

Tools that can optimize your development process

Software infrastructure for
Java development projects

Presentation plan

- Software Development Lifecycle Tools
 - What tools exist?
 - Where can tools help?
- Practical Applications
 - How are these tools used in the real world?
 - Work of the O2C2 group at Inland Revenue to improve the Inland Revenue Java development process

The Java Power Tools Book



Java Power Tools

Optimizing the Software Development Life Cycle

[create account](#) or [login](#)

search

[Home](#) [Book Outline](#) [Cool Tools](#) [Case studies](#) [Propose a topic](#) [Forum](#) [Polls](#) [The author](#)

Cool links

- [Home](#)
Get started here!
- [Book Outline](#)
See what's planned for Java Power Tools!
- [Cool Tools](#)
Propose cool Java tools you know!
- [Case studies](#)
Write about your SDLC experiences!
- [Propose a topic](#)
Write an article for Java Power Tools!
- [Polls](#)
Check out who uses what!
- [Participating Companies](#)
Who's been helping out

Home

This is the "official unofficial" web site for the "**Java Power Tools**" book, currently being written for publication by [O'Reilly](#). "**Java Power Tools**" is about software tools and techniques that can contribute to improving the SDLC (Software Development Lifecycle). This includes build tools such as Maven and Ant, CI tools, code quality tools, testing tools, collaborative tools, source version control, and more! It is a collaborative book, where other authors can (and are encouraged to!) contribute material on specific subjects. It is currently scheduled for release in late 2007 to early 2008.

By the way, I'm [John Ferguson Smart](#), the main author of the book.

PURPOSE OF THIS SITE

This site is a collaborative space where the author, contributors or potential contributors, reviewers, innocent bystanders, and any other participants can discuss book topics and help contribute to the final form and content of the book.

BOOK VISION

This book discusses key Java development problem areas and best practices, and focuses on open source tools that can help increase developer productivity in each area. The idea

The Java Power Tools Book

- **Tools to optimize the Java SDLC**
- Main author: John Smart
- Supported by Equinox
- Published by O'Reilly
- Contributions from lots of authors
- Website: <http://www.javapowertools.com>

EQUINOX
think IT • plan IT • do IT



O'REILLY®

Inland Revenue and O2C2

- Inland Revenue
 - Carries out a large number of internal and external Java-based developments
- O2C2:
 - The **Object Oriented Competency Centre**
 - Provide coaching and technical expertise in Java development
 - Optimize development practices and the SDLC infrastructure



Types of tools

- Many tools exist
- Different tools are useful for different tasks
 - What tools exist?
 - Where can tools help?
 - When are different tools appropriate?



Types of tools

- Let's define some categories:
 - Build Process tools
 - Version Control tools
 - Continuous Integration tools
 - Quality Metrics and code audit tools
 - Unit Testing and Test Coverage tools
 - Integration, Functional, Load and Performance Testing tools
 - Technical Documentation tools



Types of tools

- Let's define some categories:
 - **Build Process tools**
 - Version Control tools
 - Continuous Integration tools
 - Quality Metrics and code audit tools
 - Unit Testing and Test Coverage tools
 - Integration, Functional, Load and Performance Testing tools
 - Technical Documentation tools



Build Tools

- The cornerstone of the SDLC process:
 - Make your builds reproducible
 - Make your builds portable
 - Allows you to automate your build process
- Two main Java tools: **Ant** and **Maven 2**



Ant

- A procedural build scripting tool
 - Well-known and widely-used
 - Powerful and flexible
 - Lots of plugins
 - Needs lots of low-level plumbing code
 - Large scripts can become difficult to maintain



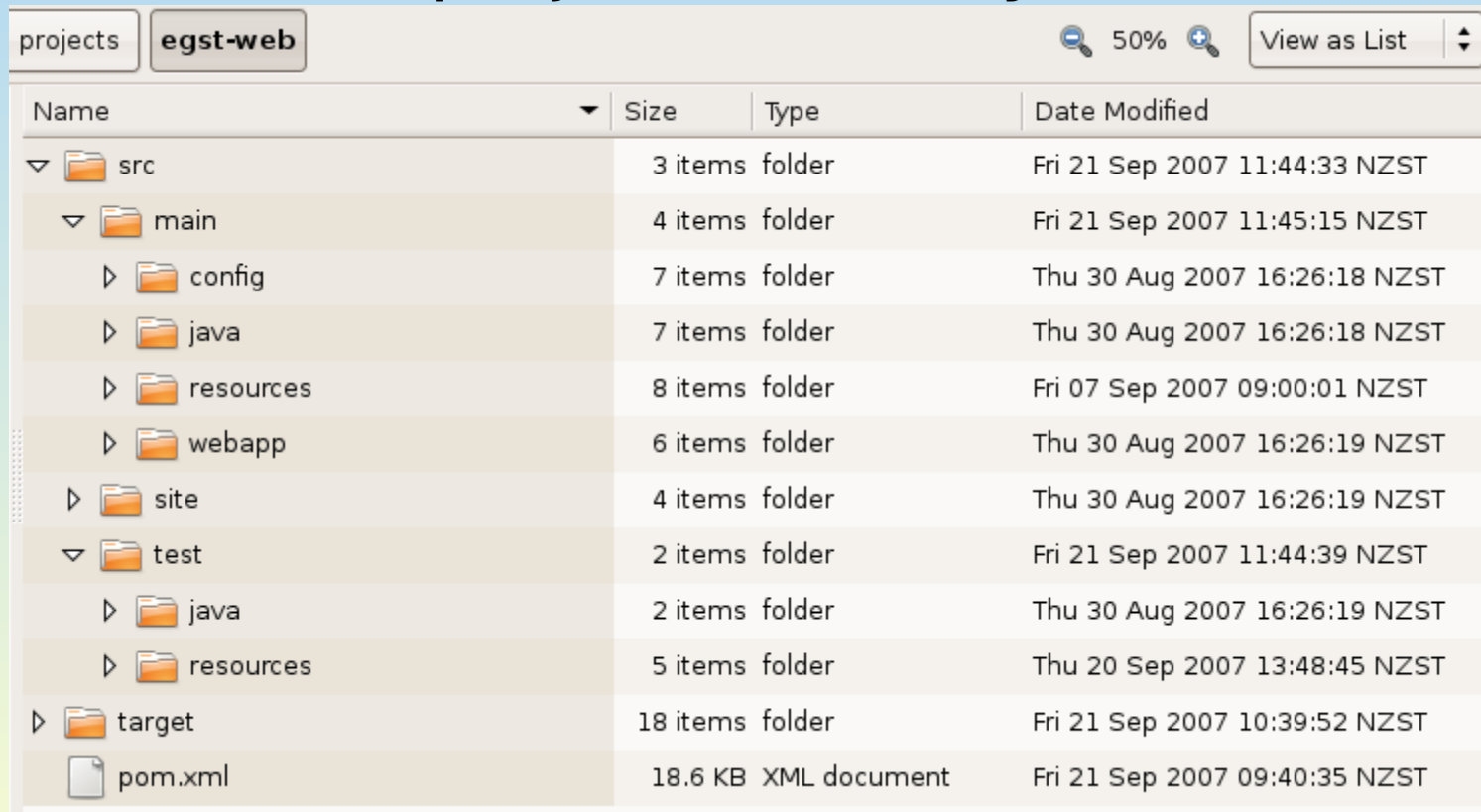
Maven 2

- A declarative build scripting framework
 - Higher level scripting
 - Strong use of standards and conventions
 - “Convention over Configuration”
 - Lots of plugins
 - Good reporting features
 - More rigid than Ant



A Standard Directory Structure

- A standard project directory structure

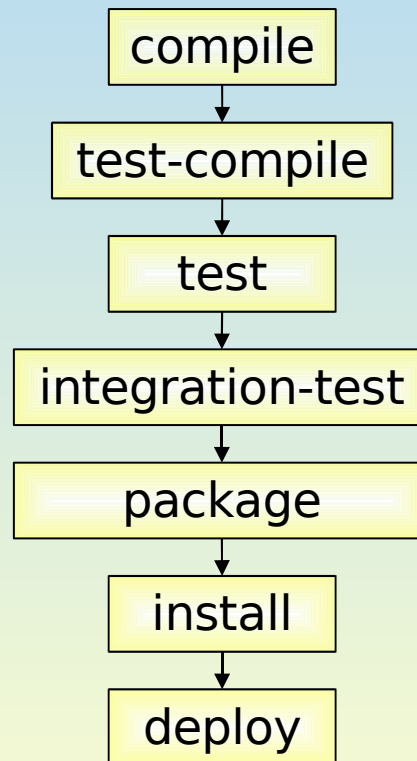


The screenshot shows a file explorer window for a project named 'egst-web'. The directory structure is as follows:

Name	Size	Type	Date Modified
src	3 items	folder	Fri 21 Sep 2007 11:44:33 NZST
main	4 items	folder	Fri 21 Sep 2007 11:45:15 NZST
config	7 items	folder	Thu 30 Aug 2007 16:26:18 NZST
java	7 items	folder	Thu 30 Aug 2007 16:26:18 NZST
resources	8 items	folder	Fri 07 Sep 2007 09:00:01 NZST
webapp	6 items	folder	Thu 30 Aug 2007 16:26:19 NZST
site	4 items	folder	Thu 30 Aug 2007 16:26:19 NZST
test	2 items	folder	Fri 21 Sep 2007 11:44:39 NZST
java	2 items	folder	Thu 30 Aug 2007 16:26:19 NZST
resources	5 items	folder	Thu 20 Sep 2007 13:48:45 NZST
target	18 items	folder	Fri 21 Sep 2007 10:39:52 NZST
pom.xml	18.6 KB	XML document	Fri 21 Sep 2007 09:40:35 NZST

Standard lifecycle

- A standard development lifecycle



Standard Build Commands

- Maven 2 - standard build commands

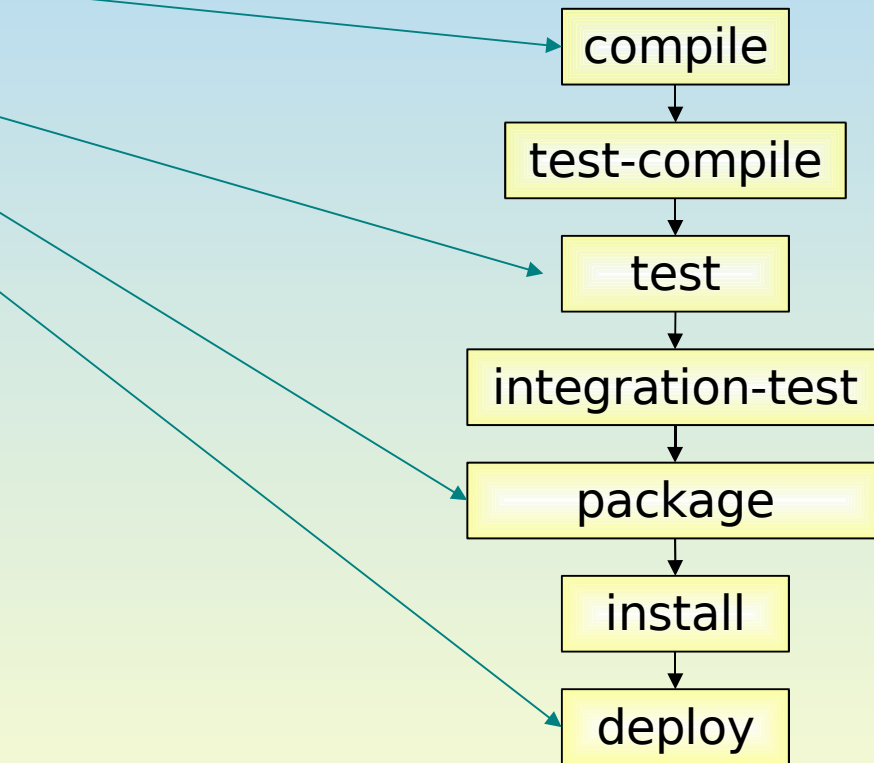
\$ mvn compile

\$ mvn test

\$ mvn package

\$ mvn deploy

...



Dependency Management

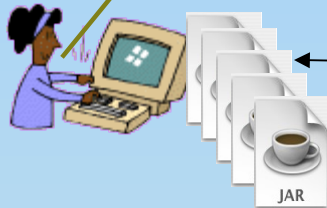
- Declarative Dependency Management
 - Important feature of Maven 2
 - A major step towards a simpler build process

Traditional Dependency Management

- Traditional approach – JAR files stored in CVS
 - Each project has its own set of JAR files
 - Unnecessary duplication
 - Hard to keep track of versions
 - Errors due to incompatible JARs
 - Overloads the source code repository

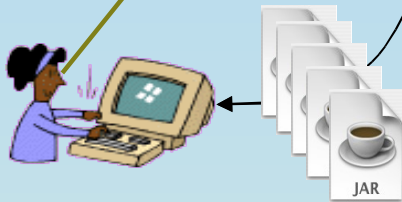
Traditional Dependency Management

I need these JARs for my project



Which versions?

