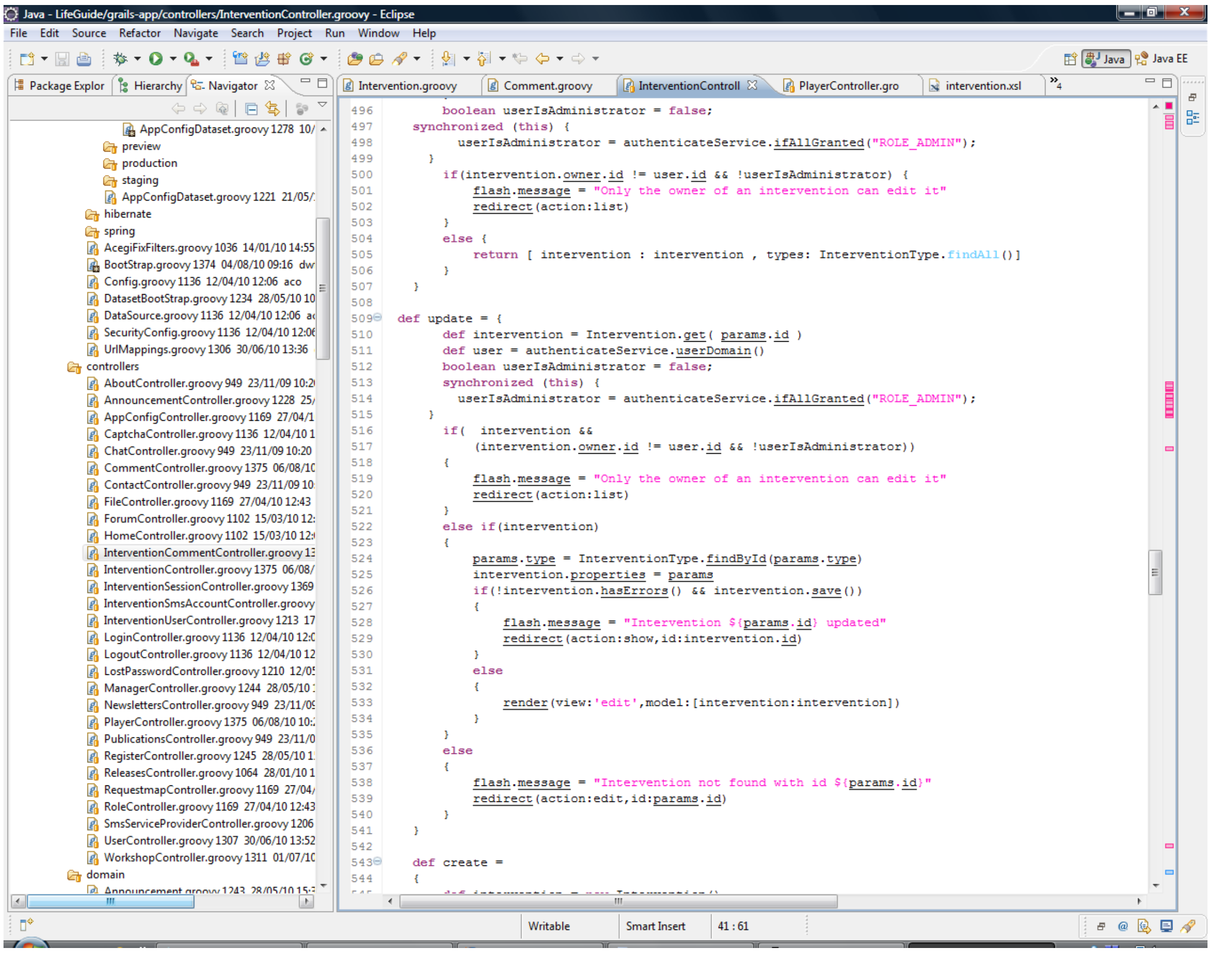
```
def acc = ApplicationHolder.application.getClassForName
("InterventionSmsAccount").findByIntervention(message.intervention)

boolean sent = false;

if (provider != null) {
       sent = provider.sendMessage (message)
       }
return sent
```

Groovy's safe dereference operator ?. is also very useful, and can be seen in the code above (for example, if `acc` is null then `acc.provider` would cause an exception, but `acc?.provider` is just null).

3.1.2 Programming IDE(s)

Most modern programming is carried out with the help of an Integrated Development Environment (IDE), which gathers together many tools (e.g. editors, compilers, debuggers, access to version control software, etc).

A typical IDE is Eclipse (http://www.eclipse.org/), which was probably the most useful for IBBRE, as the Authoring Tool builds on the Eclipse framework (to minimise disruption for developers that work on the Authoring Tool as well). Figure 1 shows Eclipse being used to work on IBBRE – other features that can be accessed from Eclipse will be discussed in later sections.



```
        boolean userIsAdministrator = false;
    synchronized (this) {
          userIsAdministrator = authenticateService.ifAllGranted("ROLE_ADMIN");
      }
        if(intervention.owner.id != user.id && !userIsAdministrator) {
            flash.message = "Only the owner of an intervention can edit it"
            redirect(action:list)
        }
        else {
            return [ intervention : intervention , types: InterventionType.findAll()]
        }
    }

  def update = {
        def intervention = Intervention.get( params.id )
        def user = authenticateService.userDomain()
        boolean userIsAdministrator = false;
        synchronized (this) {
          userIsAdministrator = authenticateService.ifAllGranted("ROLE_ADMIN");
      }
        if(  intervention &&
            (intervention.owner.id != user.id && !userIsAdministrator))
        {
            flash.message = "Only the owner of an intervention can edit it"
            redirect(action:list)
        }
        else if(intervention)
        {
            params.type = InterventionType.findById(params.type)
            intervention.properties = params
            if(!intervention.hasErrors() && intervention.save())
            {
                flash.message = "Intervention ${params.id} updated"
                redirect(action:show,id:intervention.id)
            }
            else
            {
                render(view:'edit',model:[intervention:intervention])
            }
        }
        else
        {
            flash.message = "Intervention not found with id ${params.id}"
            redirect(action:edit,id:params.id)
        }
    }

    def create =
    {
```

Figure 1: The Eclipse IDE, showing part of the Grails project structure (left) and a Grails file (right).

3.1.3 MySQL

In order to store all the data for users, interventions, comments, etc, a relational database was used – in IBBRE we used MySQL[3]. This was installed as a database server on an ECS virtual machine. The database can be manipulated via SQL commands via a command line, or by a graphical tool such as MySQL Query Browser or MySQL Workbench[4]. The example screenshots (Listing 1 to Listing 3) use the command line, but the graphical tools may be preferable.

[3] http://www.mysql.com/
[4] http://dev.mysql.com/downloads/workbench/

```
[dwf@octopussy ~]$ mysql --user=root --password=******** IBBRELifeGuideProd
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 84348
Server version: 5.0.45 Source distribution

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> SHOW TABLES;
+-------------------------------+
| Tables_in_IBBRELifeGuideProd |
+-------------------------------+
| announcement                  |
| app_config                    |
| arbitrary_file                |
| arbitrary_file_type           |
| arbitrary_file_types          |
| category                      |
| chat                          |
| comment                       |
| forum                         |
| intervention                  |
| intervention_email_template   |
| intervention_folder           |
| intervention_folder_user      |
| intervention_overview         |
| intervention_pageids          |
| intervention_report           |
| intervention_session          |
| intervention_sms_account      |
| intervention_statistic        |
| intervention_type             |
| intervention_user             |
| invitation                    |
| message                       |
| newsletter                    |
| post                          |
| report                        |
| requestmap                    |
| role                          |
| role_people                   |
| role_user                     |
| sms_service_provider          |
| topic                         |
| user                          |
| user_announcement             |
| user_followed_interventions   |
| user_intervention             |
| user_shared_folders           |
| user_shared_interventions     |
| workshop                      |
+-------------------------------+
39 rows in set (0.00 sec)

mysql>
```

Listing 1: Listing the tables in the IBBRE database.

```
mysql> DESCRIBE intervention;
+-----------------------+--------------+------+-----+---------+----------------+
| Field                 | Type         | Null | Key | Default | Extra          |
+-----------------------+--------------+------+-----+---------+----------------+
| id                    | bigint(20)   | NO   | PRI | NULL    | auto_increment |
| version               | bigint(20)   | NO   |     |         |                |
| allow_public_debug    | bit(1)       | NO   |     |         |                |
| deployed_date         | datetime     | NO   |     |         |                |
| description           | varchar(255) | YES  |     | NULL    |                |
| directory             | varchar(255) | NO   |     |         |                |
| expiry_date           | datetime     | YES  |     | NULL    |                |
| exported_fields       | longtext     | YES  |     | NULL    |                |
| filename              | varchar(255) | NO   |     |         |                |
| long_name             | varchar(255) | NO   |     |         |                |
| name                  | varchar(255) | NO   | UNI |         |                |
| owner_id              | bigint(20)   | NO   | MUL |         |                |
| type_id               | bigint(20)   | NO   | MUL |         |                |
| email_template_id     | bigint(20)   | YES  | MUL | NULL    |                |
| containing_folder_id  | bigint(20)   | YES  | MUL | NULL    |                |
+-----------------------+--------------+------+-----+---------+----------------+
15 rows in set (0.00 sec)

mysql>
```

Listing 2: A description of the Intervention table

```
mysql> SELECT id, long_name, deployed_date FROM intervention ORDER BY id LIMIT 10;
+----+-------------------------+---------------------+
| id | long_name               | deployed_date       |
+----+-------------------------+---------------------+
| 10 | Randomisation Example 1 | 2009-11-26 17:21:00 |
| 13 | Randomisation Example 2 | 2009-12-02 15:22:00 |
| 14 | Randomisation Example 3 | 2009-12-03 11:52:00 |
| 20 | email test              | 2009-12-14 13:12:00 |
| 21 | email test 2            | 2009-12-14 13:32:00 |
| 28 | Container Demo          | 2010-01-14 17:15:00 |
| 31 | Calculation Example     | 2010-01-18 10:33:00 |
| 38 | email test1             | 2010-01-19 11:25:00 |
| 40 | test email              | 2010-01-19 11:51:00 |
| 55 | all together 3          | 2010-01-28 10:24:00 |
+----+-------------------------+---------------------+
10 rows in set (0.00 sec)

mysql>
```

Listing 3: An extract from the Intervention table.

3.1.4 Software version control (Subversion)

In order to allow several developers to work on the project simultaneously, it is typical to use a software version control system. We used Subversion (http://subversion.apache.org/), which was already installed as an ECS service.

In order to commit code to the repository, and to check out existing code, either a separate SVN client can be used (for example Tortoise SVN[5]), or a plugin to Eclipse (such as Subclipse).

An issue that was raised in the SSI report is the possibility of replicating the codebase on SourceForge[6]. In theory, it should be possible to maintain the ECS Forge version, with a mirrored

[5] TortoiseSVN: http://tortoisesvn.tigris.org/
[6] SourceForge: http://sourceforge.net/

copy on SourceForge (so that any modifications to the ECS version are automatically reflected on SourceForge). This was tried (using the svnsync command), but proved impossible due to the setup of ECS Forge. Eventually we resorted to making the code available via ftp, but this was not ideal. In future, it may be best to move all the code once and for all to SourceForge.