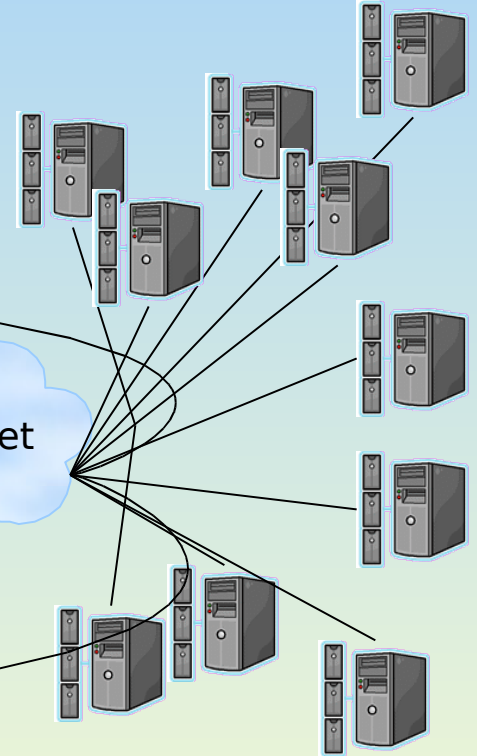
I need some too



CVS



Internet

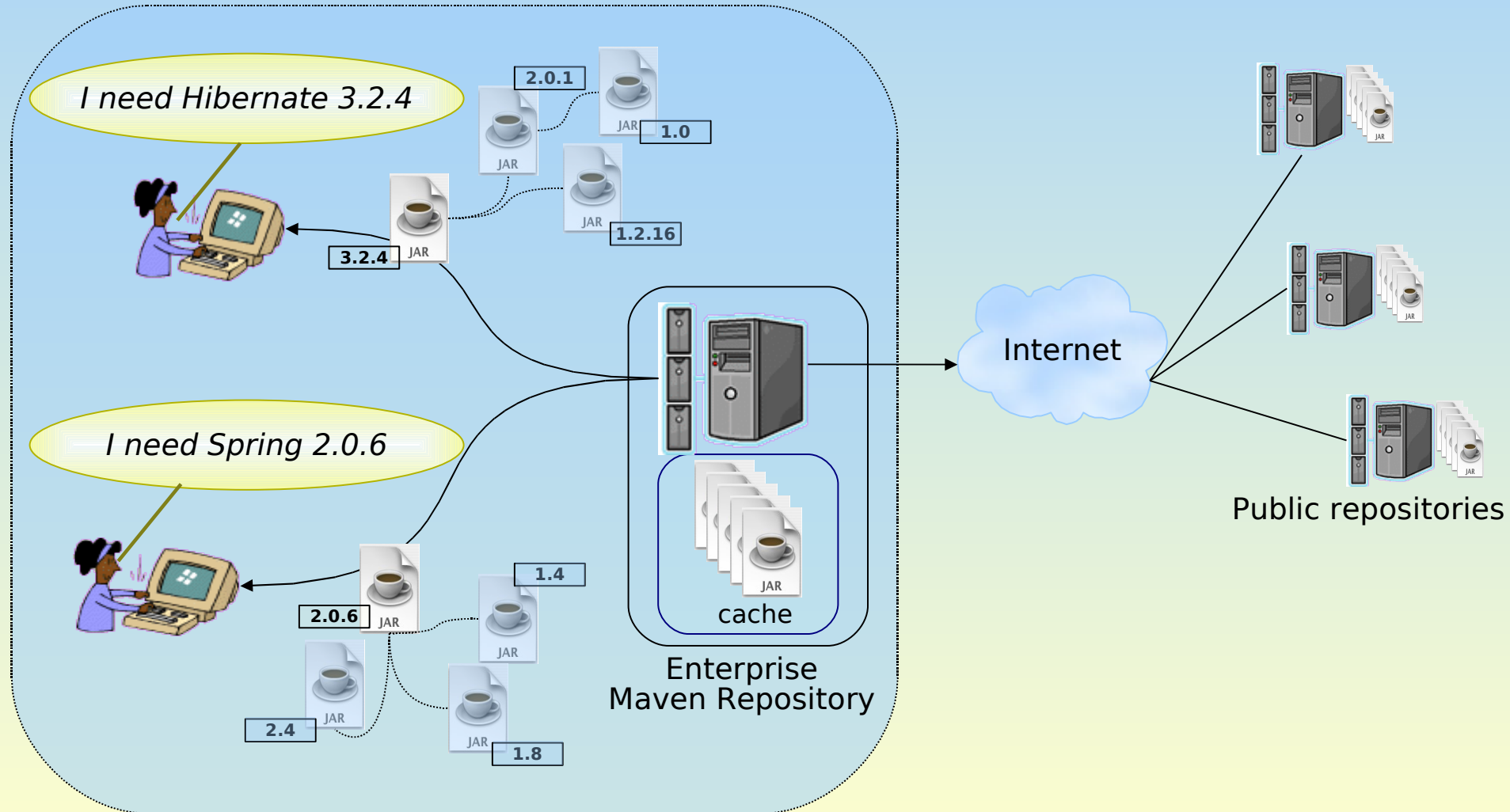


Individual Project websites

Declarative Dependency Management

- Declarative dependency management
 - Versioned JARs stored on a central server
 - Each projects “declares” what libraries it needs
 - Secondary dependencies are automatically downloaded

Declarative Dependency Management



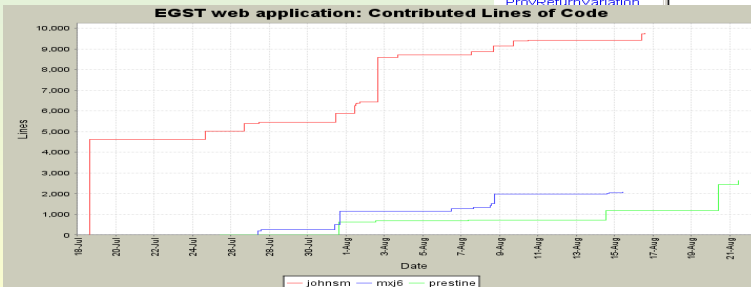
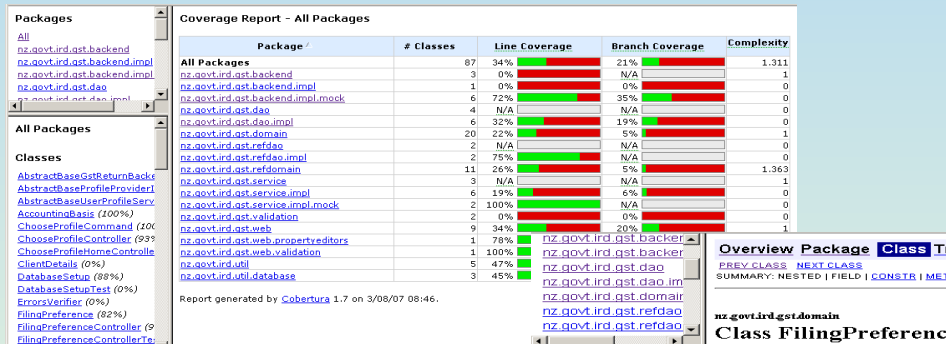
The Enterprise Maven Repository

- An Enterprise Maven Repository
 - Acts as a proxy/cache for downloaded dependencies
 - Faster and more reliable than always going to the internet
 - Store third-party and internal libraries as well



Maven 2 Reporting

- Maven 2 - reporting
 - Powerful and easy-to-use reporting features



Inland Revenue Te Tari Taake

Overview

This project uses Concurrent Versions System to manage its source code. Instructions on CVS use can be found at <http://cvobook.red-bean.com/>.

Web Access

There are no online source repository listed for this project. Please check back again later.

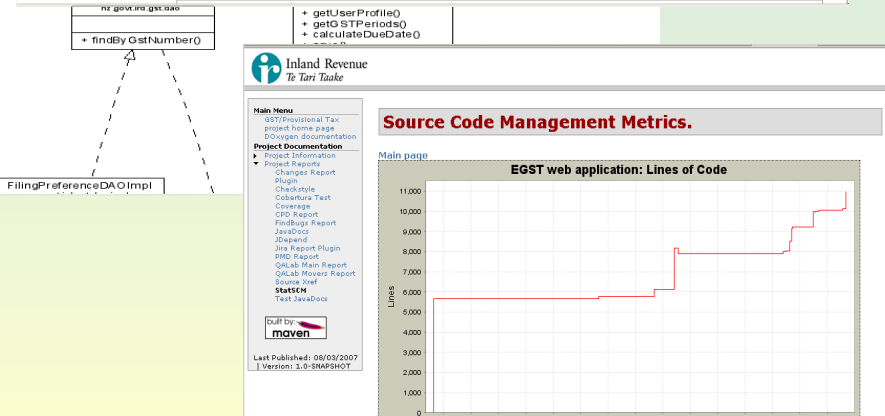
Anonymous access

This project's CVS repository can be checked out through anonymous CVS with the following instruction set. When prompted for a password for anonymous, simply press the Enter key.

```
$ cvs -d :pserver:username@cvs.dotir.ird.govt.nz:2401/cvsroot/web login
$ cvs -z3 -d :pserver:username@cvs.dotir.ird.govt.nz:2401/cvsroot/web co egst/egst-web
```

Developer access

Only project developers can access the CVS tree via this method. Substitute username with the proper value.



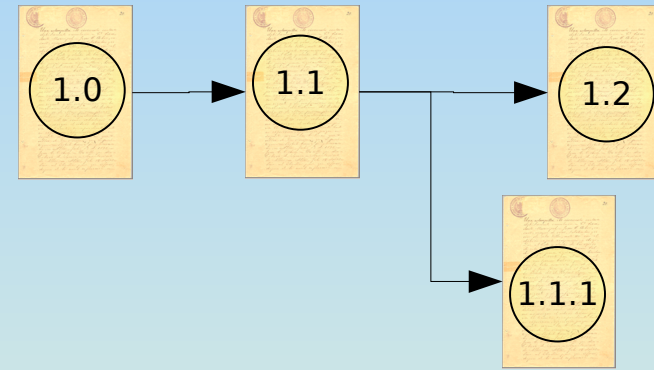
Types of tools

- Categories of tools
 - Build Process tools
 - **Version Control tools**
 - Continuous Integration tools
 - Quality Metrics and code audit tools
 - Unit Testing and Test Coverage tools
 - Integration, Functional, Load and Performance Testing tools
 - Technical Documentation tools



Version Control Tools

- Version control tools provide:
 - A central storage for source code
 - Backups and change history
 - Identify versions and releases



- Two main Open Source tools: **CVS** and **Subversion**

CVS

- A venerable open source version control tool
- Starting to show its age:
 - No atomic commits
 - Poor support for non-binary files
 - Difficult to rename or move directories
 - Slow tagging and logging

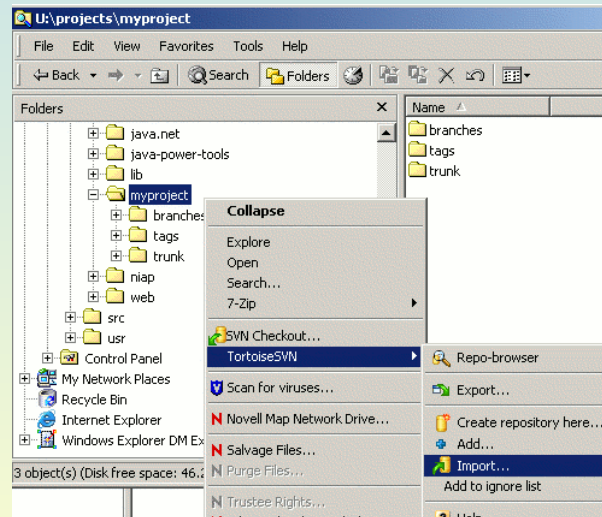
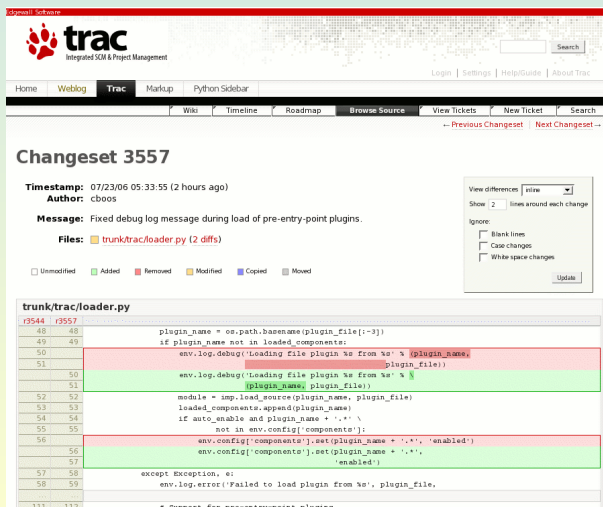
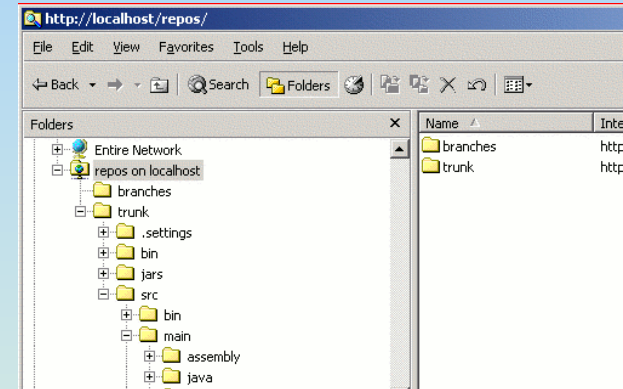
Subversion

- Designed to replace CVS
- Better adapted to Java development:
 - Atomic commits
 - Good support for binary files
 - *Fast* tagging and branching



Subversion

- Has some nice advanced features:
 - HTTP / WebDAV support
 - Windows integration (TortoiseSVN)
 - Web interface (Trac)



Types of tools

- Categories of tools
 - Build Process tools
 - Version Control tools
 - **Continuous Integration tools**
 - Quality Metrics and code audit tools
 - Unit Testing and Test Coverage tools
 - Integration, Functional, Load and Performance Testing tools
 - Technical Documentation tools



Continuous Integration

- Integrate early, integrate often:
 - Automate the build process
 - Minimise code integration issues
 - Improve visibility on the development process
 - More flexibility and agility