3.1.5 Grails plugins

Much useful functionality is provided by plugins to Grails. The main ones used in IBBRE are:

- Spring Security (formerly acegi) – provides mechanisms for security, logging in, finding the current user, etc. See Section 5.1 for more details on access control.
- Quartz – scheduling tasks (e.g. sending emails). In LifeGuide, reminder emails are sent to intervention users at various times in the future. Quartz allows you to specify a job that executes at set intervals. The class MessageSenderJob (see Listing 4) runs every five minutes, and sends any messages that have become due since the last execution.
- Geb and Spock are related plugins used for functional testing (see Section 5.3 for more details).

To see which plugins are available, and which are currently installed, use the command:

```
grails list-plugins
```

To install a plugin, use:

```
grails install-plugin <plugin name>
```

Finally, to uninstall a plugin:

```
grails uninstall-plugin <plugin name>
```

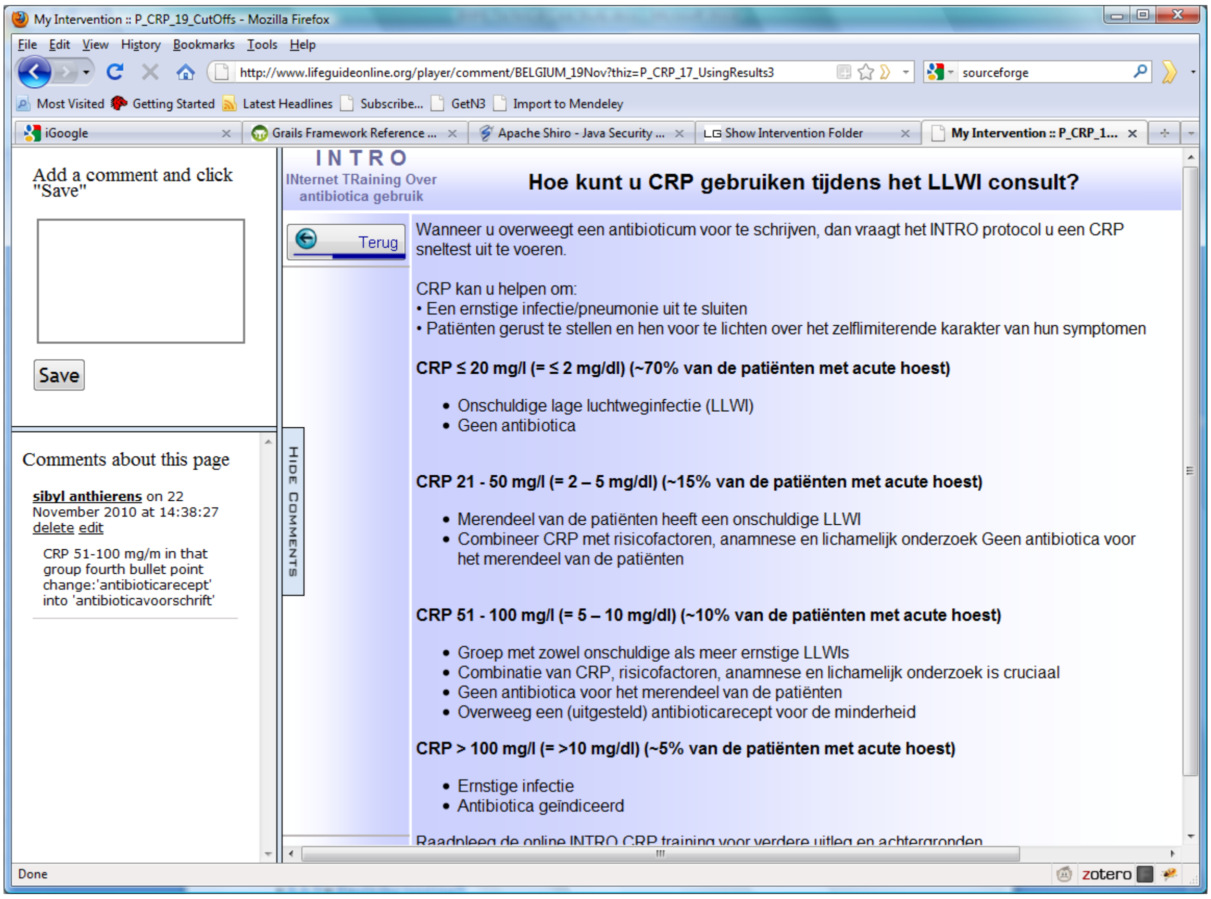
```
import uk.ac.soton.ecs.lifeguide.messaging.MessageCenter

class MessageSenderJob {
        def timeout = 1000L * 60L * 5L // execute job once in 5 mins

        def sessionFactory

        def execute() { //Send messages which haven't been sent, and are due
        def messages = Message.findAllByDateToSendLessThanAndSentDate(
                                                            new Date(), null)

        messages.each { message ->
                if (MessageCenter.sendMessage(message))
                        message.sentDate = new Date()
                        sessionFactory.currentSession.flush()

        }

        //clean up message queue by removing sent messages (over 2 weeks old)
        def messagesToDelete =
        Message.findAllBySentDateIsNotNullAndSentDateLessThan(new Date()-14)

        messagesToDelete.each { message ->      message.delete() }
        }
}
```

Listing 4: The Quartz task that sends LifeGuide messages.

3.1.6 Web technologies

For laying out web pages, and for special interactive effects, it is often necessary to make use of web technologies such as CSS[7], Javascript libraries, and Ajax. For example, the intervention commenting system (Figure 2) makes use of Javascript to render the resizable comment panes and Ajax to post the comments. More information, and tutorials, can be found on the web, and in Chapter 6 of "Grails in Action" (Smith & Ledbrook, 2009).

A very useful tool for testing and debugging web pages is Firebug[8] (see Figure 3), which allows you to compare the HTML and CSS code with the actual web page layout in your browser.


My Intervention :: P_CRP_19_CutOffs - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://www.lifeguideonline.org/player/comment/BELGIUM_19Nov?thiz=P_CRP_17_UsingResults3
sourceforge
Most Visited Getting Started Latest Headlines Subscribe... GetN3 Import to Mendeley
iGoogle
Grails Framework Reference ...
Apache Shiro - Java Security ...
Show Intervention Folder
My Intervention :: P_CRP_1...
INTRO
INternet TRaining Over antibiotica gebruik
Terug
HIDE COMMENTS
Save
Done
zotero

Add a comment and click "Save"

Comments about this page

sibyl anthierens on 22 November 2010 at 14:38:27
delete edit

CRP 51-100 mg/m in that group fourth bullet point change:'antibioticarecept' into 'antibioticavoorschrift'

Hoe kunt u CRP gebruiken tijdens het LLWI consult?

Wanneer u overweegt een antibioticum voor te schrijven, dan vraagt het INTRO protocol u een CRP sneltest uit te voeren.

CRP kan u helpen om:
• Een ernstige infectie/pneumonie uit te sluiten
• Patiënten gerust te stellen en hen voor te lichten over het zelflimiterende karakter van hun symptomen

CRP ≤ 20 mg/l (= ≤ 2 mg/dl) (~70% van de patiënten met acute hoest)

- Onschuldige lage luchtweginfectie (LLWI)
- Geen antibiotica

CRP 21 - 50 mg/l (= 2 – 5 mg/dl) (~15% van de patiënten met acute hoest)

- Merendeel van de patiënten heeft een onschuldige LLWI
- Combineer CRP met risicofactoren, anamnese en lichamelijk onderzoek Geen antibiotica voor het merendeel van de patiënten

CRP 51 - 100 mg/l (= 5 – 10 mg/dl) (~10% van de patiënten met acute hoest)

- Groep met zowel onschuldige als meer ernstige LLWIs
- Combinatie van CRP, risicofactoren, anamnese en lichamelijk onderzoek is cruciaal
- Geen antibiotica voor het merendeel van de patiënten
- Overweeg een (uitgesteld) antibioticarecept voor de minderheid

CRP > 100 mg/l (= >10 mg/dl) (~5% van de patiënten met acute hoest)

- Ernstige infectie
- Antibiotica geïndiceerd

Raadpleeg de online INTRO CRP training voor verdere uitleg en achtergronden

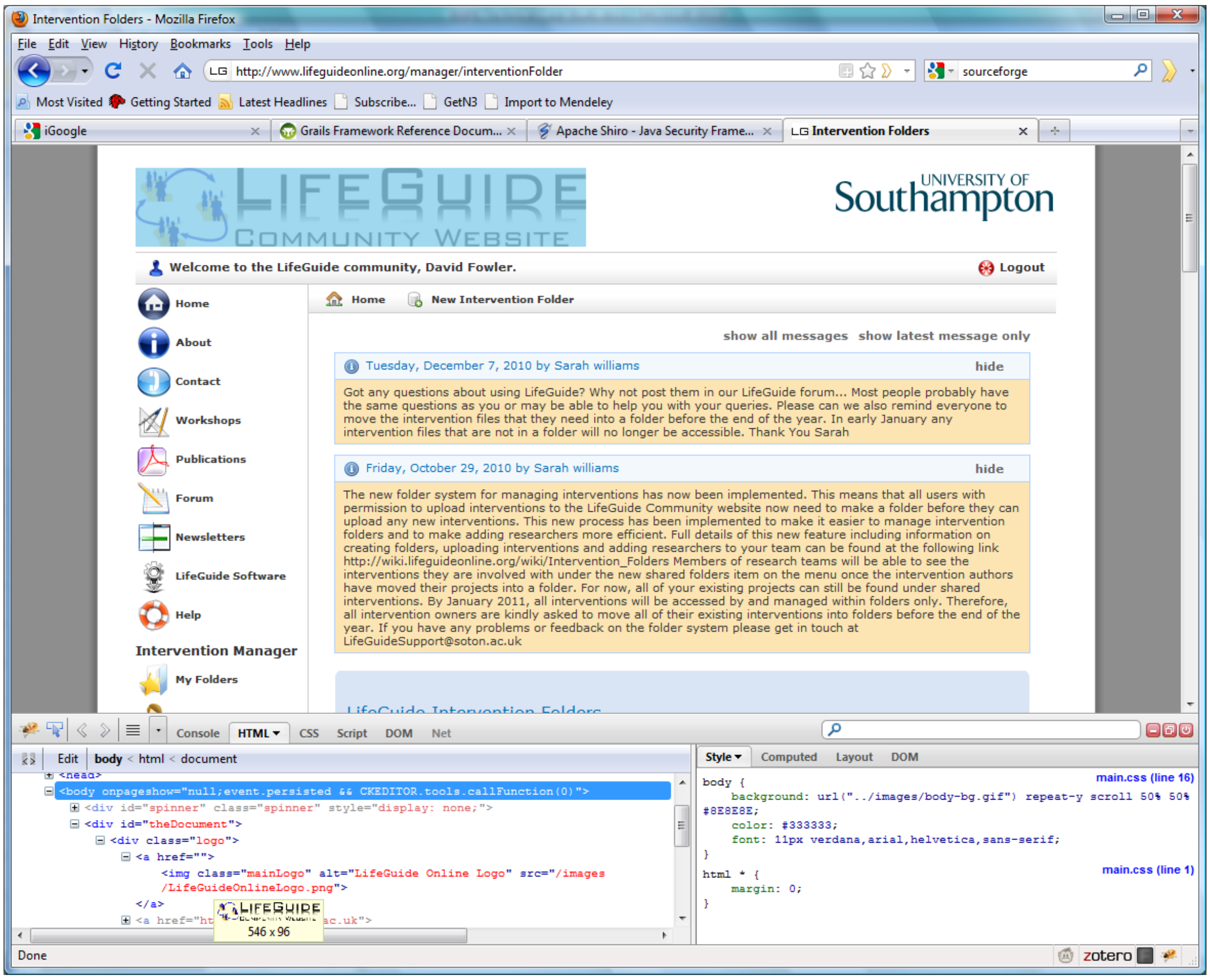
Figure 2: The intervention commenting system.