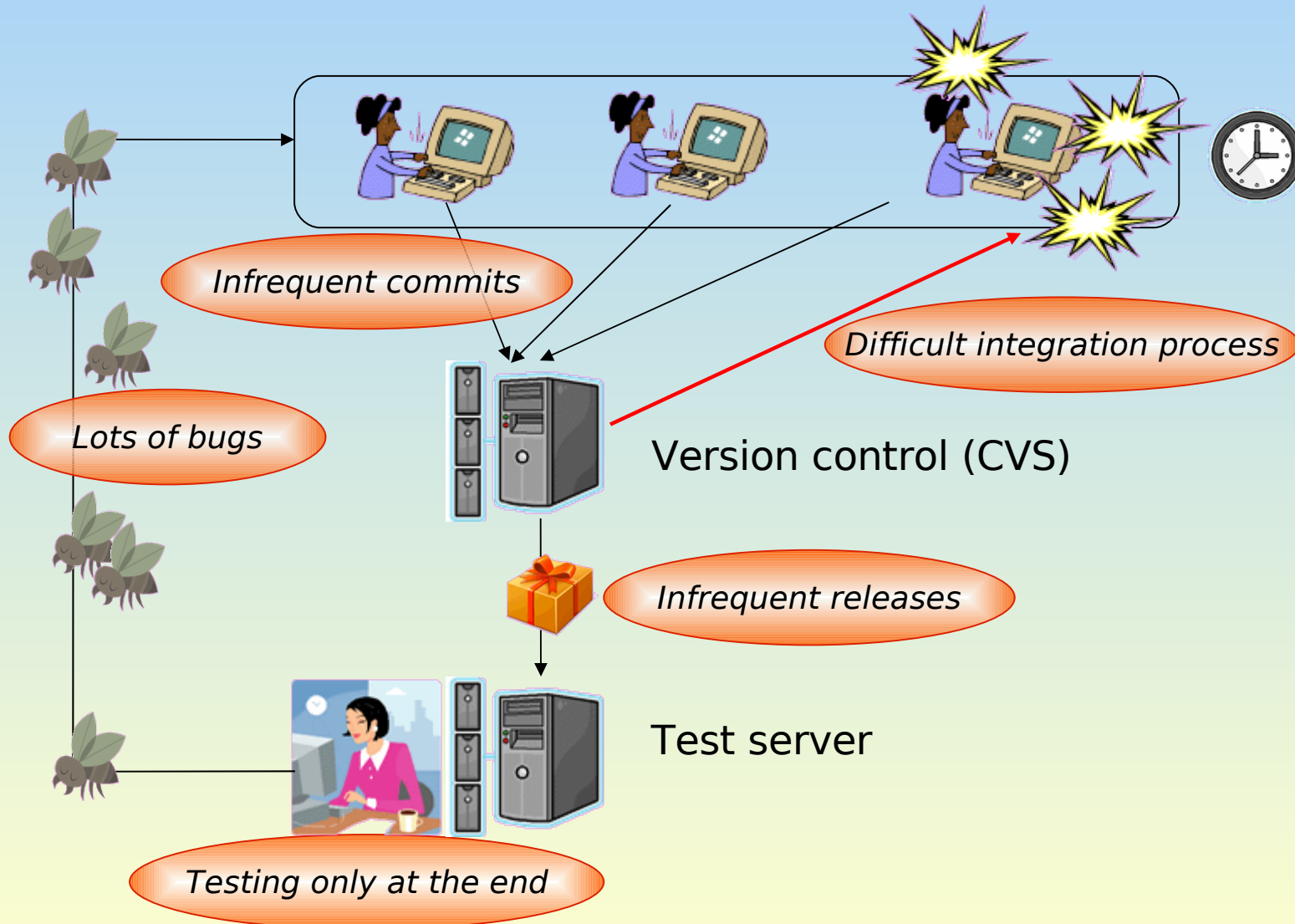


Continuous Integration

- The “Traditional” development process:
 - Coding
 - Maybe some unit tests
 - Integration
 - System and UAT testing
 - Production



The traditional approach

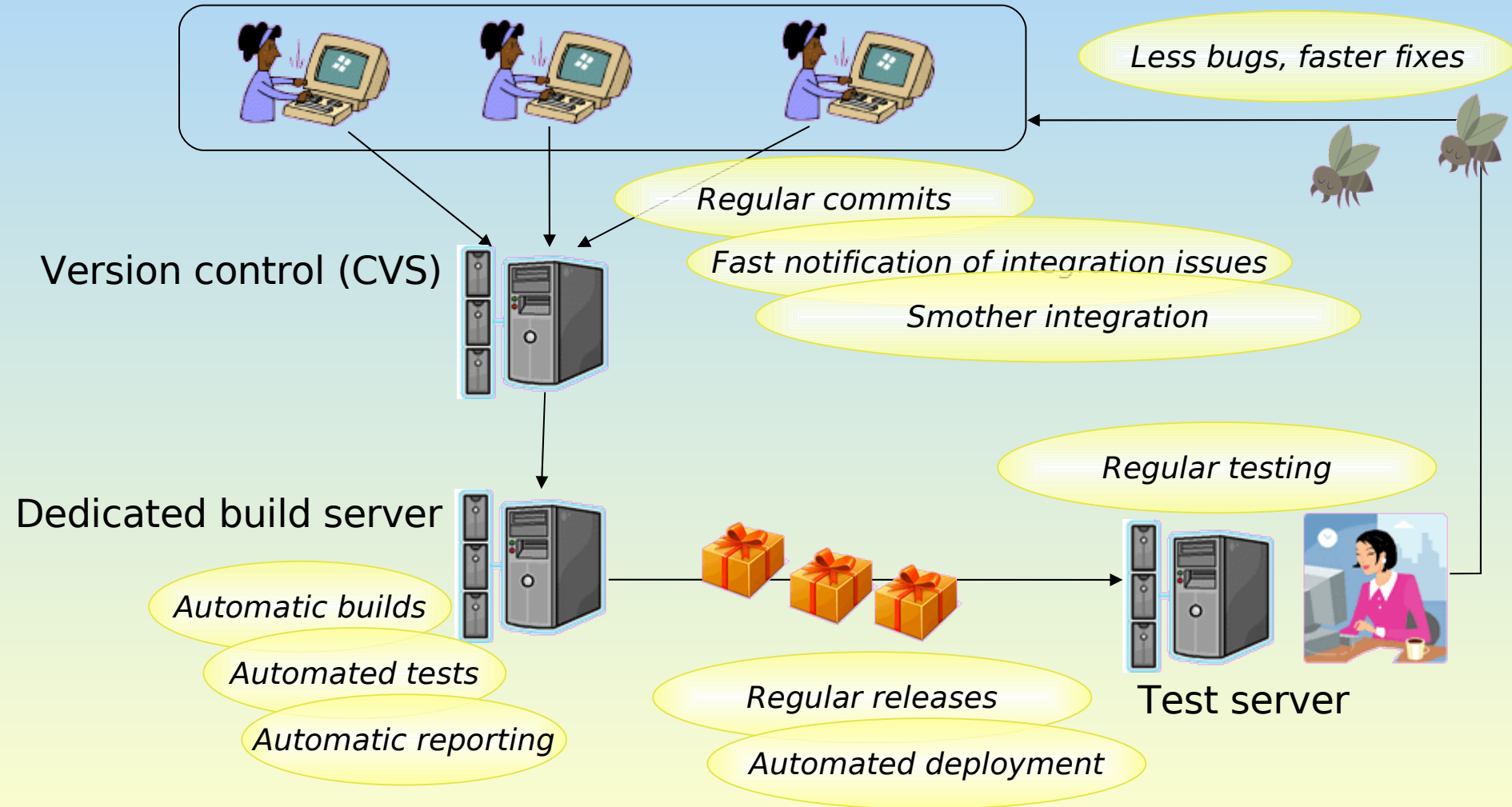


The traditional approach

- A flawed process:
 - Testing may not be done efficiently
 - Integration is long and difficult
 - Poor visibility on development progress
 - Functional tests are done too late
 - Raised issues are harder to fix
 - The client gets a sub-optimal product



How Continuous Integration helps



Advantages of Continuous Integration

- An industry best practice
 - Smoother integration
 - Automatic regression testing
 - Regular working releases
 - Earlier functional testing
 - Faster and easier bug fixes
 - Better visibility



Continuous Integration tools

- Many choices
 - CruiseControl
 - LuntBuild
 - Continuum
 - Hudson
 - ...
- Some commercial tools are also available, for example:
 - TeamCity (JetBrains)
 - Bamboo (Atlassian)
 - Pulse (Zutubi)



Continuous Integration tools

- CruiseControl
 - Mature tool
 - Strong user base
 - Lots of extensions
 - Hard to set up and maintain



CruiseControl at linux-gxu0 [18/06/06 14:22]

<u>Project</u>	<u>Status (since)</u>	<u>Last failure</u>	<u>Last successful</u>	<u>Label</u>
connectfour	waiting (14:19)		14:14	build.1

Build

RSS

Continuous Integration tools

- LuntBuild
 - Web-based, easy to use
 - Manages artifacts, dependencies and labels

LUNTBUILD

The screenshot displays the LuntBuild web application interface. At the top, the header includes the 'LUNTBUILD' logo, version '1.3.2', and navigation links like 'Home > project - creating...'. Below the header, there are tabs for 'Basic', 'VCS adaptors', 'Builders', 'Schedules', and 'Login mapping'. The 'Basic' tab is active, showing a form titled 'Creating new project (Fields marked with the * are required)'. The form contains three main sections: 'Name' with a text input field and a '*' indicating it's required; 'Description' with a large text area; and 'Project admins' and 'Project builders' sections. Each of these sections has two columns: 'Available' and 'Selected', with a list of users (Duke, dilbert, joe, john) and arrows for 'select' and 'deselect'.

LUNTBUILD 1.3.2

Home > project - creating... REFRESH IS ON system loglogout

Basic VCS adaptors Builders Schedules Login mapping [luntbuild]

Creating new project (Fields marked with the * are required)

Name *
Provide a name for this project.

Description
Provide a description for this project.

Project admins

Available		Selected
Duke dilbert joe john	→ ← select deselect	

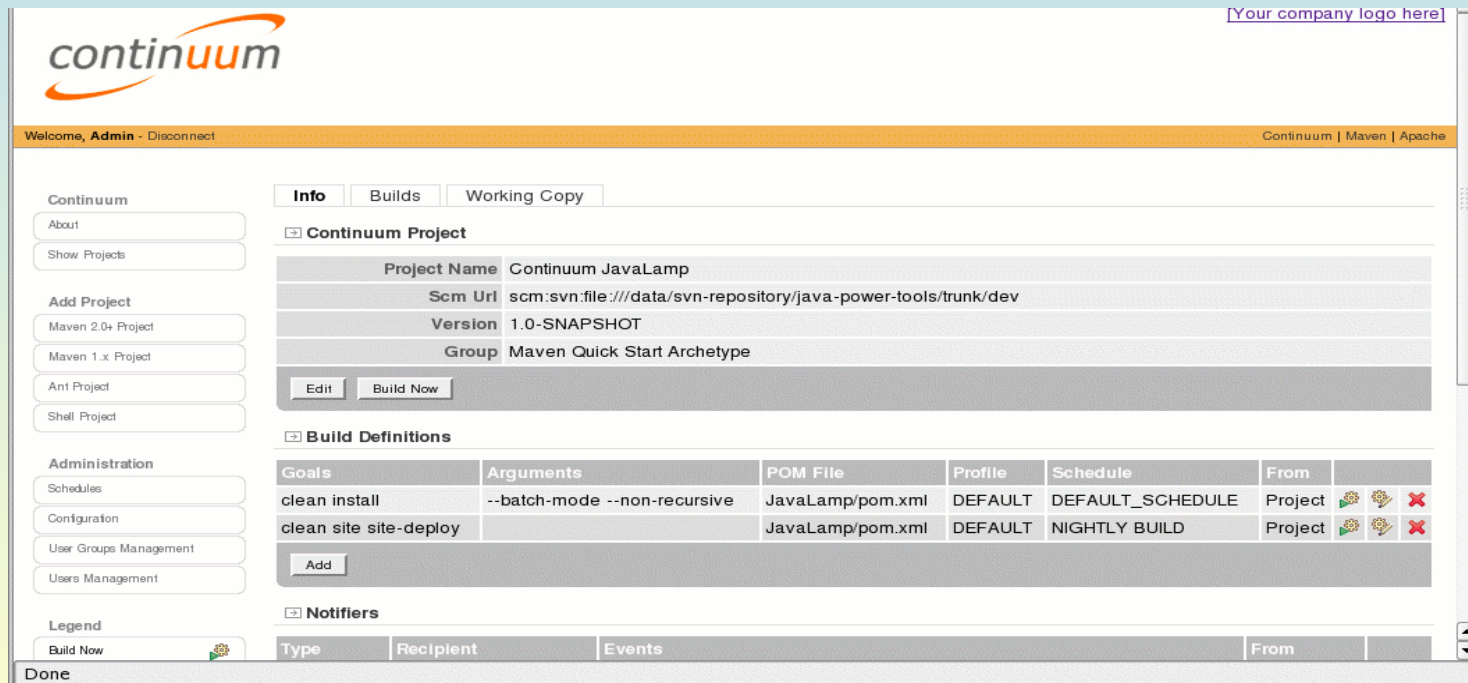
Select the users who should be assigned the role of 'project admin'.

Project builders

Available		Selected
Duke dilbert joe john	→ ← select deselect	

Continuous Integration tools

- Continuum
 - Web-based, easy to use
 - Not as feature-rich as the others
 - A bit clunky

A screenshot of the Continuum web interface. The top header bar is orange and contains the Continuum logo on the left, a "Welcome, Admin - Disconnect" message in the center, and "Continuum | Maven | Apache" on the right. Below the header, there's a sidebar on the left with navigation links: "Continuum" (with sub-links "About" and "Show Projects"), "Add Project" (with sub-links "Maven 2.0+ Project", "Maven 1.x Project", "Ant Project", and "Shell Project"), "Administration" (with sub-links "Schedules", "Configuration", "User Groups Management", and "Users Management"), and "Legend" (with a "Build Now" button). The main content area has tabs for "Info", "Builds", and "Working Copy". The "Info" tab is active, showing "Continuum Project" details: Project Name (Continuum JavaLamp), Scm Url (scm:svn:file:///data/svn-repository/java-power-tools/trunk/dev), Version (1.0-SNAPSHOT), and Group (Maven Quick Start Archetype). Below this is a "Build Definitions" table with columns: Goals, Arguments, POM File, Profile, Schedule, From, and icons. The table has two rows: "clean install" and "clean site site-deploy". At the bottom, there's a "Notifiers" section with a table with columns: Type, Recipient, Events, and From. The interface is labeled "Done" at the bottom left.