[7] Cascading Style Sheets: http://www.w3.org/Style/CSS/
[8] Firebug: http://getfirebug.com/

Lesson Learned: Test web layouts in different browsers

The majority of LifeGuide users tend to use a version of Internet Explorer for browsing (however, this is a subjective opinion and may be skewed, as we tend to get more problems with IE, and therefore we hear more from these users). It is very important to make sure that any web pages look good (and that, e.g., Javascript functionality works) in IE as well as other popular browsers. It is easy to slip into the habit of using your favourite browser, but not checking others.

You should also make sure that you do not use non-standard HTML tags that work in one browser but not others.



Tuesday, December 7, 2010 by Sarah williams

Got any questions about using LifeGuide? Why not post them in our LifeGuide forum... Most people probably have the same questions as you or may be able to help you with your queries. Please can we also remind everyone to move the intervention files that they need into a folder before the end of the year. In early January any intervention files that are not in a folder will no longer be accessible. Thank You Sarah

Friday, October 29, 2010 by Sarah williams

The new folder system for managing interventions has now been implemented. This means that all users with permission to upload interventions to the LifeGuide Community website now need to make a folder before they can upload any new interventions. This new process has been implemented to make it easier to manage intervention folders and to make adding researchers more efficient. Full details of this new feature including information on creating folders, uploading interventions and adding researchers to your team can be found at the following link http://wiki.lifeguideonline.org/wiki/Intervention_Folders Members of research teams will be able to see the interventions they are involved with under the new shared folders item on the menu once the intervention authors have moved their projects into a folder. For now, all of your existing projects can still be found under shared interventions. By January 2011, all interventions will be accessed by and managed within folders only. Therefore, all intervention owners are kindly asked to move all of their existing interventions into folders before the end of the year. If you have any problems or feedback on the folder system please get in touch at LifeGuideSupport@soton.ac.uk

```
<head>
<body onpageshow="null;event.persisted && CKEDITOR.tools.callFunction(0)">
  <div id="spinner" class="spinner" style="display: none;">
  <div id="theDocument">
    <div class="logo">
      <a href="">
          <img class="mainLogo" alt="LifeGuide Online Logo" src="/images
          /LifeGuideOnlineLogo.png">
        </a>
      <a href="ht                ac.uk">
```

```
body {
    background: url("../images/body-bg.gif") repeat-y scroll 50% 50%
#8E8E8E;
    color: #333333;
    font: 11px verdana,arial,helvetica,sans-serif;
}
html * {
    margin: 0;
}
```

Figure 3: Using Firebug to examine web page contents.

3.1.7 Tools for testing

In addition to the functional testing provided by the Geb and Spock plugins (see Sections 3.1.5 and 5.3), Grails provides facilities for unit and integration testing, based on the JUnit framework. Some use was made of these in IBBRE, but not enough. Grails will create test stubs for each domain class automatically, and the developer should add code here to check that, for example, validation is performed correctly (see Listing 5 for an example of a unit test for the User class).

```
import grails.test.*

class UserTests extends GrailsUnitTestCase {

	void testConstraints() {
		def existingUser = new User(username: "fred",
			userRealName: "Fred Bloggs",
			passwd: "abcd123",
			enabled: true,
			receiveEmailAnnouncements: false,
			email: "fred@nowhere.com")

		mockForConstraintsTests(User, [existingUser])

		def testUser = new User()
		assertFalse testUser.validate()
		assertEquals "nullable", testUser.errors["username"]
		assertEquals "nullable", testUser.errors["userRealName"]
		assertEquals "nullable", testUser.errors["passwd"]
		assertEquals "nullable", testUser.errors["email"]

		testUser = new User(email: "whatever")
		assertFalse testUser.validate()
		assertEquals "email", testUser.errors["email"]

		testUser = new User(username: "fred")
		assertFalse testUser.validate()
		assertEquals "unique", testUser.errors["username"]

		testUser = new User(username: "alice",
				userRealName: "Alice Bloggs",
				passwd: "abcd123",
				enabled: true,
				receiveEmailAnnouncements: false,
				email: "alice@nowhere.com")
		assertTrue testUser.validate()
	}
}
```

Listing 5: A Grails unit test for the User class.

3.2 Additional web-based features

3.2.1 Forums

Forums can be set up for particular interventions, or possibly groups of related interventions (there is no strictly enforced connection, but typically a forum is intended for discussion by participants and owners of an intervention).

We used phpBB[9] for setting up forums. Rather than installing a new copy of phpBB for each forum, it is possible to use a single installation of phpBB to host multiple forums, which reduces setup time to just a few minutes (see Section 4.4 for details).[10]

[9] phpBB Free and Open Source Forum Software: http://www.phpbb.com/

[10] The original online discussion describing the solution that we adapted can be found at http://www.phpbb.com/community/viewtopic.php?t=272407 [accessed 11 October 2010]

3.3 Project monitoring technologies

There are several ways in which the project progress is monitored (new features requested, bugs reported). These include:

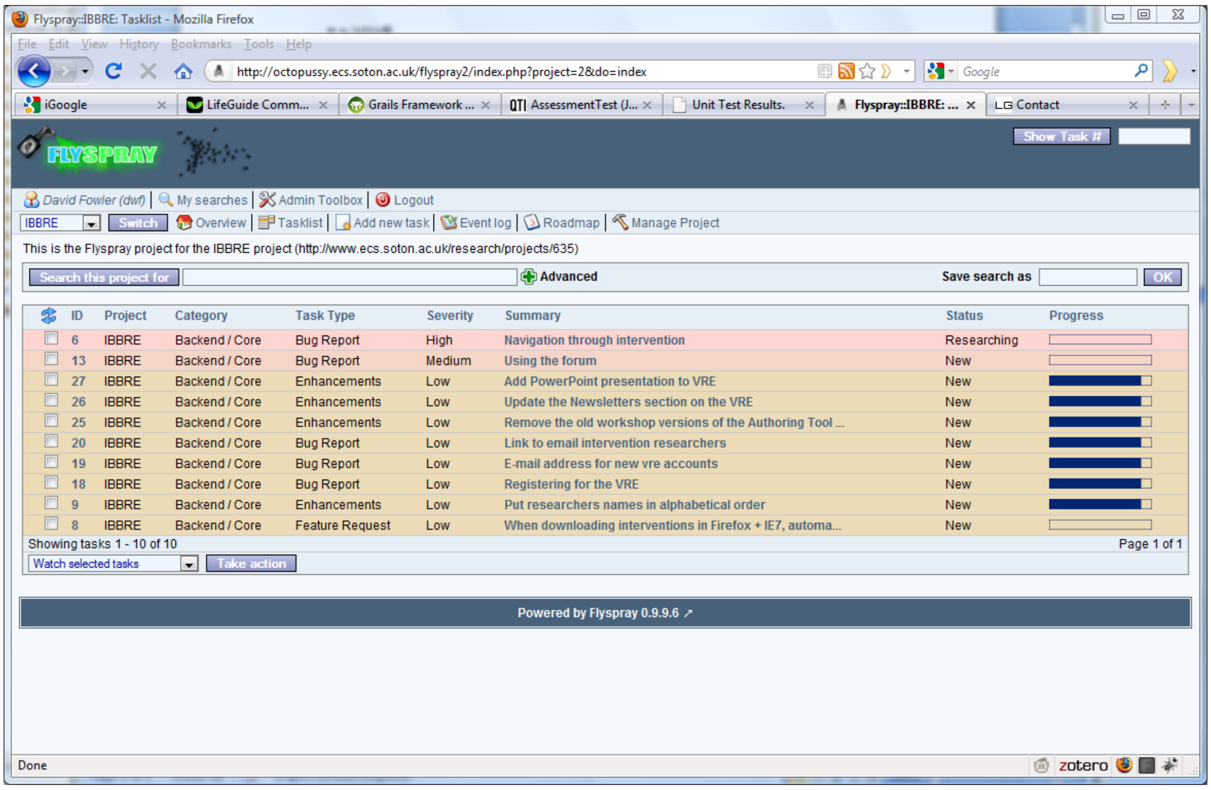
- informal feedback (emails, word of mouth, forums)
- prioritisation list
- GANTT chart
- meeting agendas and minutes, with actions
- server logs, including automated emails to admin
- bug tracking software

3.3.1 Informal feedback

Many requests come via email, word of mouth, or a forum. For reporting bugs, this is typically done by email. However, this is not always ideal, as it is difficult to see quickly which tasks are outstanding, and what progress has been made on each. It is often difficult for the originator of the bug report to know which developer to send it to, so he or she tends to send it to all developers.

3.3.2 Bug tracking software

For bug tracking and for feature requests, we experimented with FlySpray (http://flyspray.org/), a system that allows tasks to be added and monitored for a project. The software is open-source, written in PHP, and was installed on a local ECS server.



Flyspray::IBBRE: Tasklist - Mozilla Firefox

http://octopussy.ecs.soton.ac.uk/flyspray2/index.php?project=2&do=index

This is the Flyspray project for the IBBRE project (http://www.ecs.soton.ac.uk/research/projects/635)

ID	Project	Category	Task Type	Severity	Summary	Status	Progress
6	IBBRE	Backend / Core	Bug Report	High	Navigation through intervention	Researching	
13	IBBRE	Backend / Core	Bug Report	Medium	Using the forum	New	
27	IBBRE	Backend / Core	Enhancements	Low	Add PowerPoint presentation to VRE	New	
26	IBBRE	Backend / Core	Enhancements	Low	Update the Newsletters section on the VRE	New	
25	IBBRE	Backend / Core	Enhancements	Low	Remove the old workshop versions of the Authoring Tool ...	New	
20	IBBRE	Backend / Core	Bug Report	Low	Link to email intervention researchers	New	
19	IBBRE	Backend / Core	Bug Report	Low	E-mail address for new vre accounts	New	
18	IBBRE	Backend / Core	Bug Report	Low	Registering for the VRE	New	
9	IBBRE	Backend / Core	Enhancements	Low	Put researchers names in alphabetical order	New	
8	IBBRE	Backend / Core	Feature Request	Low	When downloading interventions in Firefox + IE7, automa...	New	

Showing tasks 1 - 10 of 10 Page 1 of 1

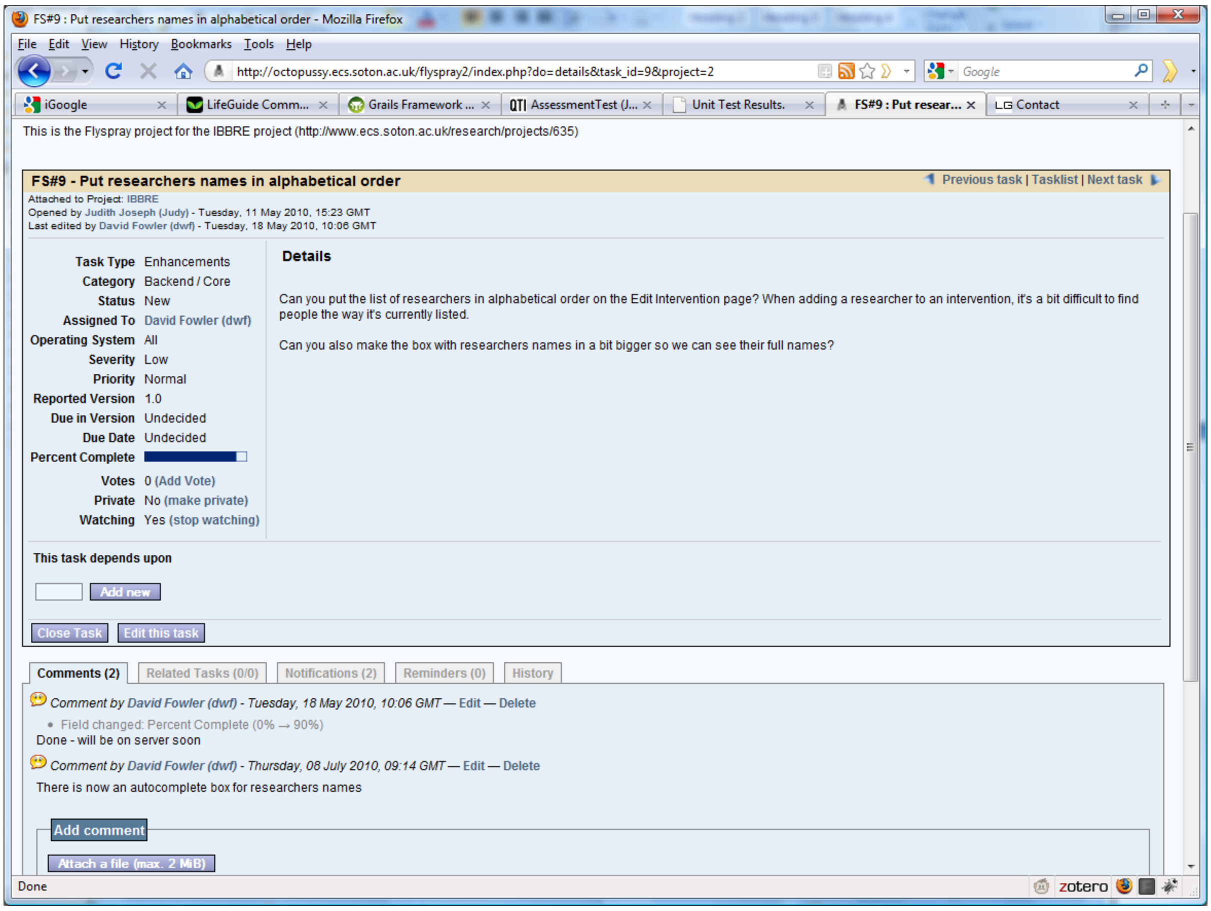
Powered by Flyspray 0.9.9.6

Figure 4: The IBBRE task list in FlySpray

FlySpray was used fairly intensively after we adopted it, but the usage tailed off towards the end of the project. Although it was very useful in alerting developers to new problems and tasks (an email alert is sent to the relevant developer), after a while new problems and tasks were no longer being added, and the informal approach of directly emailing the developers was reverted to.

The main advantage of using FlySpray is that it allows tabular summaries of tasks, sorted by priority, and with a clear depiction of how much progress has been made on each task. This is much easier than searching through past emails to find outstanding tasks, or searching through notes made in meetings.

The main drawback is that it is more effort to create a new task in FlySpray than to send an email directly or to report a problem verbally. It seems that the initial overhead of going to the FlySpray website and creating a new task is too high, or perceived to be too high, even though the end benefits are quite good. Some way of reducing this initial effort, or of making the end benefit more visible, may be needed. As well as, or instead of this, simply insisting that end users report problems or request new features via FlySpray before they are dealt with may be a solution.


FS#9 : Put researchers names in alphabetical order - Mozilla Firefox

http://octopussy.ecs.soton.ac.uk/flyspray2/index.php?do=details&task_id=9&project=2

This is the Flyspray project for the IBBRE project (http://www.ecs.soton.ac.uk/research/projects/635)

FS#9 - Put researchers names in alphabetical order

Attached to Project: IBBRE
Opened by Judith Joseph (Judy) - Tuesday, 11 May 2010, 15:23 GMT
Last edited by David Fowler (dwf) - Tuesday, 18 May 2010, 10:06 GMT

Task Type	Enhancements
Category	Backend / Core
Status	New
Assigned To	David Fowler (dwf)
Operating System	All
Severity	Low
Priority	Normal
Reported Version	1.0
Due in Version	Undecided
Due Date	Undecided
Percent Complete	
Votes	0 (Add Vote)
Private	No (make private)
Watching	Yes (stop watching)

Details

Can you put the list of researchers in alphabetical order on the Edit Intervention page? When adding a researcher to an intervention, it's a bit difficult to find people the way it's currently listed.

Can you also make the box with researchers names in a bit bigger so we can see their full names?

This task depends upon

Comment by David Fowler (dwf) - Tuesday, 18 May 2010, 10:06 GMT — Edit — Delete

- Field changed: Percent Complete (0% → 90%)

Done - will be on server soon

Comment by David Fowler (dwf) - Thursday, 08 July 2010, 09:14 GMT — Edit — Delete

There is now an autocomplete box for researchers names

Done

Figure 5: Details of a single task in FlySpray

4 Tutorial section

In this section, we describe how to perform common tasks with the IBBRE server, and give some guidance on future development and testing of the IBBRE software. This section will be of interest to future developers of IBBRE, but it may also give some ideas for developers of similar systems.

4.1 Common operations with Grails

In this section we list just the most common Grails operations. For details of other operations, consult the Grails website, or type:

```
grails help
```

at the command line for a quick list (after installing Grails as described in section 4.1.1 below).

4.1.1 Installation

To install Grails on your machine, follow these steps:

- Download the latest version of Grails from http://www.grails.org/Download.
- Set the `GRAILS_HOME` environment variable to refer to your downloaded directory.
- Add `$GRAILS_HOME/bin` to your system path, so that you can run Grails scripts from your command line console.

4.1.2 Upgrading Grails

To upgrade your version of Grails to the latest version, use the command:

```
grails upgrade
```

4.1.3 Running Grails applications on your local computer

To run the Grails application in development mode (i.e. on your local machine, with a temporary database), first change directory to that of the application, for example:

```
cd workspace/LifeGuide
```

and then:

```
grails run-app
```

You can then point your web browser to http://localhost:8080/LifeGuide/ and use the application.

4.1.4 Testing applications

To run all unit, integration, and functional tests in sequence, use the command:

```
grails test-app
```

For individual types of test, you can use (for example):

```
grails test-app –functional
```

(In fact, in the current setup, the environment for functional tests seems to conflict with that of other tests, so it is advisable to run functional tests separately – otherwise the functional tests will fail, even when they shouldn't). The results from testing are stored in html files that can be viewed in a browser (see Figure 6).

4.1.5 Creating new domain classes, controllers, etc

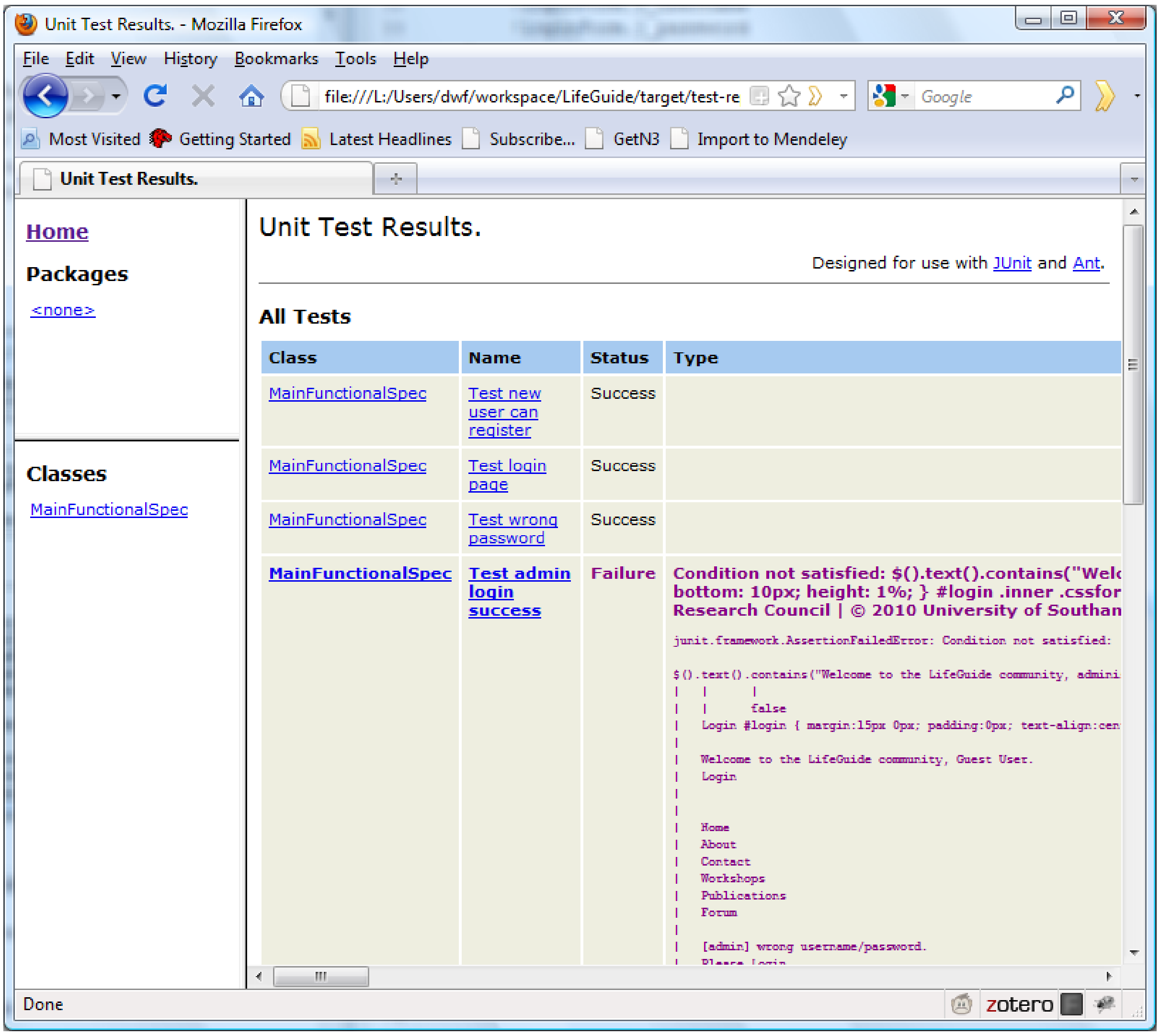
Although it is possible to write classes and controllers from scratch, using a text editor (or more likely, an IDE such as Eclipse), it is usually easier to use the commands:

```
grails create-domain-class <class name>

grails create-controller <class name>
```

These produce the skeleton code that can be filled in later. To create views, you can use:

```
grails generate-views <class name>
```

The views will generally require much tailoring to suit your needs. For more details, please see the web resources or the Grails in Action book.