Continuous Integration tools

- Hudson
 - Web-based, easy to use
 - Lots of plugins
 - Cool user interface

Hudson

The screenshot displays the Hudson web interface. At the top, there's a blue header with the 'Hudson' logo, a search bar, and a 'logout' link. Below the header, a sidebar on the left contains navigation links: 'New Job', 'Manage Hudson', 'People', 'Project Relationship', and 'Check File Fingerprint'. The main content area features a 'Welcome to the O2C2 Continuous Integration server!' message and an 'edit description' link. A table lists three jobs: 'egst-integration-deployment', 'egst-maven-site', and 'egst-unit-tests', each with columns for status, last success, last failure, and last duration. Below this, a 'Build Queue' section shows 'No builds in the queue.' and a 'Build Executor Status' table with two executors in an 'Idle' state. At the bottom, a 'W Description' table shows build stability and test results. The footer indicates 'Hudson ver. 1.132'.

Hudson search ? logout

[Hudson](#) [DISABLE AUTO REFRESH](#)

[New Job](#) [Manage Hudson](#) [People](#) [Project Relationship](#) [Check File Fingerprint](#)

Welcome to the O2C2 Continuous Integration server! [edit description](#)

S	W	Job ↓	Last Success	Last Failure	Last Duration
		egst-integration-deployment	28 minutes (#42)	2 days (#29)	6 minutes
		egst-maven-site	22 minutes (#52)	3 hours (#48)	13 minutes
		egst-unit-tests	34 minutes (#60)	4 hours (#56)	5 minutes

Icon: [S](#) [M](#) [L](#)

W	Description	%
	Build stability: 1 out of the last 5 builds failed.	79
	Test Result: 0 tests failing out of a total of 119 tests.	100

[for all](#) [for failures](#)

Hudson ver. 1.132

The Hudson CI Server

- A project build status dashboard

Hudson

[?](#) [logout](#)

[Hudson](#) [DISABLE AUTO REFRESH](#)

[New Job](#)

[Manage Hudson](#)

[People](#)

[Project Relationship](#)

[Check File Fingerprint](#)

Build Queue

No builds in the queue.

Build Executor Status

No.	Status
1	Idle
2	Idle

Welcome to the O2C2 Continuous Integration server!

[edit description](#)

All +

S	W	Job ↓	Last Success	Last Failure	Last Duration	
		egst-integration-deployment	28 minutes (#42)	2 days (#29)	6 minutes	
		egst-maven-site	22 minutes (#52)	3 hours (#48)	13 minutes	
		egst-unit-tests	34 minutes (#60)	4 hours (#56)	5 minutes	

Icon: [S](#) [M](#) [L](#)

W	Description	%
	Build stability: 1 out of the last 5 builds failed.	79
	Test Result: 0 tests failing out of a total of 119 tests.	100

[for all](#) [for failures](#)

Hudson ver. 1.132

The Hudson CI Server

- Automatic builds whenever code is committed

The screenshot displays the Hudson CI Server interface. The top navigation bar includes the 'Hudson' logo, a search bar, a help icon, and a 'logout' link. Below the navigation bar, the left sidebar contains links for 'New job', 'Manage Hudson', 'People', 'Project Relationship', and 'Check File Fingerprint'. The main content area features a welcome message and a table of jobs. The 'Build Queue' section on the left shows no builds in the queue. The 'Build Executor Status' section shows two executors: one idle and one building 'egst-unit-tests #61', which is circled in red. The job table lists three jobs: 'egst-integration-deployment', 'egst-maven-site', and 'egst-unit-tests'.

Hudson [?](#) [logout](#)

[Hudson](#) [DISABLE AUTO REFRESH](#)

[New job](#)
[Manage Hudson](#)
[People](#)
[Project Relationship](#)
[Check File Fingerprint](#)

Welcome to the O2C2 Continuous Integration server! [edit description](#)

Build Queue
No builds in the queue.

Build Executor Status

No.	Status
1	idle
2	Building egst-unit-tests #61

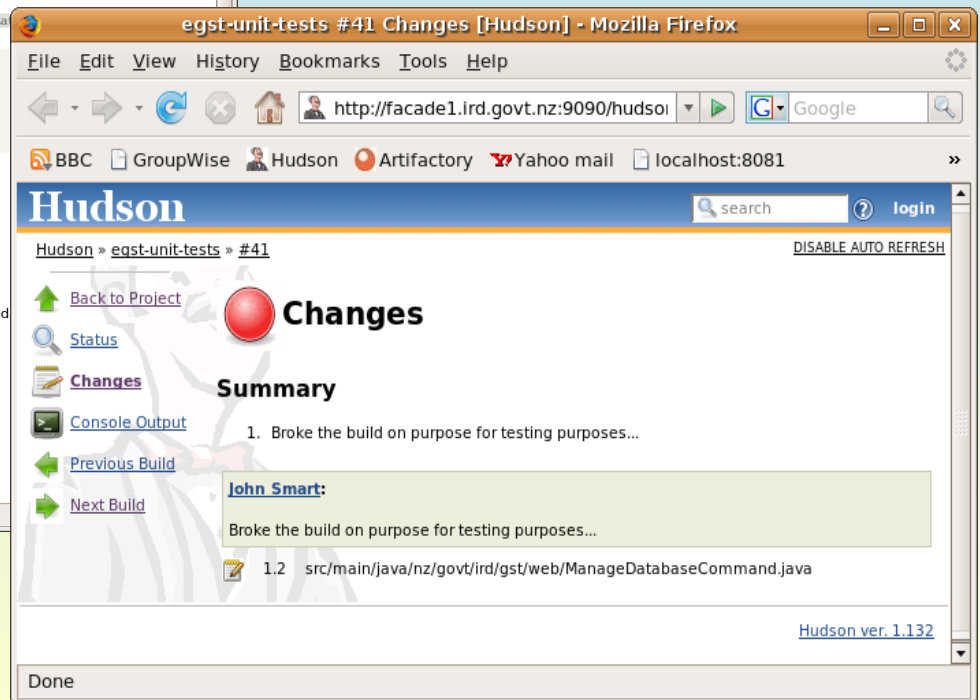
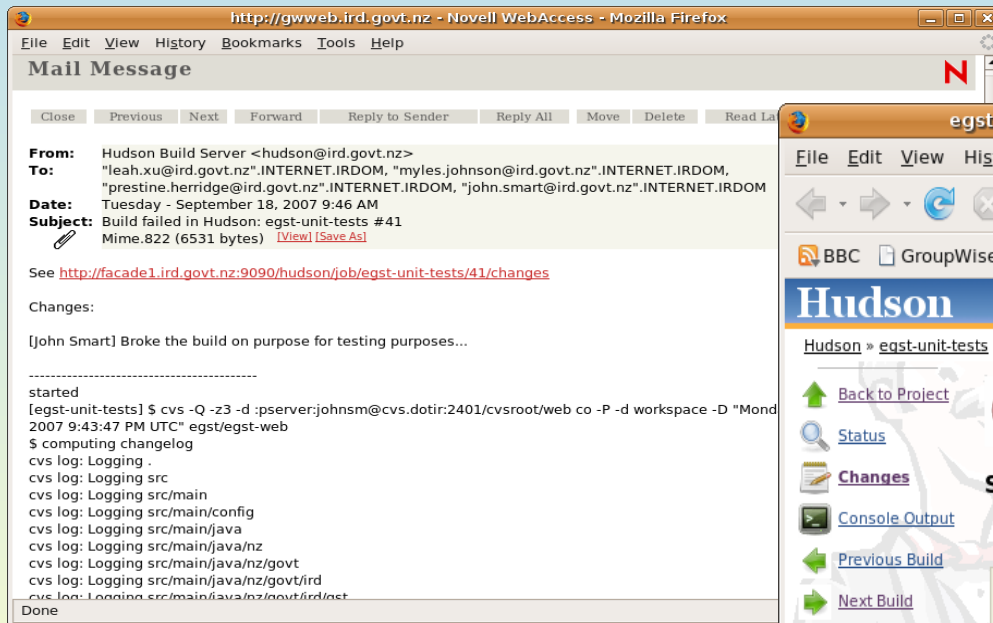
Icon: [S](#) [M](#) [L](#)

S	W	Job ↓	Last Success	Last Failure	Last Duration	
		egst-integration-deployment	3 hours (#42)	3 days (#29)	6 minutes	
		egst-maven-site	2 hours (#53)	5 hours (#48)	11 minutes	
		egst-unit-tests	3 hours (#60)	6 hours (#56)	5 minutes	

[Legend](#) [for all](#) [for failures](#)

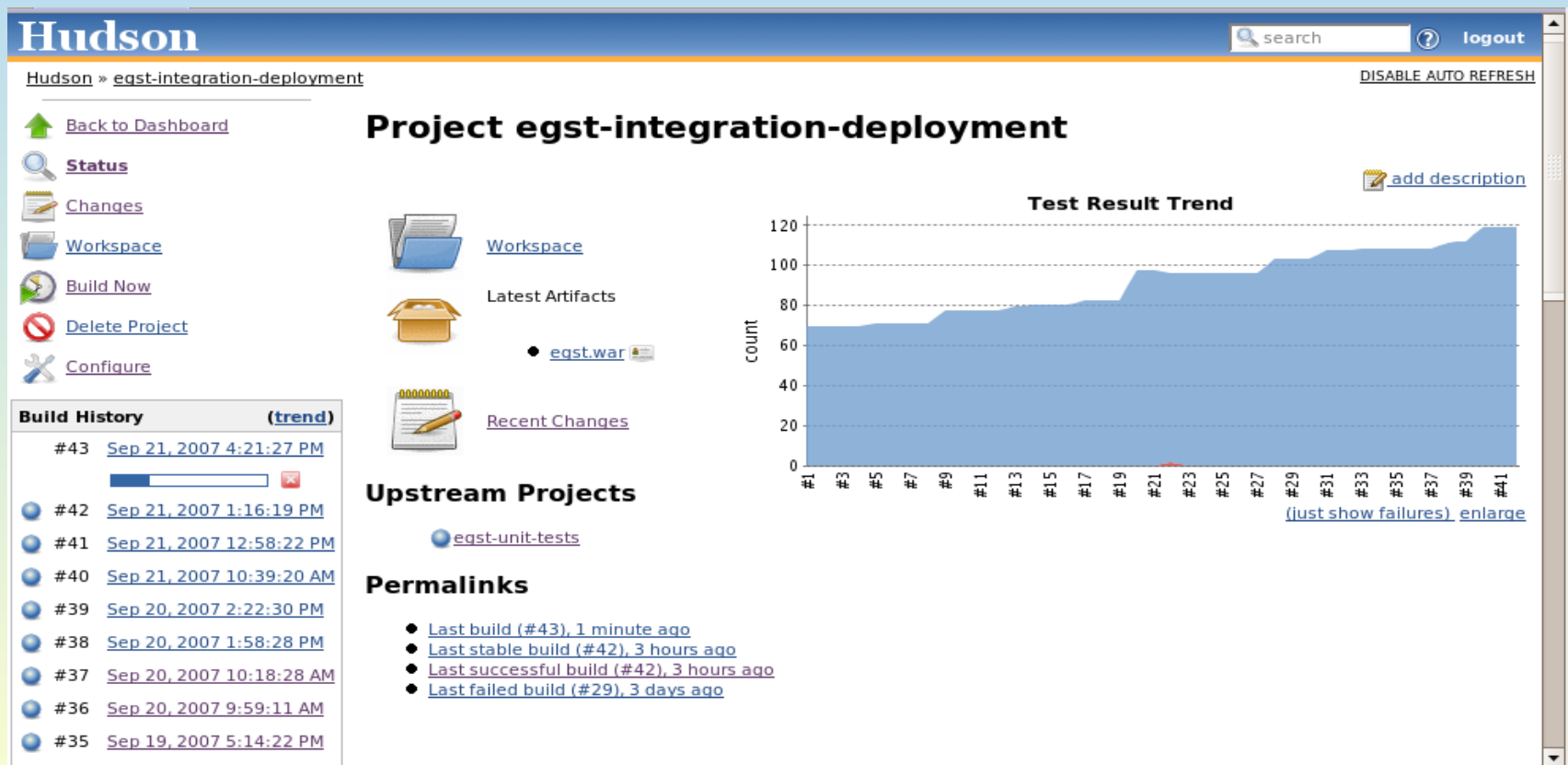
The Hudson CI Server

- Notification by email
 - When a build fails
 - When it is fixed



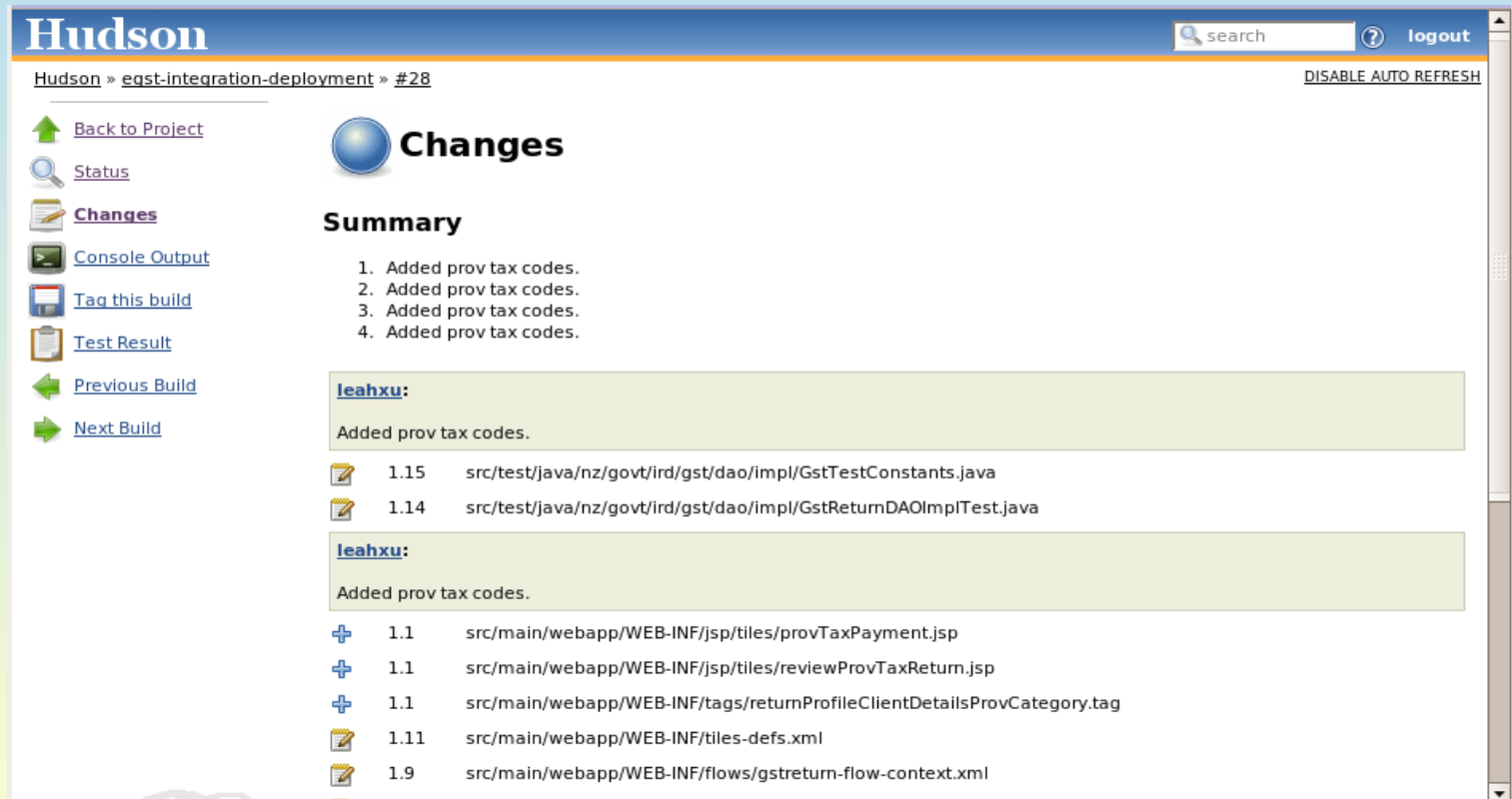
The Hudson CI Server

- Keeps a history of build results and artifacts



The Hudson CI Server

- View changes in a particular build
- Tag a build in CVS



The screenshot displays the Hudson CI Server web interface. At the top, the 'Hudson' logo is on the left, and a search bar and 'logout' link are on the right. Below the header, the breadcrumb 'Hudson » egst-integration-deployment » #28' is shown, along with a 'DISABLE AUTO REFRESH' link. A left-hand navigation menu contains links: 'Back to Project', 'Status', 'Changes' (highlighted), 'Console Output', 'Tag this build', 'Test Result', 'Previous Build', and 'Next Build'. The main content area is titled 'Changes' and includes a 'Summary' section with a list of four items: '1. Added prov tax codes.', '2. Added prov tax codes.', '3. Added prov tax codes.', and '4. Added prov tax codes.'. Below this, two change blocks are shown, each starting with 'leahxu:' and 'Added prov tax codes.'. The first block lists two changes: '1.15 src/test/java/nz/govt/ird/gst/dao/impl/GstTestConstants.java' and '1.14 src/test/java/nz/govt/ird/gst/dao/impl/GstReturnDAOImplTest.java'. The second block lists five changes: '1.1 src/main/webapp/WEB-INF/jsp/tiles/provTaxPayment.jsp', '1.1 src/main/webapp/WEB-INF/jsp/tiles/reviewProvTaxReturn.jsp', '1.1 src/main/webapp/WEB-INF/tags/returnProfileClientDetailsProvCategory.tag', '1.11 src/main/webapp/WEB-INF/tiles-defs.xml', and '1.9 src/main/webapp/WEB-INF/flows/gstreturn-flow-context.xml'.

Hudson [logout](#)

Hudson » [egst-integration-deployment](#) » #28 [DISABLE AUTO REFRESH](#)

[Back to Project](#)

[Status](#)

Changes

[Console Output](#)

[Tag this build](#)

[Test Result](#)

[Previous Build](#)

[Next Build](#)

Changes

Summary

1. Added prov tax codes.
2. Added prov tax codes.
3. Added prov tax codes.
4. Added prov tax codes.

leahxu:

Added prov tax codes.

1.15	src/test/java/nz/govt/ird/gst/dao/impl/GstTestConstants.java
1.14	src/test/java/nz/govt/ird/gst/dao/impl/GstReturnDAOImplTest.java

leahxu:

Added prov tax codes.

1.1	src/main/webapp/WEB-INF/jsp/tiles/provTaxPayment.jsp
1.1	src/main/webapp/WEB-INF/jsp/tiles/reviewProvTaxReturn.jsp
1.1	src/main/webapp/WEB-INF/tags/returnProfileClientDetailsProvCategory.tag
1.11	src/main/webapp/WEB-INF/tiles-defs.xml
1.9	src/main/webapp/WEB-INF/flows/gstreturn-flow-context.xml

Types of tools

- Categories of tools
 - Build Process tools
 - Version Control tools
 - Continuous Integration tools
 - **Quality Metrics and code audit tools**
 - Unit Testing and Test Coverage tools
 - Integration, Functional, Load and Performance Testing tools
 - Technical Documentation tools



Quality Metrics

- Why enforce coding standards?
 - Better quality code
 - Code is easier to maintain
 - Detect potential bugs
 - Train staff



Quality Metrics

- Manual code reviews are good, but...
 - Slow and time-consuming
 - Tend not to be done systematically
- Automatic code audits
 - Automatically enforce organisation coding standards
 - Detect bad coding practices and potential bugs
 - Facilitate developer training



Quality Metrics tools

- We use three complementary tools
 - **Checkstyle** – coding standards
 - **PMD** – best practices
 - **FindBugs** – potential bugs



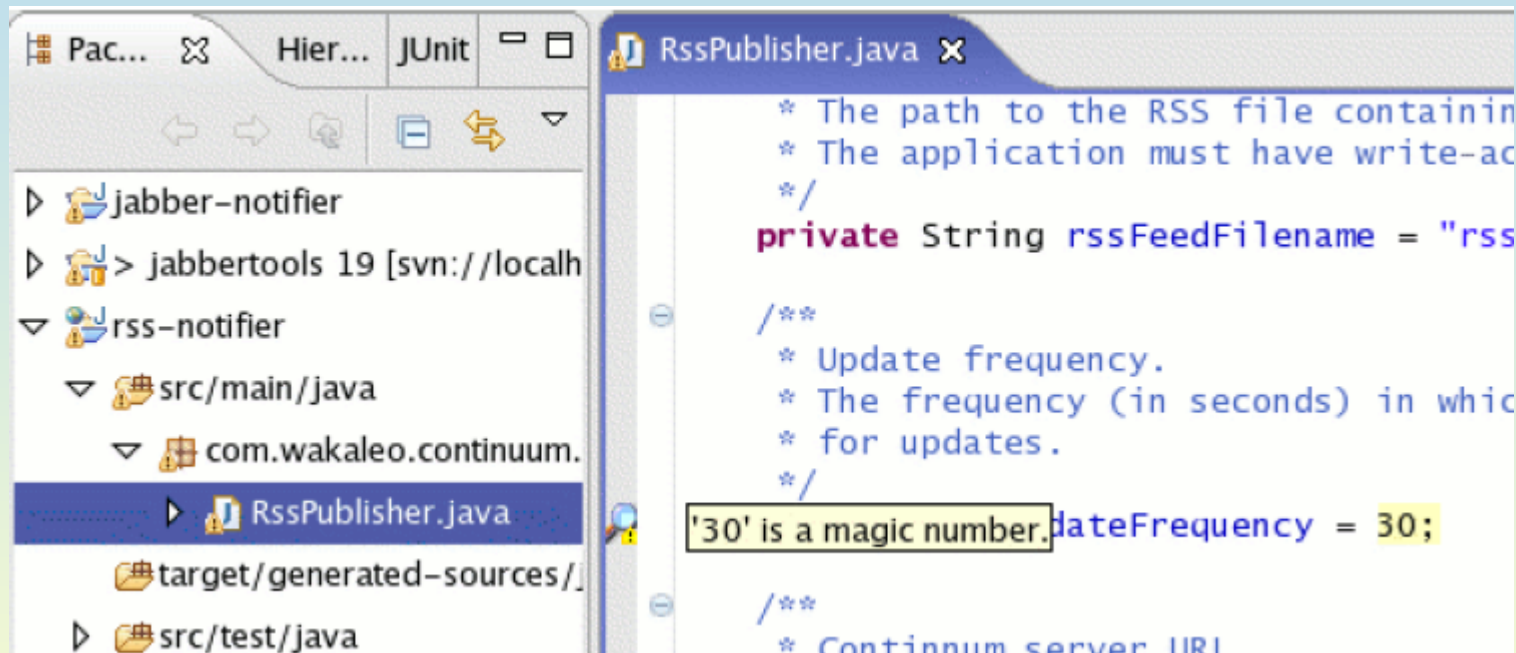
Quality Metrics

- **Checkstyle** - Enforce coding standards
 - Formatting and indentation
 - Naming conventions
 - Javadocs
 - etc...



Quality Metrics



- **Checkstyle** - Enforce coding standards
 - Eclipse plugin



Quality Metrics

- **Checkstyle** - Enforce coding standards
 - Maven reports




	Line is longer than 90 characters.	62
	Line is longer than 90 characters.	123

nz/govt/ird/gst/dao/impl/RegistrationProfileDAOImpl.java

Violation	Message	Line
	Unused import - nz.govt.ird.gst.domain.UnfiledReturnSummary.	9

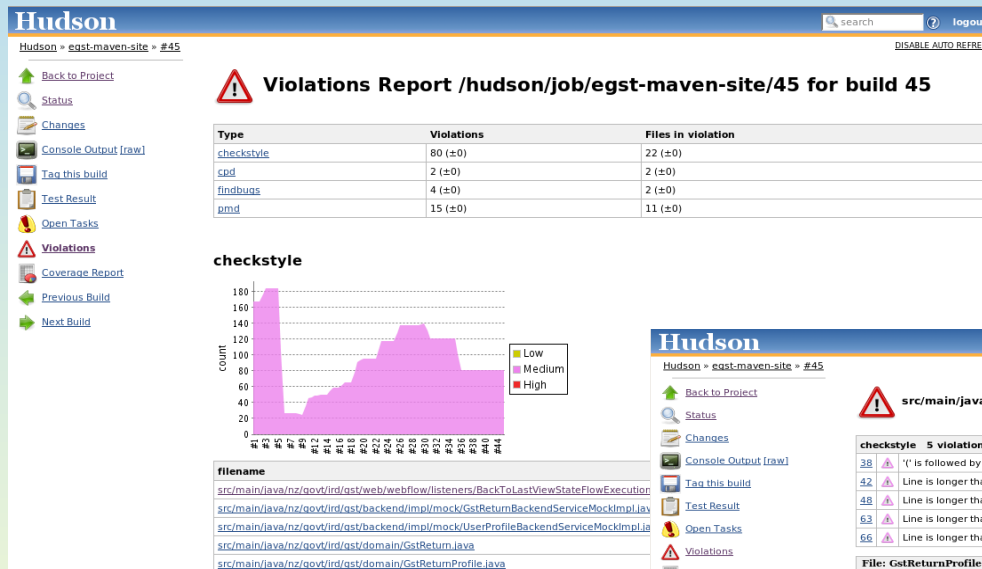
nz/govt/ird/gst/backend/UserProfileBackendService.java

Violation	Message	Line
	Line is longer than 90 characters.	29

nz/govt/ird/gst/web/GstFormAction.java

Quality Metrics

- **Checkstyle** - Enforce coding standards
 - Continuous Integration server



Hudson Hudson » east-maven-site » #45

src/main/java/nz/govt/ird/gst/domain/GstReturnProfile.java

checkstyle 5 violations

38	' ' is followed by whitespace.
42	Line is longer than 100 characters.
48	Line is longer than 100 characters.
63	Line is longer than 100 characters.
66	Line is longer than 100 characters.