Unit Test Results. - Mozilla Firefox
File Edit View History Bookmarks Tools Help
file:///L:/Users/dwf/workspace/LifeGuide/target/test-re
Google
Most Visited Getting Started Latest Headlines Subscribe... GetN3 Import to Mendeley
Unit Test Results.
Home
Packages
<none>
Classes
MainFunctionalSpec
Done
zotero

Unit Test Results.

Designed for use with JUnit and Ant.

All Tests

Class	Name	Status	Type
MainFunctionalSpec	Test new user can register	Success	
MainFunctionalSpec	Test login page	Success	
MainFunctionalSpec	Test wrong password	Success	
MainFunctionalSpec	**Test admin login success**	**Failure**	**Condition not satisfied: $().text().contains("Welc bottom: 10px; height: 1%; } #login .inner .cssfor Research Council | © 2010 University of Southan**

```
junit.framework.AssertionFailedError: Condition not satisfied:

$().text().contains("Welcome to the LifeGuide community, admini
|   |       |
|   |       false
|   Login #login { margin:15px 0px; padding:0px; text-align:cen
|
|   Welcome to the LifeGuide community, Guest User.
|   Login
|
|
|   Home
|   About
|   Contact
|   Workshops
|   Publications
|   Forum
|
|   [admin] wrong username/password.
|   Please Login
```

Figure 6: Automated test results in Grails.

4.2 Common operations with SVN

For more background information on SVN, see the online book "Version Control with Subversion" (Collins-Sussman, Fitzpatrick, & Pilato, 2010).

To check out (or import) the IBBRE code from the SVN repository into Eclipse, you need to have the Subclipse plugin installed in Eclipse.

4.2.1 Installing the Subclipse SVN plugin to Eclipse

In order to install SVN, follow these steps:

- Inside your eclipse IDE, Go to: Help ->Software Updates;
- Change to the Available Software tab;
- Click on Add Site;
- Paste the update site URL (http://subclipse.tigris.org/update_1.0.x) and click OK;
- Restart Eclipse.

4.2.2 Importing the IBBRE code

In order to reuse IBBRE source code for your own development, follow these steps:

- From Eclipse's File menu, choose Import to display the import manager. Choose Checkout Projects from SVN, then click Next;
- On the Select/Create Location panel, we need to create a new location (since we don't have any configured yet), so click Next to continue. If the Next button is disabled, switch to the Use existing repository location option, then back to Create a new repository location to enable the Next button;
- In the next section add the repository URL (in our setup this is svn+ssh://yourname@svn.forge.ecs.soton.ac.uk/projects/Lifeguide) to the URL field, and then click Next. After a moment, Eclipse prompts you for your password. Check the Save Password box, and click OK;
- Expand branches/ibbre-branch/LifeGuide and click OK.

To update your code (i.e. to import any changes that other users have made to the code), right click on the file(s) or folder(s) that you wish to update, and select "Team/Update to HEAD".

To commit changes, right click the file(s) or folder(s) and select "Team/Commit".

To compare your version of a file with the repository, select "Compare With/Latest From Repository". This gives a display of both versions of the file, with the differences highlighted.

4.3 Project deployment

Warning – the commands in the following section will briefly shutdown the server, meaning that any logged in users will be unable to use the system. We use a specified time each week (Wednesdays from 9-10am), when the server is "at risk", so that impact on users is minimised.

As I cannot use the main server for experimentation, the example screenshots will generally be for the "beta" server, with explanations as to how to modify the commands for the main server.

If new code has been committed in SVN, the server has to be updated in order to use the code. To do this:

- the new code should be tested, to reduce the chances of errors (see section 4.1.4);
- a WAR file has to be generated from the Grails project;
- the WAR file must be copied to the correct location on the server;
- the server container can be restarted (this is not always necessary, as just copying the WAR file is usually sufficient)

4.3.1 Testing the code

The testing is done by the command

```
grails test-app
```

or `grails test-app -unit` (or `-integration` or `-functional`) for specific types of tests (the current setup for functional tests is incompatible with the setup for unit and integration tests, so it is best to do the functional tests separately to avoid misleading results).

4.3.2 Creating the WAR file

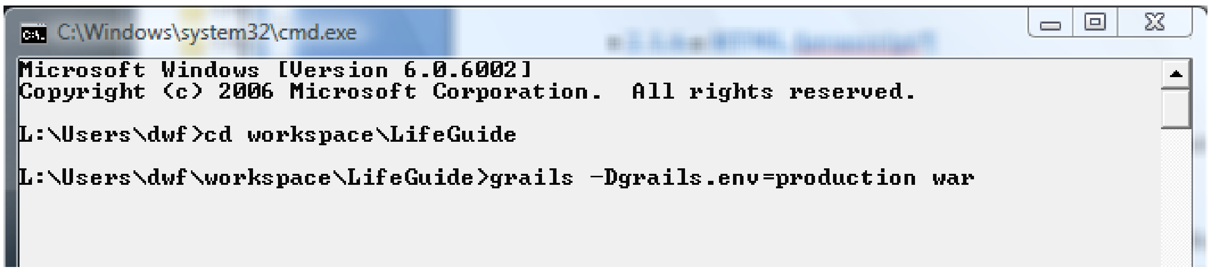
Creating the WAR file is done on the developer's machine (Figure 7 and Figure 8), using the command:

```
grails -Dgrails.env=production war
```

For other environments (usually the beta server, where the staging environment is used), you can change "production" to the name of the environment, for example:

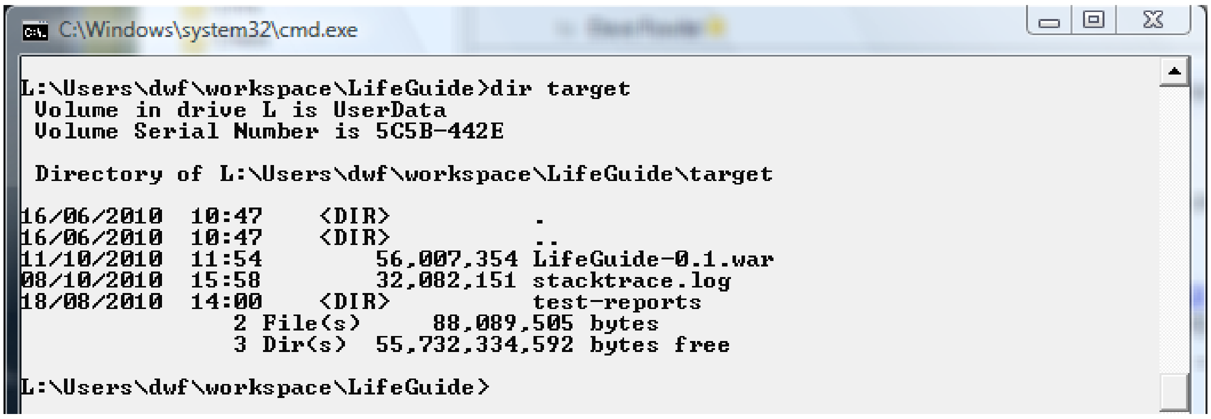
```
grails -Dgrails.env=staging war
```

You will almost certainly not need any environments other than production and staging for building WAR files (as the development and test versions are run without building a WAR file).



```
Microsoft Windows [Version 6.0.6002]
Copyright (c) 2006 Microsoft Corporation.  All rights reserved.

L:\Users\dwf>cd workspace\LifeGuide

L:\Users\dwf\workspace\LifeGuide>grails -Dgrails.env=production war
```

Figure 7: Creating the WAR file (you will see many diagnostic messages after entering the command)



```
L:\Users\dwf\workspace\LifeGuide>dir target
 Volume in drive L is UserData
 Volume Serial Number is 5C5B-442E

 Directory of L:\Users\dwf\workspace\LifeGuide\target

16/06/2010  10:47    <DIR>          .
16/06/2010  10:47    <DIR>          ..
11/10/2010  11:54        56,007,354 LifeGuide-0.1.war
08/10/2010  15:58        32,082,151 stacktrace.log
18/08/2010  14:00    <DIR>          test-reports
               2 File(s)     88,089,505 bytes
               3 Dir(s)  55,732,334,592 bytes free

L:\Users\dwf\workspace\LifeGuide>
```

Figure 8: Checking that the WAR file was created.

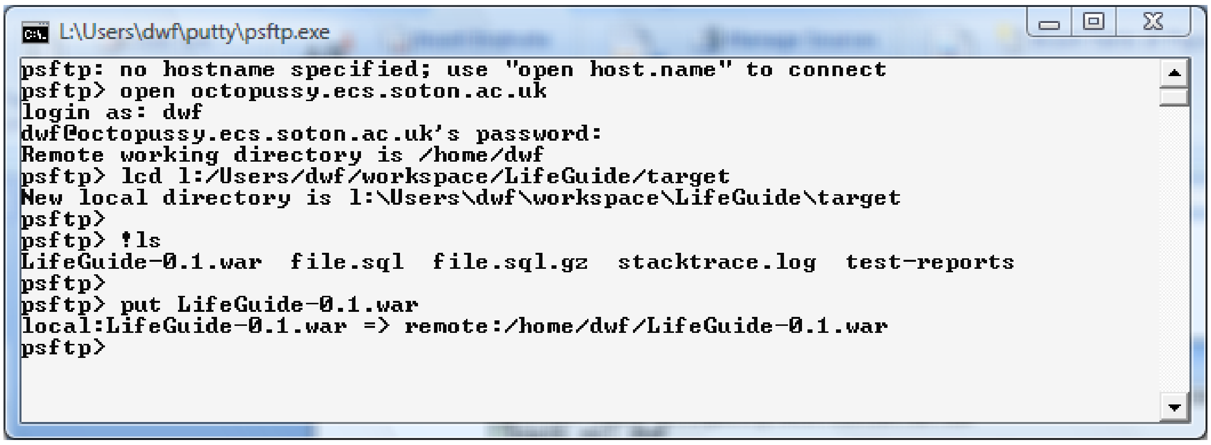
4.3.3 Copying the WAR file

Using ftp (for example, the psftp program that is bundled with putty[11]) you need to copy the WAR file to the server. The location in our setup is `/projects/lifeguide/tomcats/staging/webapps`

The example screenshot here (Figure 9) uses psftp:

First make sure that the local directory is where the newly compiled WAR file is. By default the remote directory will be your home file directory on the server, and there is no need to change this.

```
    put LifeGuide-0.1.war
```



```
psftp: no hostname specified; use "open host.name" to connect
psftp> open octopussy.ecs.soton.ac.uk
login as: dwf
dwf@octopussy.ecs.soton.ac.uk's password:
Remote working directory is /home/dwf
psftp> lcd l:/Users/dwf/workspace/LifeGuide/target
New local directory is l:\Users\dwf\workspace\LifeGuide\target
psftp>
psftp> !ls
LifeGuide-0.1.war  file.sql  file.sql.gz  stacktrace.log  test-reports
psftp>
psftp> put LifeGuide-0.1.war
local:LifeGuide-0.1.war => remote:/home/dwf/LifeGuide-0.1.war
psftp>
```

Figure 9: Copying the WAR file to the server

4.3.4 Restarting the server (may be optional)

Depending on your web server setup, the action of copying the the WAR file to the webapps directory may be enough for deployment to start. In other cases, you will need to restart the server using the instructions here. There is no harm in restarting the server if you are not sure.

Using putty (or a similar SSH client) log on to the server. Change directory to the webapps directory for the server container. Copy the WAR file from your home directory to the webapps directory.

```
    sudo cp ~/LifeGuide-0.1.war .
```

[11] PuTTY: A Free Telnet/SSH Client: http://www.chiark.greenend.org.uk/~sgtatham/putty/

To restart the server, use the following command (details may vary depending on your local setup):

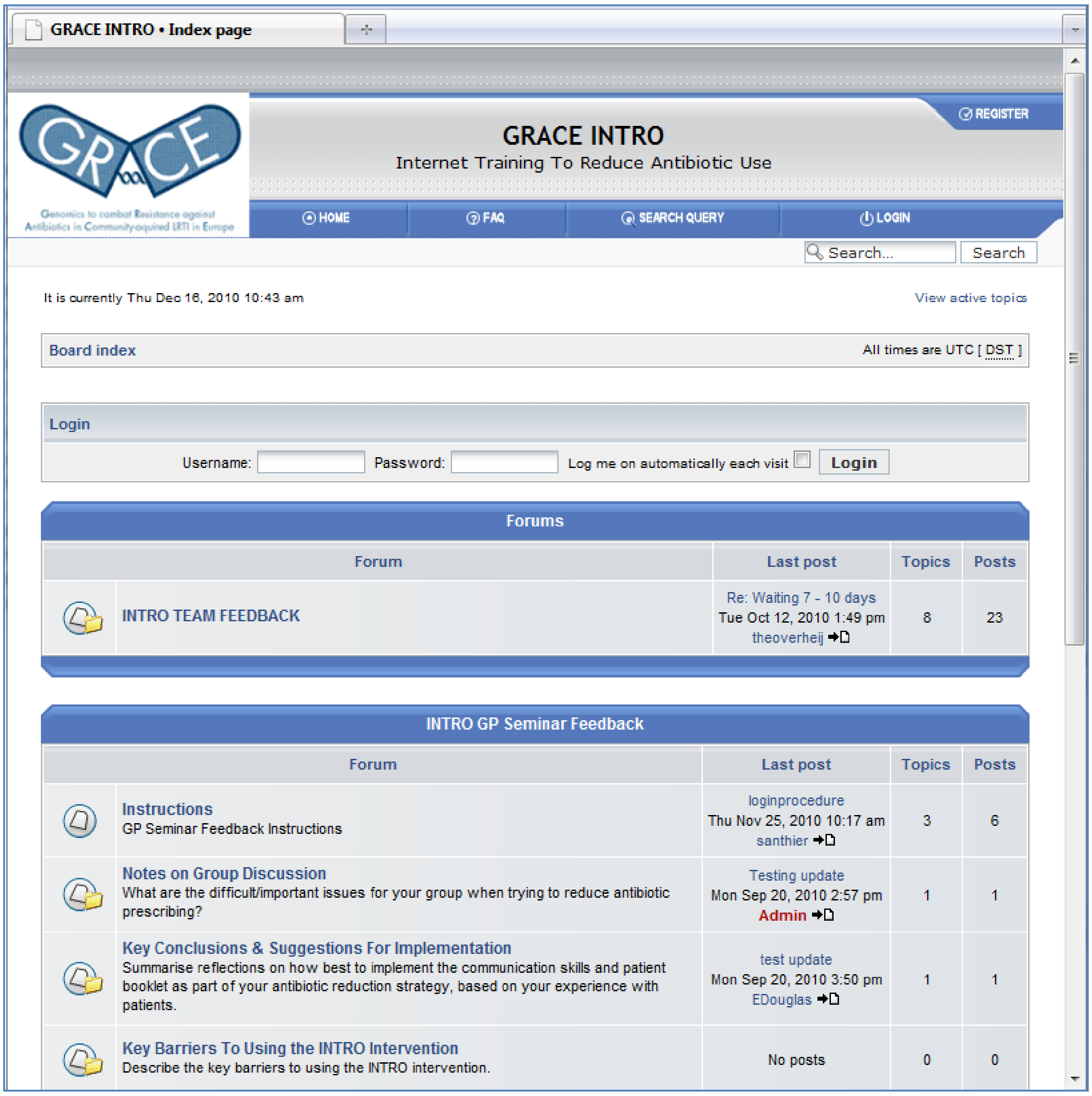
```
    sudo /sbin/service tomcat_lifeguide_production restart
```

If desired, you can monitor the server startup process by viewing the server log file:

```
    tail -f ../logs/catalina.out
```

4.4 Setting up a discussion forum for an intervention

Occasionally, an intervention owner will request that a forum is set up for discussion of the intervention – this could be discussion between researchers on the intervention (see Figure 10), or possibly for discussions between the end users of the intervention. We used phpBB for the forums, which seems to work well. To avoid having to make a new installation of phpBB for each forum, we adopted a procedure that allows multiple forums to be hosted with one installation of phpBB. As new forums are not requested very frequently, it is a good idea to have a written record of the steps involved, as they are easy to forget.



It is currently Thu Dec 16, 2010 10:43 am

View active topics

Board index

All times are UTC [DST]

Login

Username: Password: Log me on automatically each visit Login

Forums

Forum	Last post	Topics	Posts
INTRO TEAM FEEDBACK	Re: Waiting 7 - 10 days Tue Oct 12, 2010 1:49 pm theoverheij	8	23

INTRO GP Seminar Feedback

Forum	Last post	Topics	Posts
Instructions GP Seminar Feedback Instructions	loginprocedure Thu Nov 25, 2010 10:17 am santhier	3	6
Notes on Group Discussion What are the difficult/important issues for your group when trying to reduce antibiotic prescribing?	Testing update Mon Sep 20, 2010 2:57 pm Admin	1	1
Key Conclusions & Suggestions For Implementation Summarise reflections on how best to implement the communication skills and patient booklet as part of your antibiotic reduction strategy, based on your experience with patients.	test update Mon Sep 20, 2010 3:50 pm EDouglas	1	1
Key Barriers To Using the INTRO Intervention Describe the key barriers to using the INTRO intervention.	No posts	0	0

Figure 10: A phpBB forum for discussion of the GRACE INTRO intervention.

The steps required are (some locations of files may be different in your setup):

1. Choose the name for the forum (eg forum1) and configure it in /etc/httpd/conf.host/ibbre.conf, adding an Alias and a ProxyPass line:

   ```
   ProxyPass /forum1 !
   Alias /forum1 /var/www/html/phpBB3
   ```

2. reload Apache:

   ```
   sudo /sbin/service httpd reload
   ```

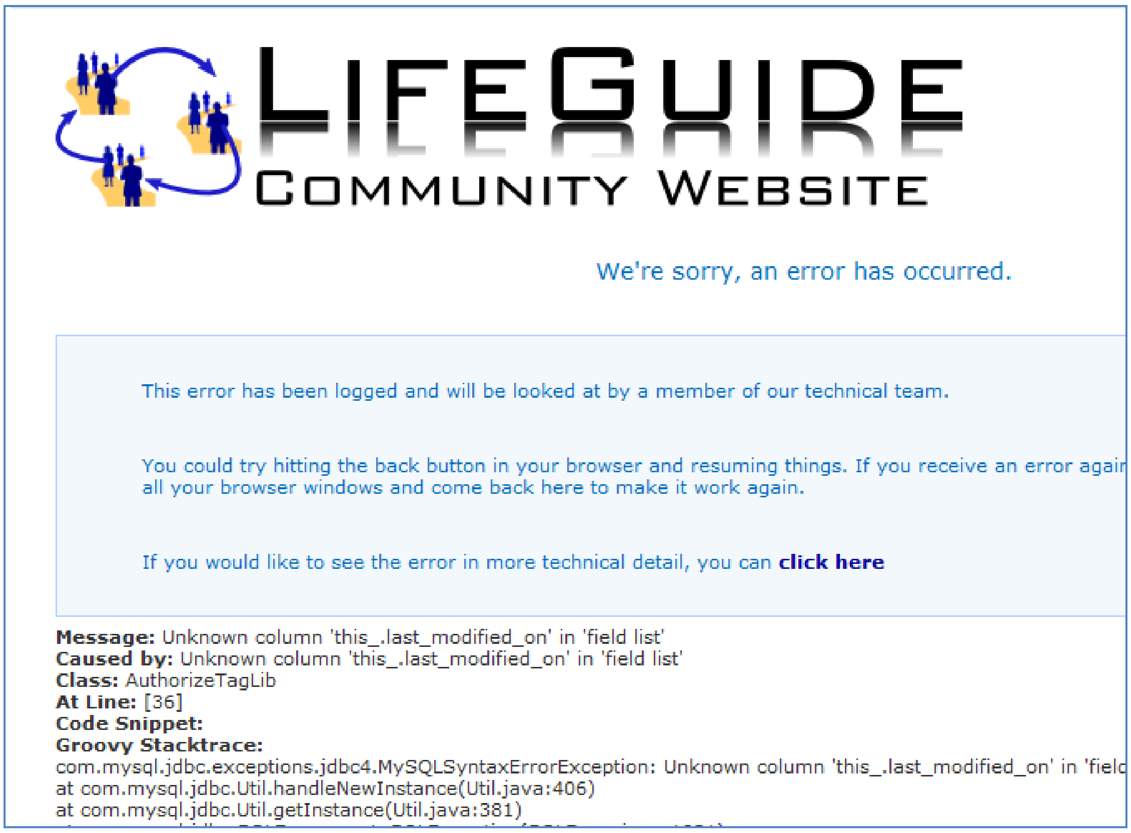
3. Go to /var/www/html/phpBBinstaller and empty the config.php file;
4. Browse to http://octopussy.ecs.soton.ac.uk/phpBBinstaller, and run the installation, with the database name as phpBB3 and using the table prefix of the forum name from step 1 followed by an underscore (e.g. forum1_).

```
<?php
// phpBB 3.0.x auto-generated configuration file
// Do not change anything in this file!

//NO REALLY, DON'T DELETE THIS. I DID IT ONCE - DON'T DO IT AGAIN.
//  -- ACO, 20/09/10

//$fName=substr($_SERVER["REQUEST_URI"], 1, strlen($_SERVER["REQUEST_URI"])-2);
//echo "<!--\n";
//echo $_SERVER["REQUEST_URI"] . "\n";
//echo strpos($_SERVER["REQUEST_URI"], "/", 2). "\n";
//echo  substr($_SERVER["REQUEST_URI"], 1, strpos($_SERVER["REQUEST_URI"], "/", 2)
-1 ) . "\n";
//echo "-->";

$dbms = 'mysqli';
$dbhost = '';
$dbport = '';
$dbname = 'phpBB3';
$dbuser = 'root';
$dbpasswd = 'octopussy';
$table_prefix = substr($_SERVER["REQUEST_URI"], 1, strpos($_SERVER["REQUEST_URI"],
"/", 2) -1 ) . '_';
$acm_type = 'file';
$load_extensions = '';

@define('PHPBB_INSTALLED', true);
// @define('DEBUG', true);
// @define('DEBUG_EXTRA', true);
?>
```

Listing 6: The config.php file for multiple forums. Do not modify this (except for the dbname, dbuser, etc, database settings, which should be set once initially according to your setup)

4.5 Modifying the production database schema

A common programming task involves adding an extra field to a domain class. To persist the domain objects to a database will require some modification (usually just an extra column in a table). In the development and staging environments, this is not a problem, as the databases are created from scratch each time. But as the production database is not cleared, you will get errors (Figure 11) created by trying to access the missing column in the table. The simplest option is often to modify the production database schema by hand.