File: GstReturnProfile.java Lines 29 to 76

```
29 * gauthr Myles Johnson
30 *
31 */
32
33 @Entity
34 /**
35 * Returns a list of details to identify returns saved in local database.
36 * This modifies the initial drop-down list on the home page.
37 */
38 @NamedQueries(
39 {
40     /**
41      * Returns a GstReturnProfile for a Gst Number and Period End Date
42      * A GstReturnProfile with no attached GstReturn entity should be considered to be 'Issued Return' in
43      * FIRST speak.
44      */
45     @NamedQuery(name = "gstReturnProfile.findByGstNumberAndPeriodEndDate",
46         query = "select gstReturnProfile from GstReturnProfile gstReturnProfile "
47             + "where gstReturnProfile.periodEndDate = :periodEndDate"
48             + " and gstReturnProfile.registrationProfile.clientDetails.gstRegistrationNumber = "
49             + ":gstNumber")
49 }
```

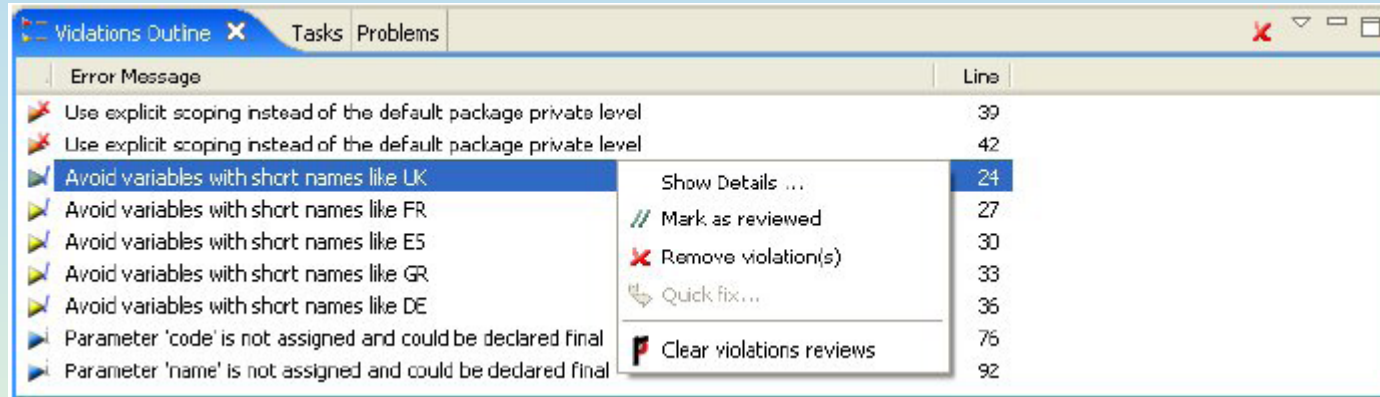
Quality Metrics

- **PMD** – Best practices
 - Empty try/catch/finally blocks
 - Incorrect null pointer checks
 - Excessive method length or complexity
 - etc...
 - Some overlap with Checkstyle



Quality Metrics

- **PMD** – Best practices
 - Eclipse plugin



Quality Metrics

- **PMD** – Best practices
 - Maven reports



PMD Results

The following document contains the results of [PMD 3.9](#).

Files

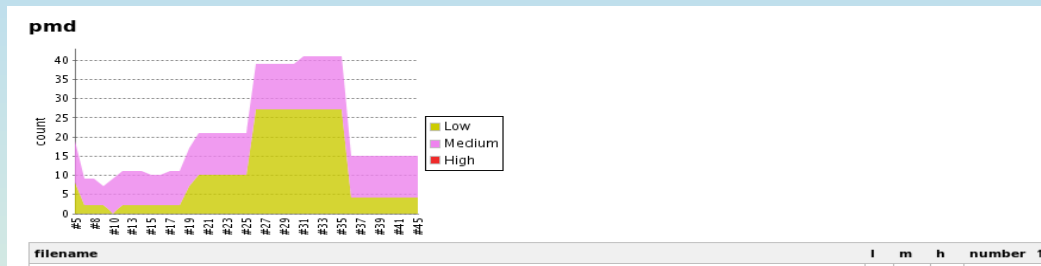
[nz/govt/ird/gst/backend/impl/mock/GstReturnBackendServiceMockImpl.java](#)

Violation	Line
Avoid unused imports such as 'java.util.Date'	4
Avoid unused imports such as 'org.joda.time.DateTimeUtils'	9

[nz/govt/ird/gst/dao/GstReturnDAO.java](#)

Quality Metrics

- **PMD** – Best practices
 - Continuous Integration server



filename

src/main/java/nz/govt/ird/gst/backend/impl/mock/GstReturnBackendImpl.java
src/main/java/nz/govt/ird/gst/dao/impl/GstReturnProfileDAOImpl.java
src/main/java/nz/govt/ird/gst/domain/GstReturn.java
src/main/java/nz/govt/ird/gst/util/database/DatabaseSetup.java
src/main/java/nz/govt/ird/gst/domain/UnfiledReturnSummary.java
src/main/java/nz/govt/ird/gst/service/impl/DatabaseAdminServiceImpl.java
src/main/java/nz/govt/ird/gst/web/ChoosePeriodController.java
src/main/java/nz/govt/ird/gst/web/ProvTaxFormAction.java
src/main/java/nz/govt/ird/gst/web/form/UnfiledReturnForm.java
src/main/java/nz/govt/ird/gst/web/validation/ProvTaxValidator.java
src/main/java/nz/govt/ird/gst/util/PlatformManager.java

Hudson

Hudson » egst-maven-site » #45

[Back to Project](#) [Status](#) [Changes](#) [Console Output \[raw\]](#) [Tag this build](#) [Test Result](#) [Open Tasks](#) [Violations](#) [Coverage Report](#) [Previous Build](#) [Next Build](#)

src/main/java/nz/govt/ird/gst/dao/impl/GstReturnProfileDAOImpl.java

pmd 2 violations

20	⚠	Avoid unused private fields such as 'LOGGER'.
63	⚠	Avoid unused local variables such as 'gstReturnProfile'.

File: GstReturnProfileDAOImpl.java Lines 11 to 30

```
11
12 /**
13  * The DAO class for GST/Prov returns.
14  */
15 public class GstReturnProfileDAOImpl extends
16     GenericDAOImpl<GstReturnProfile, Long> implements GstReturnProfileDAO {
17     /**
18      * Logger.
19      */
20     private static final Logger LOGGER = Logger
21         .getLogger(GstReturnDAOImpl.class);
22
23     /**
24      * Find the list of returns for a given IRD number.
25      *
26      * @param gstNumber
27      *     a valid IRD number (9 digit)
28      * @return a list of GstProvReturn objects, ordered by descending date.
29      */
30     @SuppressWarnings("unchecked")
```

Quality Metrics

- **FindBugs** – Potential defects
 - Potential NullPointerExceptions
 - Infinite loops
 - etc...



Quality Metrics

- **FindBugs** – Potential defects
 - Eclipse plugin



The screenshot displays the Eclipse IDE with a Java source file open. The code contains two conditional blocks. The first block checks if 'ref' is null and 'updated' is a Collection, then calls 'getObjectChanges().add(changedCollection)' and returns true. The second block checks if 'ref' is not null and 'updated' is null, then creates a new 'ChangedCollection' and adds it to 'getObjectChanges()'. A tooltip is visible over the first block, indicating a 'NP: Load of known null value' warning.

Below the code editor, the 'Problems' view is open, showing 18 errors and 288 warnings. The table below lists the first five warnings:

Description	Resource	Path	Location
Errors (18 items)			
Warnings (100 of 288 items)			
⚠ DLS: Dead store to baseDir in method nz.govt.natlib.symbols.core.searchengine.DirectoryProviderI...	library-symbols-web/src/test/java/nz/govt/natlib/symb...	line 80	
⚠ DLS: Dead store to c in method nz.govt.natlib.symbols.core.updates.ChangedLibra ChangedLibraries.j...	library-symbols-web/src/main/java/nz/govt/natlib/sym...	line 36	
⚠ DLS: Dead store to c in method nz.govt.natlib.symbols.core.updates.ChangedLibra ChangedLibrary.java	library-symbols-web/src/main/java/nz/govt/natlib/sym...	line 60	
⚠ DLS: Dead store to categories in method nz.govt.natlib.symbols.core.updates.Libra LibrarySymbolsUpd...	library-symbols-web/src/main/java/nz/govt/natlib/sym...	line 156	
⚠ DLS: Dead store to cycle0 in method nz.govt.natlib.symbols.web.pages.Application ApplicationBasePan	library-symbols-web/src/main/java/nz/govt/natlib/sym...	line 226	

Quality Metrics

- **FindBugs** – Potential defects
 - Maven reports



Files

Class	Bugs
nz.govt.ird.gst.domain.HistoryDetailEntry	4
nz.govt.ird.gst.refdomain.RefGSTProfile	3
nz.govt.ird.gst.refdomain.RefGstProvProfile	3
nz.govt.ird.gst.refdomain.RefGstProvReturn	4
nz.govt.ird.gst.refdomain.RefOldGstProvReturn	4

[nz.govt.ird.gst.domain.HistoryDetailEntry](#)

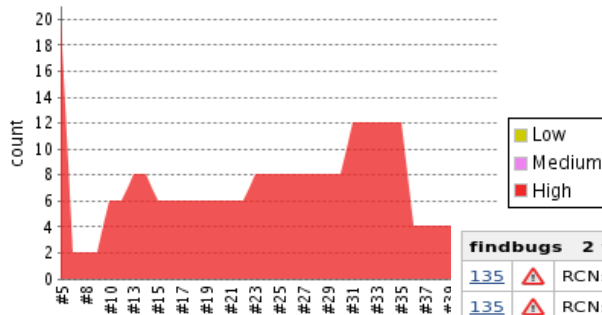
Bug	Category	Details	Line
nz.govt.ird.gst.domain.HistoryDetailEntry.getRtnPeriodEnd() may expose internal representation by returning nz.govt.ird.gst.domain.HistoryDetailEntry.rtnPeriodEnd	MALICIOUS_CODE	EI_EXPOSE_REP	27
nz.govt.ird.gst.domain.HistoryDetailEntry.getStatusDate() may expose internal representation by returning nz.govt.ird.gst.domain.HistoryDetailEntry.statusDate	MALICIOUS_CODE	EI_EXPOSE_REP	43

Quality Metrics

- **FindBugs** – Potential defects
 - Continuous Integration server



findbugs



filename

[src/main/java/nz/qovt/ird/gst/service/impl/GstReturnImpl.java](#)
[src/main/java/nz/qovt/ird/gst/web/form/UnfiledReturnForm.java](#)

findbugs 2 violations

135	RCN: Redundant nullcheck of processedReturn which is known to be null in nz.govt.ird.gst.service.impl.GstReturnImpl.java
135	RCN: Redundant nullcheck of processedReturn which is known to be null in nz.govt.ird.gst.service.impl.GstReturnImpl.java

File: GstReturnProviderImpl.java Lines 88 to 107

```
88      // decide how we will manage this
89      }
90
91      }
92
93      if (!(ReturnStatus.Saved.equals(gstReturn.getStatus()) || ReturnStatus.New
94          .equals(gstReturn.getStatus())) {
95          // TODO - throw proper exception instead of runtime exception.
96          throw new RuntimeException(
97              "TODO Error trying to fetch an unfiled return that is currently consi
98              + gstReturn.getStatus() + """);
99      }
100
101      return gstReturn;
102  }
103
104  /**
```

Types of tools

- Categories of tools
 - Build Process tools
 - Version Control tools
 - Continuous Integration tools
 - Quality Metrics and code audit tools
 - **Unit Testing and Test Coverage tools**
 - Integration, Functional, Load and Performance Testing tools
 - Technical Documentation tools



Unit Testing

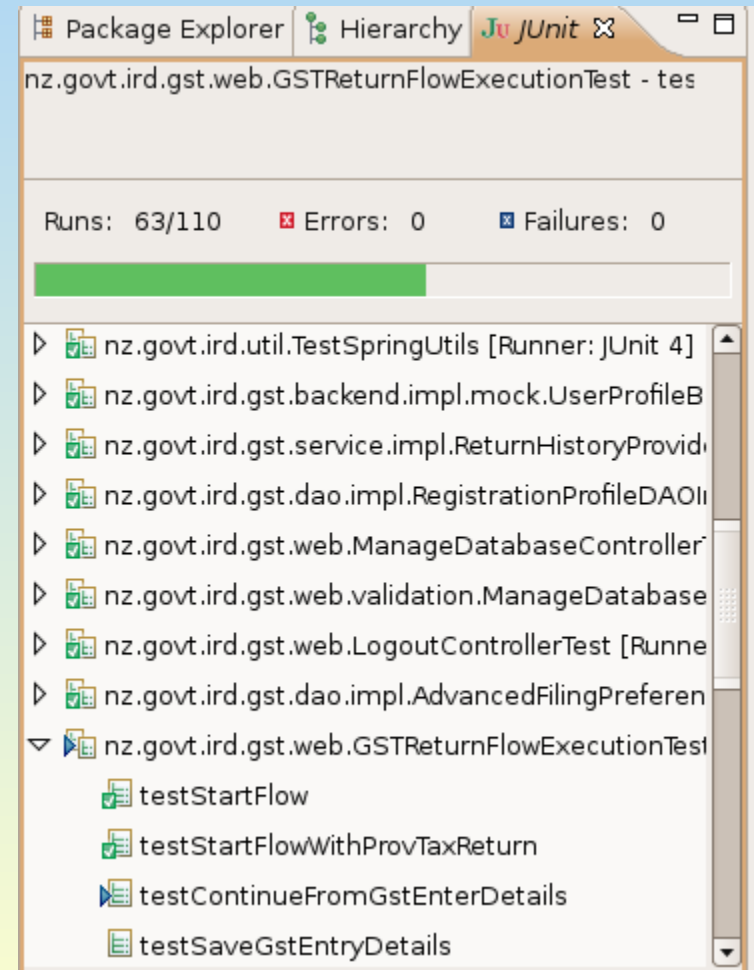
- A recommended best practice
- Unit testing results in:
 - More reliable code
 - Better designed code
 - More flexible code

Unit Testing Tools

- Main tools
 - JUnit 3.x – *the original, with lots of extensions*
 - JUnit 4 – *annotations and stuff*
 - TestNG – *the cutting edge*

Unit Testing in Eclipse

- JUnit Eclipse integration



Test reports

- Maven test reports

Summary

[Summary][Package List][Test Cases]

Tests	Errors	Failures	Skipped	Success Rate	Time
114	0	0	0	100%	17.617

Note: failures are anticipated and checked for with assertions while errors are unanticipated.