We're sorry, an error has occurred.

This error has been logged and will be looked at by a member of our technical team.

You could try hitting the back button in your browser and resuming things. If you receive an error agair
all your browser windows and come back here to make it work again.

If you would like to see the error in more technical detail, you can **click here**

```
Message: Unknown column 'this_.last_modified_on' in 'field list'
Caused by: Unknown column 'this_.last_modified_on' in 'field list'
Class: AuthorizeTagLib
At Line: [36]
Code Snippet:
Groovy Stacktrace:
com.mysql.jdbc.exceptions.jdbc4.MySQLSyntaxErrorException: Unknown column 'this_.last_modified_on' in 'field
at com.mysql.jdbc.Util.handleNewInstance(Util.java:406)
at com.mysql.jdbc.Util.getInstance(Util.java:381)
```

Figure 11: An error message returned, indicating a problem with the database schema.

You need to be very careful when doing these operations. Make a backup of the database first (you can use the mysqldump command, as in the box “Lesson learned: use real data for testing” in Section 5.2.

In the example (Listing 7), I use a recent case where I needed to insert a column “last_modified_on” to the intervention_folder table.

```
mysql> DESCRIBE intervention_folder;
+-----------------------+--------------+------+-----+---------+----------------+
| Field                 | Type         | Null | Key | Default | Extra          |
+-----------------------+--------------+------+-----+---------+----------------+
| id                    | bigint(20)   | NO   | PRI | NULL    | auto_increment |
| version               | bigint(20)   | NO   |     |         |                |
| folder_name           | varchar(255) | NO   | UNI |         |                |
| owner_id              | bigint(20)   | NO   | MUL |         |                |
| main_intervention_id  | bigint(20)   | YES  | MUL | NULL    |                |
+-----------------------+--------------+------+-----+---------+----------------+
5 rows in set (0.00 sec)

mysql> ALTER TABLE intervention_folder
    -> ADD last_modified_on DATETIME NOT NULL DEFAULT "2010-12-01";
Query OK, 28 rows affected (0.07 sec)
Records: 28  Duplicates: 0  Warnings: 0

mysql> DESCRIBE intervention_folder;
+-----------------------+--------------+------+-----+---------------------+----------------+
| Field                 | Type         | Null | Key | Default             | Extra          |
+-----------------------+--------------+------+-----+---------------------+----------------+
| id                    | bigint(20)   | NO   | PRI | NULL                | auto_increment |
| version               | bigint(20)   | NO   |     |                     |                |
| folder_name           | varchar(255) | NO   | UNI |                     |                |
| owner_id              | bigint(20)   | NO   | MUL |                     |                |
| main_intervention_id  | bigint(20)   | YES  | MUL | NULL                |                |
| last_modified_on      | datetime     | NO   |     | 2010-12-01 00:00:00 |                |
+-----------------------+--------------+------+-----+---------------------+----------------+
6 rows in set (0.00 sec)

mysql> SELECT * FROM intervention_folder LIMIT 3;
+----+---------+-----------------+----------+----------------------+---------------------+
| id | version | folder_name     | owner_id | main_intervention_id | last_modified_on    |
+----+---------+-----------------+----------+----------------------+---------------------+
|  2 |       8 | Adrian's Folder |       27 |                  763 | 2010-12-01 00:00:00 |
|  4 |      36 | POWeR           |       13 |                  805 | 2010-12-01 00:00:00 |
| 13 |       2 | NCSCT           |       21 |                 NULL | 2010-12-01 00:00:00 |
+----+---------+-----------------+----------+----------------------+---------------------+
3 rows in set (0.00 sec)

mysql>
```

Listing 7: An example of modifying the production database schema.

5 Commonly used Grails features

In this section, we describe features of Grails that are used in IBBRE, that are used frequently, and may be counter-intuitive. Also, we describe features that should be used in future, but are currently underemployed. The book by Smith and Ledbrook (Smith & Ledbrook, 2009) is highly recommended.

5.1 Access Control

Security and access control is a very important feature, as certain users (administrators) should be able to access features that other users can't. To do this, we use the Spring Security (formerly called Acegi Security) plugin for Grails[12]. Domain classes called `User`, `Role` and `Requestmap` are created with this plugin.

The class `User` is self-explanatory, representing a single user of the system, with a name, email, encrypted password, and so on. It can easily be extended by added other fields as required.

`Role` represents a level of access and which users belong to each role.

`Requestmap` represents which URLs can be accessed by users from which roles. These can be set up in the Bootstrap.groovy file for the development, testing and staging environments (Listing 8). However, remember that the Bootstrap.groovy works differently depending on the environment. In our setup, the Requestmaps are created for the development environment, but not for the production environment (the production database is not wiped between restarts, so we do not want to repeatedly add data that is already there). So, you will also need to add the new Requestmap to the production database by hand (see Listing 10).

```
// Let moderators modify users
new Requestmap(url: '/admin/user/**', configAttribute: "ROLE_MODERATOR").save()
new Requestmap(url: '/manager/user/**', configAttribute: defaultRole).save()
```

Listing 8: An extract from Bootstrap.groovy, showing the Requestmap mechanism for access control.

A grails service called `AuthenticateService` is also provided that allows you to find the current user, and which access role(s) they have. For example, in Listing 9 (part of `InterventionController.groovy`) `authenticateService` is used to find the logged-in user, and also to check if the user has administrator rights.

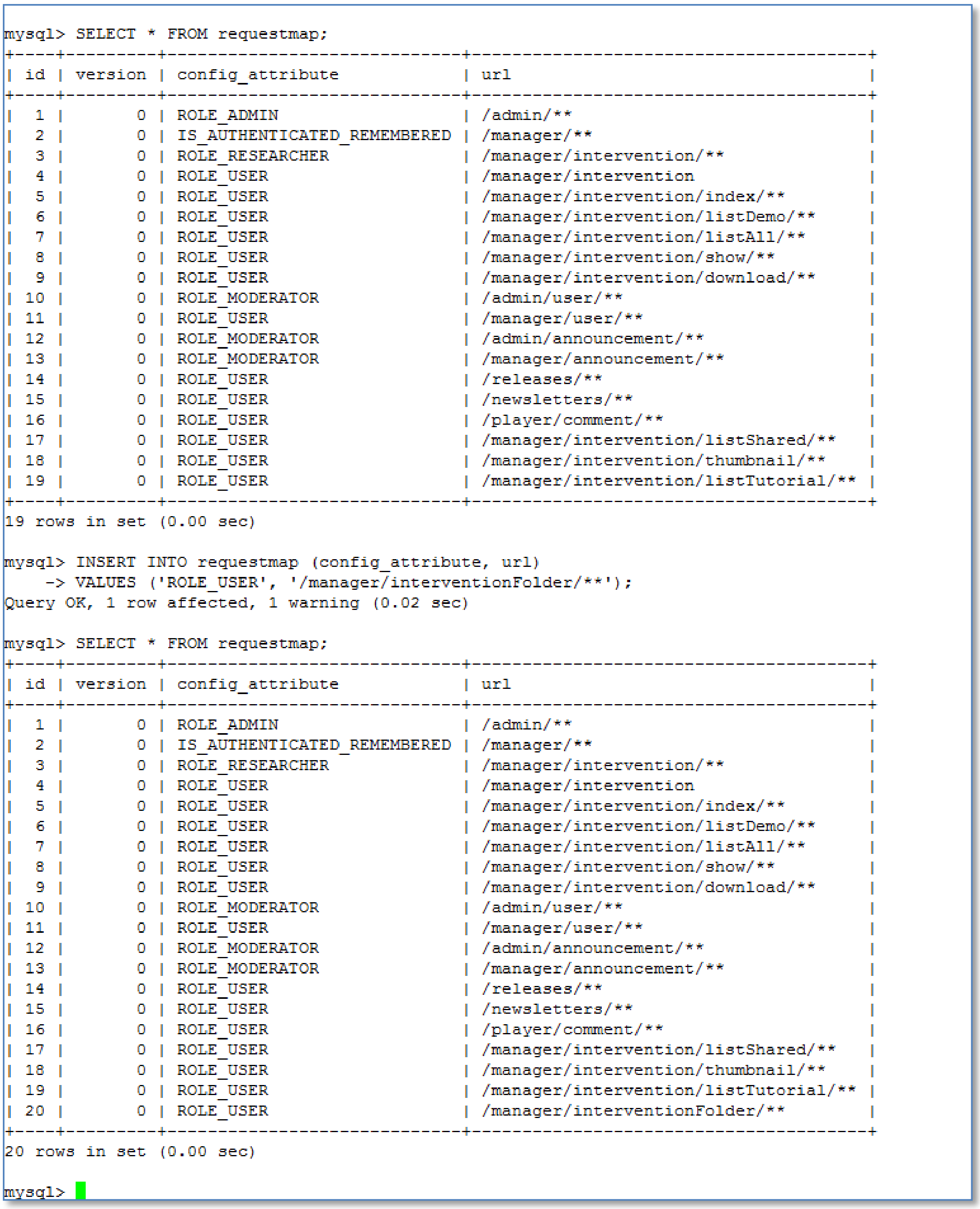
5.1.1 A possible alternative – Shiro

As an alternative to Spring Security, it may be worth considering the use of Apache Shiro[13]. This uses a similar mechanism of users and roles, but the access control is specified in each controller – removing the need for setting up Requestmaps in the Bootstrap.groovy file or in the database. See Listing 11 for an example, and refer to the Grails documentation for more details.