Package List

[Summary][Package List][Test Cases]

Package	Tests	Errors	Failures	Skipped	Success Rate	Time
nz.govt.ird.gst.dao.impl	29	0	0	0	100%	2.262
nz.govt.ird.util	16	0	0	0	100%	7.861
nz.govt.ird.gst.web	28	0	0	0	100%	4.463
nz.govt.ird.gst.web.propertyeditors	2	0	0	0		
nz.govt.ird.gst.web.form	3	0	0	0		
nz.govt.ird.gst.web.validation	5	0	0	0		
nz.govt.ird.gst.domain	7	0	0	0		
nz.govt.ird.util.database	2	0	0	0		
nz.govt.ird.gst.service.impl	18	0	0	0		
nz.govt.ird.gst.backend.impl.mock	4	0	0	0		

Note: package statistics are not computed recursively, they only sum up all of its test

[nz.govt.ird.gst.dao.impl](#)

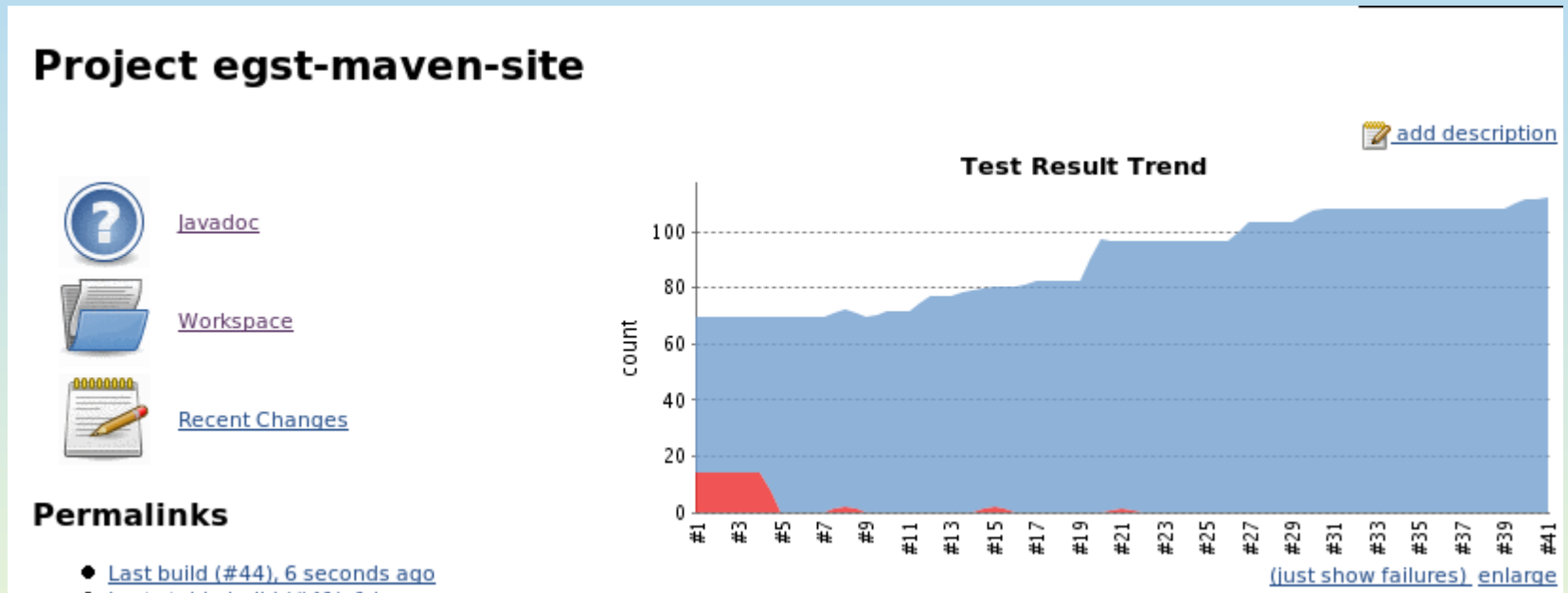
	Class	Tests	Errors	Failures	Skipped
	ClientDetailsDAOImplTest	3	0	0	0
	GstReturnDAOImplTest	5	0	0	0
	FilingPreferenceDAOImplIntegrationTest	5	0	0	0
	AdvancedFilingPreferenceDAOImplTest	1	0	0	0

PlatformManagerTest

	testSetEnvironmentWithSystemProperty	0.002
	testExtendedProdValue	0.003
	testDevValue	0.004
	testDefaultDevValue	0.002
	testIntegrationValue	0.003
	testTestValue	0.003
	testUATValue	0.003
	testInvalidValue	0.004

Test result trends

- Hudson dashboard



Test Coverage

- Test Coverage:
 - Are unit tests being written for all classes?
 - How much code is really executed by the unit tests?
 - Un-executed code will contain bugs

- Tool used:
 - Cobertura



```
public Object getJNDILookupDtbRef()
{
    return mJNDILookupDtbRef;
}

/**
 * Returns the EJB JNDI object reference for the Dtb
 * @return Object EJB JNDI object reference
 */
public Object getJNDILookupGlsRef()
{
    return mJNDILookupGlsRef;
}

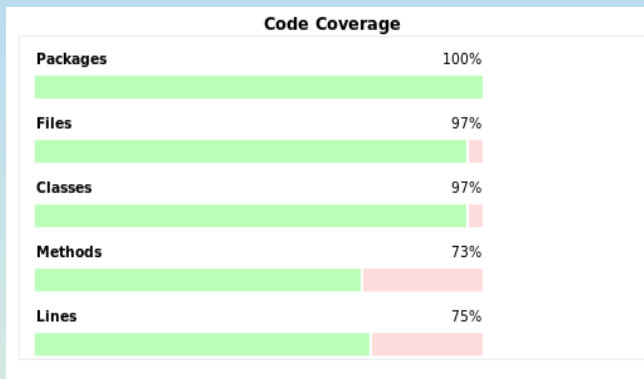
/**
 * Sets the EJB JNDI object reference to the Dtb ReferenceBean
 * @param Object EJB JNDI object reference
 */
public void setJNDILookupDtbRef(Object pJNDILookupRef)
{
    mJNDILookupDtbRef = pJNDILookupRef;
}

/**
 * Sets the EJB JNDI object reference to the Gls Session Bean
 * @param Object EJB JNDI object reference
 */
public void setJNDILookupGlsRef(Object pJNDILookupRef)
{
    mJNDILookupGlsRef = pJNDILookupRef;
}

/**
 * Overrides the default setting of the Log4j logger for this class.
 * @param Logger
 */
public void setLogger(Logger pLogger) { mLogger = pLogger; }
```


Test Coverage

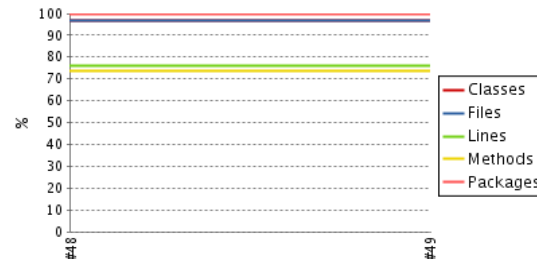
- Hudson Dashboard



Code Coverage

Cobertura Coverage Report

Trend



Project Coverage Summary

Name	Packages	Files	Classes	Methods	Lines
Cobertura Coverage Report	100% (13/13)	97% (60/62)	97% (64/66)	74% (330/448)	76% (957/1257)

Coverage Breakdown by Package

Name	Files	Classes	Methods	Lines
nz.govt.ird.gst.service	N/A	N/A	N/A	N/A
nz.govt.ird.gst.backend.impl	50% (1/2)	50% (1/2)	25% (2/8)	22% (2/9)
nz.govt.ird.gst.web.webflow.listeners	100% (1/1)	100% (1/1)	75% (6/8)	74% (43/58)
nz.govt.ird.gst.web.webflow.listeners	100% (1/1)	100% (1/1)	75% (6/8)	74% (43/58)

Types of tools

- Categories of tools
 - Build Process tools
 - Version Control tools
 - Continuous Integration tools
 - Quality Metrics and code audit tools
 - Unit Testing and Test Coverage tools
 - **Integration, Functional, Load and Performance Testing tools**
 - Technical Documentation tools

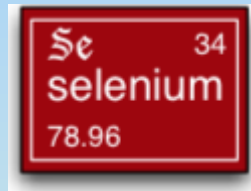


Integration, Functional, UI and Perf Tests

- Cover all your bases
 - DBUnit - *database testing*
 - Spring testing framework (Spring MVC) or StrutsTestCase
 - Eclipse Profiling (TPTP) - *profiling*
 - JMeter – *load and performance testing*
 - SoapUI – *web service testing*
 - Selenium – *web interface testing*
 - FEST – *Swing applications*
 - ...

User Interface Testing

- Automatic UI testing with Selenium:



enter-gst-only.html - Selenium IDE

File Edit Options Help

Base URL

Run Walk Step

Command	Target	Value
open	/egst/welcome.do	
waitForPageToLoad	5000	
verifyTextPresent	File goods and services (GST) return	
clickAndWait	//input[@value='Continue']	
waitForPageToLoad	5000	
verifyTextPresent	End date of GST period	
verifyValue	//input[@value='Continue']	Continue
clickAndWait	//input[@value='Continue']	
waitForPageToLoad	5000	
verifyTextPresent	Total sales and income	
verifyTextPresent	Zero-rated supplies included in the total sales	
verifyTextPresent	Adjustments	
verifyTextPresent	Total purchases and expenses	

Command: open

Target: /egst/welcome.do

Value:

Test Suite

- Logout
- Update Filing Preferences
- Side menu (user not logged on)
- GST-Only

enter-gst-only

Command	Target	Value
open	/egst/welcome.do	
waitForPageToLoad	5000	
verifyTextPresent	File goods and services (GST) return	
clickAndWait	//input[@value='Continue']	
waitForPageToLoad	5000	
verifyTextPresent	End date of GST period	
verifyValue	//input[@value='Continue']	Continue
clickAndWait	//input[@value='Continue']	
waitForPageToLoad	5000	
verifyTextPresent	Total sales and income	
verifyTextPresent	Zero-rated supplies included in the total sales	
verifyTextPresent	Adjustments	
verifyTextPresent	Total purchases and expenses	
verifyTextPresent	Credit adjustments	

Selenium TestRunner

Execute Tests

Fast Slow

Highlight elements

Elapsed: 00:33

Tests	Commands
3 run	33 passed
0 failed	0 failed
	0 incomplete

Tools

View DOM Show Log

Inland Revenue
Te Tari Taake

You are here: Online Services

File goods and services (GST) return

10000125 MERVIN GRANITE (MOCK CURRENT PROFILE)

- Inland Revenue home
- Online Services home
- IGST home
- Modify Filing Preferences
- View History
- Maintain your contact details
- Change your

IRD number: 10000125

Name: Mervin Granite (mock current profile)

Accounting Basis: Invoice Filing frequency: Monthly

Please select one of the following options:

- Select GST filing preference
- View History
- Add a return

Help

Set GST filing preferences

View history

Regression Testing

- All tests automatically run before each build:
 - Unit tests
 - Integration tests
 - User Interface tests

Types of tools

- Categories of tools
 - Build Process tools
 - Version Control tools
 - Continuous Integration tools
 - Quality Metrics and code audit tools
 - Unit Testing and Test Coverage tools
 - Integration, Functional, Load and Performance Testing tools
 - **Technical Documentation tools**



Technical Documentation

- Human-written documentation:
 - (potentially) highest possible quality
- BUT
 - Often incomplete
 - Hard to keep up-to-date

Technical Documentation

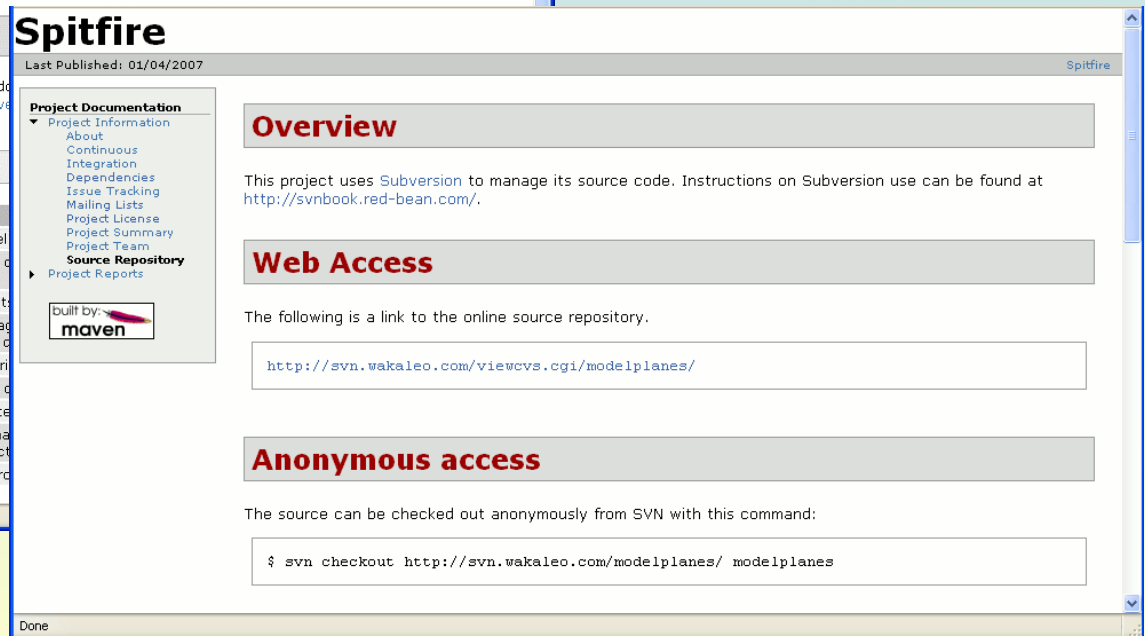
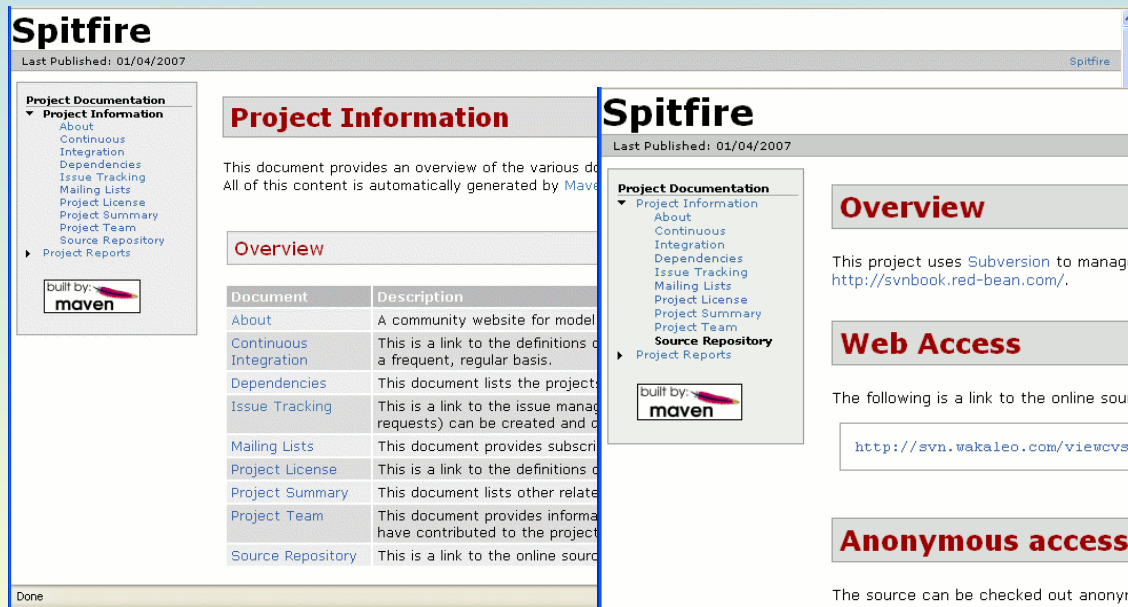
- Automatically-generated documentation:
 - Complete
 - Always up-to-date
 - Cheap to produce
- BUT
 - Lacks “higher vision”

Technical Documentation

- Many tools available:
 - Maven web site – *lots of reports and project info*
 - SchemaSpy – *database schemas*
 - Doxygen – *UML documentation*
 - UmlGraph – *UML-enabled Javadoc*

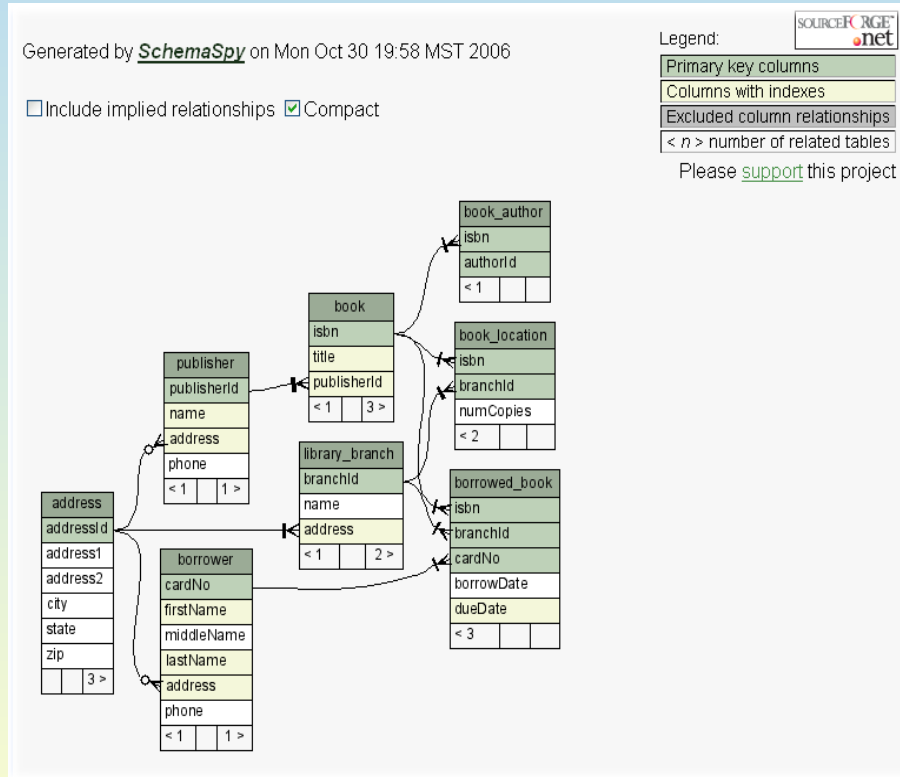
Technical Documentation

- Maven web site
 - Rich reporting capabilities
 - General project information
 - Detailed, customizable reports



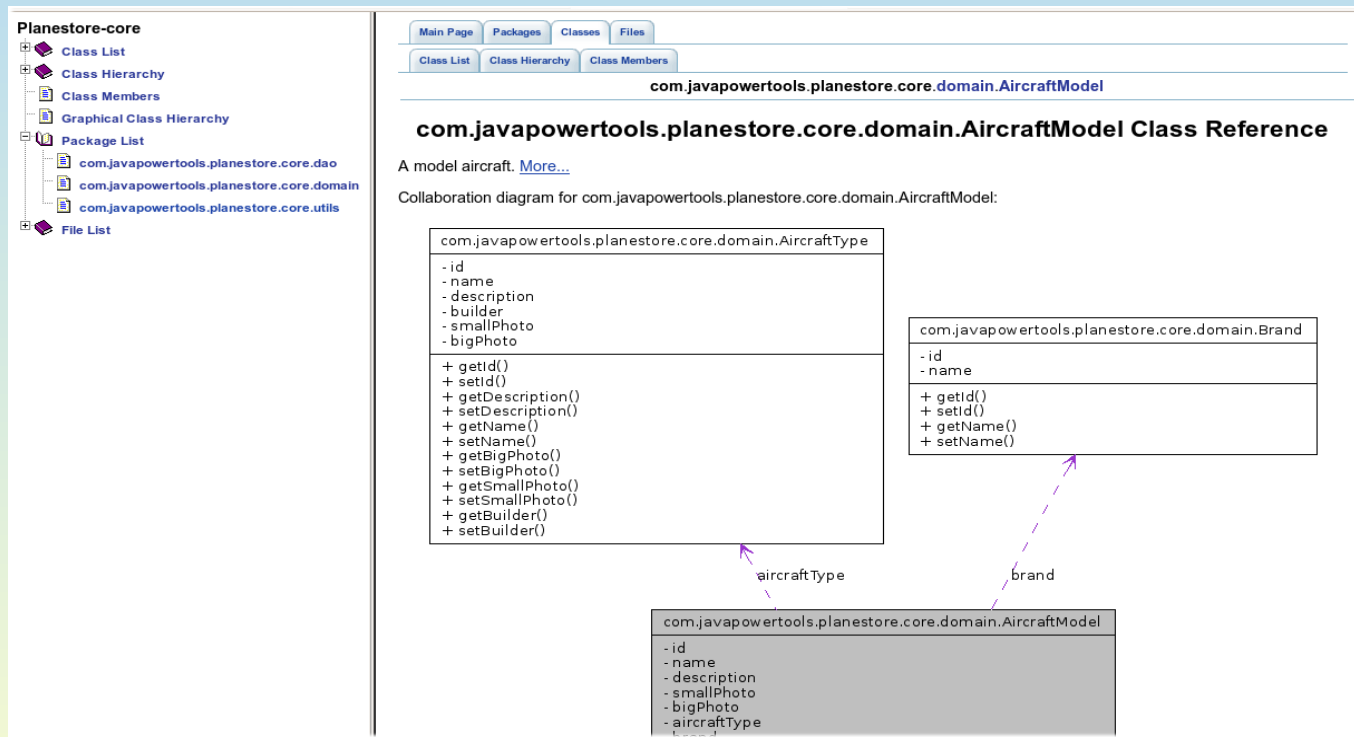
Technical Documentation

- SchemaSpy
 - Database schemas



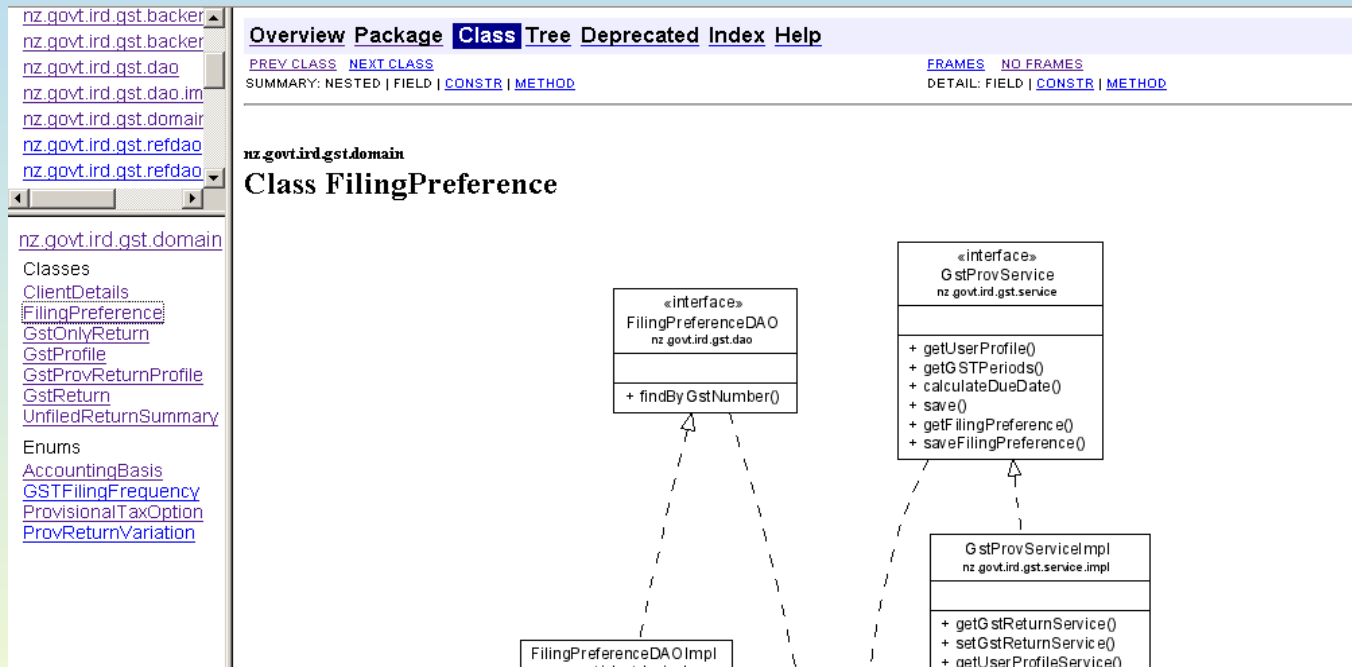
Technical Documentation

- DOxygen
 - UML-embedded documentation



Technical Documentation

- UMLGraph
 - Javadoc with UML diagrams



Technical Documentation

- Also some human-written documentation – project Wiki
 - Architecture vision
 - High-level design
 - Collaborative

O2C2 Edit Attach Printable O2C2.GstDomainModel r1.3 - 1

OSWiki
About O2C2

O2C2 Section
Section Home
Topic List
Detailed Index
Changes
Search
Users
Usage Stats

O2C2
High Level Information
About O2C2
Products
Services
Research
Design
Developer Resources
Testing Resources
Topics Under Discussion
Contact us

O2C2 team area
O2C2 Workspace
System Issues & Diagnosis

OSWiki Sections
Filmpart
IAMS
Inventory
Main
O2C2
OpenSystems
Operations
Sandbox
TWiki

O2C2 > ProductsO2C2WorkOn > InternetGSTProvTax > GstDomainModel

iGST Domain Model

DesignDocument - Design Documentation - (Current)
Overview of the iGST domain model

High-level domain model

The chief elements of the iGST domain model are illustrated here:

```
classDiagram
    class RegistrationProfile {
        +String registrationNumber
        +String primeNumber
        +String primeLocation
    }
    class ClientDetails {
        +String gstRegistrationNumber
        +String primeNumber
        +String primeLocation
    }
    class GstProfile {
        +GstFilingFrequency filingFrequency
        +AccountingBasis accountingBasis
    }
    class GstReturnProfile {
        +Date periodEndDate
        +Date dueDate
        +ProvReturnVariation returnVariationType
        +boolean voluntaryProvPeriod
        +int ratio
        +ProvisionalTaxOption provisionalTaxOption
    }
    class GstOnlyReturn
    class GstDerivedProdReturn
    class GstProvRatio
    class GstProvStandardOrEstimate
    class GstReturnPeriod {
        +Date periodEndDate
        +String gstNumber
        +ReturnStatus status
    }

    RegistrationProfile "1" -- "1" ClientDetails
    RegistrationProfile "1" -- "1" GstProfile
    ClientDetails "1" -- "1" GstProfile
    GstReturnProfile "1" -- "1" GstOnlyReturn
    GstReturnProfile "1" -- "1" GstDerivedProdReturn
    GstReturnProfile "1" -- "1" GstProvRatio
    GstReturnProfile "1" -- "1" GstProvStandardOrEstimate
    GstReturnProfile "1" -- "1" GstReturnPeriod
```

Application architecture

The iGST project uses a fairly standard MVC architecture approach, using the Spring-MVC framework and Hibernate. The target architecture involves a portal-enabled front-end using Spring-MVC (more precisely, Spring Portlet MVC), and Spring Webflow (with the [PortletFlowController](#)) and a business services layer implemented using Spring and Hibernate, backed by an Oracle database. The business layer communicates with FIRST via EAI:



The application architecture is described in detail in the [GstApplicationArchitecture](#) page.

Domain model

The domain model plays a central part in the iGST project, and has been elaborated in a series of workshops. The domain model is designed as an object-oriented class model, and implemented using plain Java classes (POJOs). Database persistence is implemented using Hibernate.

Overview

A Standard Build Process



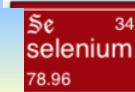
Dependency Management



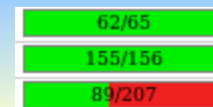
Continuous Integration



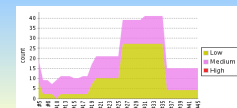
Continuous Testing



Continuous Quality



Continuous Reporting



Continuous Documentation



Questions?