[12] See http://www.grails.org/AcegiSecurity+Plugin+-+Basic+Tutorial+with+Annotations for a basic tutorial.
[13] Apache Shiro: http://shiro.apache.org/

```
class InterventionController {
	AuthenticateService authenticateService

// …

def delete = {
	def intervention = Intervention.get( params.id )
	def user = authenticateService.userDomain()

	if (!intervention) {
		flash.message = "Intervention not found with id ${params.id}"
		redirect(action:list)
	}
	boolean userIsAdministrator = false;
	synchronized (this) {
		userIsAdministrator = authenticateService.ifAllGranted("ROLE_ADMIN");
	}
	if(intervention.owner.id != user.id && !userIsAdministrator) {
		flash.message = "Only the owner of an intervention can delete it"
		redirect(action:list)
	}
	else { //…
```

Listing 9: Checking user access in Groovy, with AuthenticateService.



```
mysql> SELECT * FROM requestmap;
+----+---------+------------------------------+------------------------------------------+
| id | version | config_attribute             | url                                      |
+----+---------+------------------------------+------------------------------------------+
|  1 |       0 | ROLE_ADMIN                   | /admin/**                                |
|  2 |       0 | IS_AUTHENTICATED_REMEMBERED  | /manager/**                              |
|  3 |       0 | ROLE_RESEARCHER              | /manager/intervention/**                 |
|  4 |       0 | ROLE_USER                    | /manager/intervention                    |
|  5 |       0 | ROLE_USER                    | /manager/intervention/index/**           |
|  6 |       0 | ROLE_USER                    | /manager/intervention/listDemo/**        |
|  7 |       0 | ROLE_USER                    | /manager/intervention/listAll/**         |
|  8 |       0 | ROLE_USER                    | /manager/intervention/show/**            |
|  9 |       0 | ROLE_USER                    | /manager/intervention/download/**        |
| 10 |       0 | ROLE_MODERATOR               | /admin/user/**                           |
| 11 |       0 | ROLE_USER                    | /manager/user/**                         |
| 12 |       0 | ROLE_MODERATOR               | /admin/announcement/**                   |
| 13 |       0 | ROLE_MODERATOR               | /manager/announcement/**                 |
| 14 |       0 | ROLE_USER                    | /releases/**                             |
| 15 |       0 | ROLE_USER                    | /newsletters/**                          |
| 16 |       0 | ROLE_USER                    | /player/comment/**                       |
| 17 |       0 | ROLE_USER                    | /manager/intervention/listShared/**      |
| 18 |       0 | ROLE_USER                    | /manager/intervention/thumbnail/**       |
| 19 |       0 | ROLE_USER                    | /manager/intervention/listTutorial/**    |
+----+---------+------------------------------+------------------------------------------+
19 rows in set (0.00 sec)

mysql> INSERT INTO requestmap (config_attribute, url)
    -> VALUES ('ROLE_USER', '/manager/interventionFolder/**');
Query OK, 1 row affected, 1 warning (0.02 sec)

mysql> SELECT * FROM requestmap;
+----+---------+------------------------------+------------------------------------------+
| id | version | config_attribute             | url                                      |
+----+---------+------------------------------+------------------------------------------+
|  1 |       0 | ROLE_ADMIN                   | /admin/**                                |
|  2 |       0 | IS_AUTHENTICATED_REMEMBERED  | /manager/**                              |
|  3 |       0 | ROLE_RESEARCHER              | /manager/intervention/**                 |
|  4 |       0 | ROLE_USER                    | /manager/intervention                    |
|  5 |       0 | ROLE_USER                    | /manager/intervention/index/**           |
|  6 |       0 | ROLE_USER                    | /manager/intervention/listDemo/**        |
|  7 |       0 | ROLE_USER                    | /manager/intervention/listAll/**         |
|  8 |       0 | ROLE_USER                    | /manager/intervention/show/**            |
|  9 |       0 | ROLE_USER                    | /manager/intervention/download/**        |
| 10 |       0 | ROLE_MODERATOR               | /admin/user/**                           |
| 11 |       0 | ROLE_USER                    | /manager/user/**                         |
| 12 |       0 | ROLE_MODERATOR               | /admin/announcement/**                   |
| 13 |       0 | ROLE_MODERATOR               | /manager/announcement/**                 |
| 14 |       0 | ROLE_USER                    | /releases/**                             |
| 15 |       0 | ROLE_USER                    | /newsletters/**                          |
| 16 |       0 | ROLE_USER                    | /player/comment/**                       |
| 17 |       0 | ROLE_USER                    | /manager/intervention/listShared/**      |
| 18 |       0 | ROLE_USER                    | /manager/intervention/thumbnail/**       |
| 19 |       0 | ROLE_USER                    | /manager/intervention/listTutorial/**    |
| 20 |       0 | ROLE_USER                    | /manager/interventionFolder/**           |
+----+---------+------------------------------+------------------------------------------+
20 rows in set (0.00 sec)

mysql>
```

Listing 10: Adding a row to a database manually.

```
class ExampleController extends JsecAuthBase {

       static accessControl = {

       // All actions require the 'Observer' role.
       role(name: 'Observer')

       // The 'edit' action requires the 'Administrator' role.
       role(name: 'Administrator', action: 'edit')

       // Alternatively, several actions can be specified.
       role(name: 'Administrator', only: [ 'create', 'edit', 'save', 'update' ])
       }
       …
}
```

Listing 11: An example using Shiro from the Grails framework reference

5.2 Bootstrapping

Commands to initialise test data are contained in the file BootStrap.groovy. This typically creates some users and interventions in order to provide data for experimentation (as the development and staging databases are usually set to be wiped when at server restart). Such data can be satisfactory for basic testing, but for better results you should consider using a copy of real data (see the box "Lesson learned: use real data for testing").

However, for the production database, it is clearly **not** advisable to wipe the database when the server is restarted. The Bootstrap.groovy file therefore contains `if` statements that check the current environment, and take actions accordingly. For example, Listing 12 shows an extract from the Bootstrap.groovy file that creates new Roles (see Section 5.1) for non-production environments.

```
//…
if (!GrailsUtil.environment.equals("production")) {
       def defaultRole = authenticateService.securityConfig.security.defaultRole

       new Role(description:"default user group", authority:defaultRole).save()

       new Role(description:"admin group",authority:"ROLE_ADMIN").save()

       new Role(description:"moderate other users",
           authority:"ROLE_MODERATOR").save()

       new Role(description:"researcher",authority:"ROLE_RESEARCHER").save()
//…
```

Listing 12: An extract from the IBBRE Bootstrap.groovy file

Lesson learned: use real data for testing

The Bootstrap.groovy file is very useful for creating small amounts of data to try out the LifeGuide server. However, we learned that for improved testing it was necessary to use more realistic data. The advantages of this include:

Views that require pagination of list results are easier to test
There are a wider range of users, with different amounts of data to view
Server performance problems (e.g. memory leaks) can be more easily traced

One solution is to use a copy of the production database for testing. (Of course, in the early stages of building and testing a new system, such data won't exist. But for testing a new feature in a working system, this approach is very useful). For mySQL, the database copy can be made using the `mysqldump` command on the server, for example:

```
mysqldump --user=<username> --password=<password> productiondbname > file.sql
```

This may take some time, but the resulting file can be used many times, with new dumps performed only if an updated copy is needed. The dump file (`file.sql` here) should then be copied to your local machine with ftp (compressing the file before copying it will save a lot of time).

For testing, start the local copy of the server using `grails run-app` as usual. When it is running, you can restore the backed up copy of the database with:

```
mysql --user=<username> --password=<password> localdbname < file.sql
```

Finally, to log in as different users, you will have to modify some passwords in the local database. Using a graphical database tool such as MySQL Query Browser or MySQL Workbench (http://dev.mysql.com/downloads/workbench/) is easiest for this task. Find your own account in the database, and copy the encrypted password field to the rows for any other users that you wish to test for. Following this, you will be able to log in as any other user, and get the same views that they would see.

5.3 Grails Testing

One key recommendation of the SSI report is to create a suite of functional unit/integration tests. While some steps have been taken in this direction in IBBRE, there remains much to be done.

Lesson learned: functional testing

Peter Ledbrook, co-author of the book "Grails in Action" (Smith & Ledbrook, 2009), suggested on the IBBRE blog (http://blogs.ecs.soton.ac.uk/ibbre/) that we use the Geb and Spock plugins for functional testing. This seems a promising approach, and is very useful for automating security tests (e.g. making sure that regular users cannot access parts of the website that should be restricted to administrators). It is very easy for the addition of new features to cause existing features to break – and therefore it is important to be able to quickly check that this does not happen.

The installation of the plugins is currently done with the commands:

```
grails install-plugin geb
grails install-plugin spock 0.5-groovy-1.7-SNAPSHOT
```

(You should check if the version of Spock should be changed.)

Tests are kept in the test/functional directory, and are written in a fairly simple language. A small example is:

```
def "Test regular user CANNOT access restricted stuff"() {
        given:
        login "demo_bob", "demo_bob"

        when:
        go "admin/user/index"

        then:
        verifyUnauthorised()
}
```

There are three main sections to most tests, an optional setup section ("given"), a mandatory stimulus section ("when"), and a mandatory response check section ("then"). In this test, `login` and `verifyUnauthorised` are separate helper functions to simplify coding.

There are also useful shorthands that allow testing on multiple sets of data. For example, the next test is performed twice for different users:

```
def "Test user CAN see owned or shared folder"() {
            given:
            login username, username

            when:
            go "manager/interventionFolder/show/1"

            then:
            $().text().contains("Owner")
            $().text().contains("bob")

            where:
            username << ["bob", "steve"]
      }
```

6 Recommendations

6.1 Testing

Testing should be improved, and be planned for in project schedules. Before performing the weekly server update, the tests should be run to improve the confidence in the code. Improving test coverage will take some time – it is recommended that whenever a problem is reported (for example a user cannot access a page that they should be able to), then a new functional test is added for that issue. In addition, other tests should be added as time permits. It is important to run the tests after each major code change, and to keep the tests up to date as necessary.

6.2 Code refactoring

Refactoring of repeated elements in the code needs to be done. In particular, checking of access permissions is repeated in several places. In the early stages of coding, the permission checking was not too complex, but with the addition of intervention folders, and of demo and tutorial intervention types, the checking takes several lines of code, which is replicated. It is very easy to make a mistake in this code, which may not be spotted for some time. Functional testing should reveal most of these errors, but it is better to simplify the code.

6.3 Encourage/enforce the use of bug tracking software

It seems that using a central bug tracker reduces confusion over which tasks are to be done by whom, and what progress has been made on each. If email or word of mouth is used, tasks will be forgotten, or the exact requirements for tasks will be lost. However, most users will be reluctant to report bugs or request features using a bug tracker, preferring email instead.

6.4 Making the source code more widely available

The source code should be either mirrored or transferred to an open source repository such as SourceForge. Attempts were made during the project to set up a mirror of the code on SourceForge, so that changes made in the ECS code repository were also made in the SourceForge version, but this proved impossible to set up. It may be best to simply transfer the entire code to SourceForge for long term stability, and thereafter use it as the only version. This will enable other institutions to easily get a copy of the system to run on their servers.

6.5 Database migrations

The schema of the production database that is used in IBBRE has gradually been modified as the domain classes have changed during the project. Sometimes these changes will lead to errors or involve troublesome modifications of table schemas by hand. If other institutions want to use our code on their servers, they need to make sure that their database schema matches ours. Some plugins for Grails, such as Liquibase[14], may help in automating these tasks.

[14] Liquibase: http://www.liquibase.org/manual/grails

7 Glossary

Term	Explanation
Grails environment	Grails can be run in different modes, for example, development, testing, staging, and production. Each will access different databases, and be configured in different ways. The development environment is generally for trial running on a local machine, whereas the production environment is the main server configuration that is seen by users.
Grails	A development framework that uses the Model View Controller pattern and many ancillary technologies for developing web applications.
GORM	Grails Object Relational Mapping. The mechanism that persists Grails objects to a relational database. It uses Hibernate to implement this. It is mostly transparent to the developer, although it can cause some unexpected problems.
Quartz	A scheduler that can be used as a plugin to Grails to execute tasks at specific times/intervals.
Spring security	Another Grails plugin, that provides an access control and security mechanism for web applications.
WAR	Web Archive – a single file that can be copied into a web container such as Tomcat, which will then run the web application.

8 References and Further Reading

Collins-Sussman, B., Fitzpatrick, B. W., & Pilato, C. M. (2010). *Version Control with Subversion*. Retrieved from http://svnbook.red-bean.com/.

Crouch, S. (2010). *LifeGuide Software Evaluation*.

Smith, G., & Ledbrook, P. (2009). *Grails in Action* (p. 520). Manning Publications. Retrieved from http://www.amazon.co.uk/Grails-Action-Glen-Smith/dp/1933988932.

Wills, G. (2009). A VRE to support cross-disciplinary and cross-institutional collaboration in internet-based behavioural research (IBBRE project proposal). University of Southampton.