



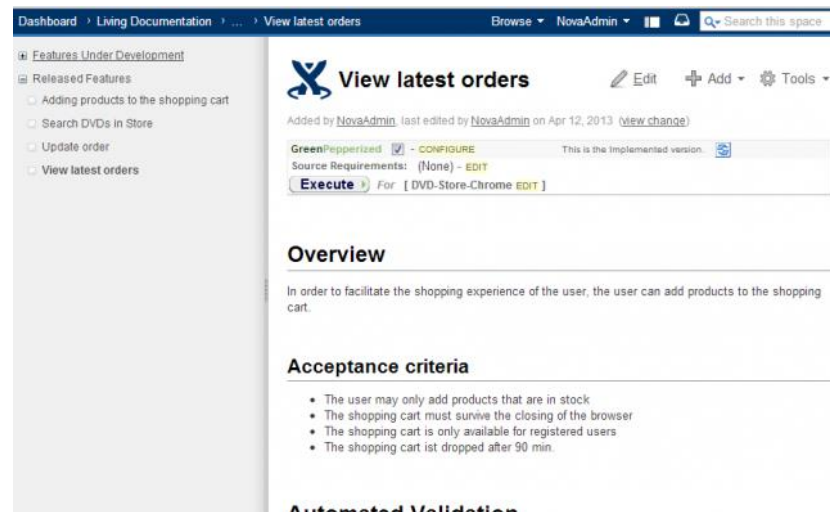
Automatisation des tests d'acceptation



- **Les spécifications exécutable**
- **Découvrir FitNesse**
 - Introduction à Fitnesse
 - Principe et architecture de Fitnesse
 - De FIT à **SLIM**
 - Installer Fitnesse
 - Comprendre les fixtures
- **Ecrire des tests d'acceptation**
- **Implémentation des tests**
 - Coder les fixtures
 - Exécution des tests
 - Debug
- **Les plugins**
- **Intégrer Selenium**
 - Gestion des librairies Selenium
 - Ecriture des tests Fixtures Driver (pour piloter les tests Selenium)
 - Construire son langage pour automatiser les tests Web
 - Plugin : BrowserTest

Chap. 1: Spécifications exécutables

- Rappel



1. Approche Test First

- Tester de la valeur métier portée dans les specs
 - Les tests doivent permettre de représenter les comportements des fonctionnalités attendues afin de tester ce qui est nécessaire et qui apporte de la valeur métier.
 - La qualité est de la responsabilité de tous les acteurs du projet. Ils doivent intervenir pour garantir que chaque couche de construction du produit répond aux attentes et apporte cette valeur.

1. Approche Test First

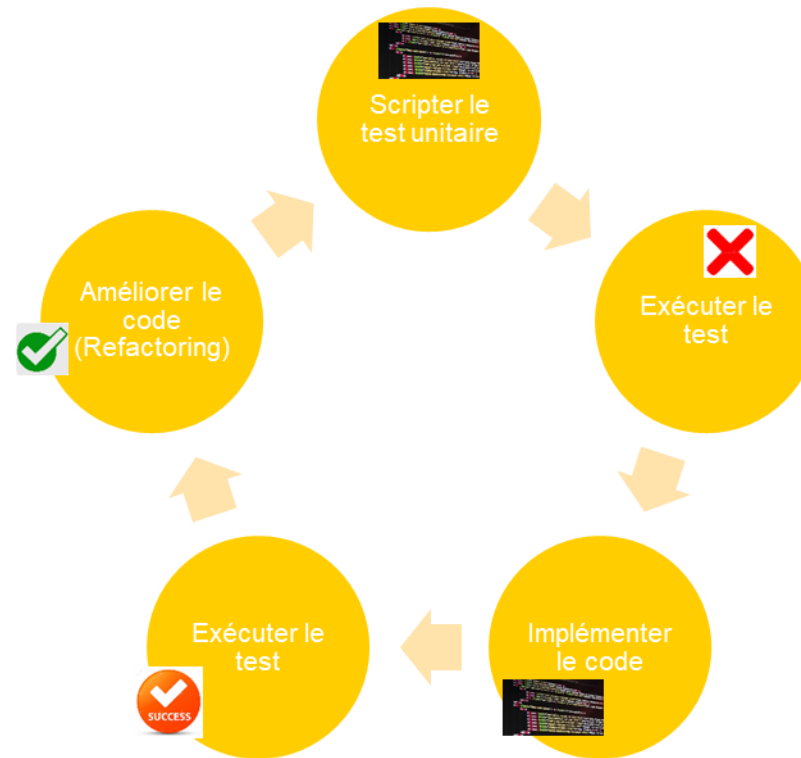
- TDD (Test Driven Development)
 - Réalisation en amont des Tests Unitaires par les développeurs
- BDD (Behavior Driven Development)
 - Réalisation en amont des tests basés sur le comportement du système
- ATDD (Acceptance Test Driven Development)
 - Réalisation en amont des tests d'acceptance

1. Ecrire les tests

2. Développer

3. Exécuter les tests

1. Approche Test First



1. Living Documentation

- Les spécifications sont exécutables
 - Permet de tester le système avec la vision réelle attendue sur les différents niveaux de test

Chap. 2: Découvrir Fitnessse

- Introduction
- Principe
- Architecture
- Premiers pas

2.1 Introduction



Serveur Web

- Application serveur autonome écrit en Java
- Déployable avec les outils standards de la stack Java: Maven, Junit



Wiki

- Solution collaborative de gestion des spécifications logicielles
- Mise à disposition d'outil de formatage pour la documentation des tests

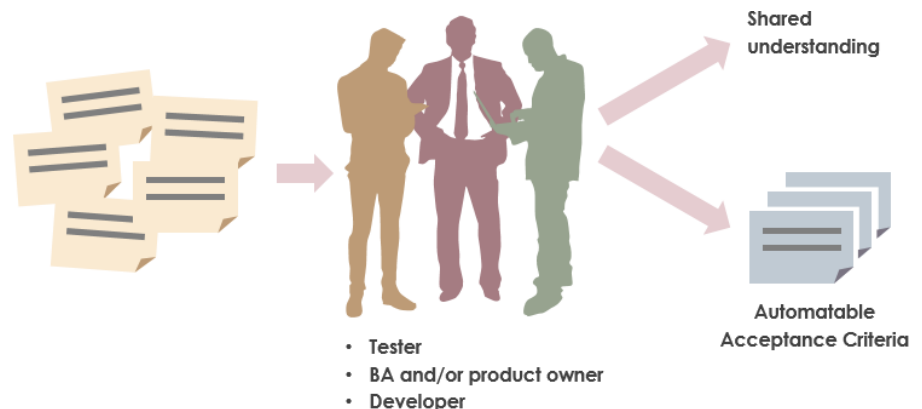


Outil de test

- Outil de conception des tests
- Framework d'automatisation des tests d'acceptation et des tests unitaires

2.1 Introduction: Histoire

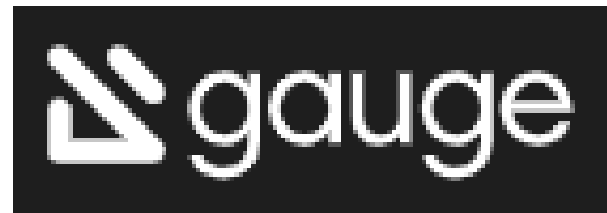
- 2001: FIT (Framework for Integrated Testing)
 - Créé par Ward Cunningham pour mettre l'accent sur la collaboration et la communication
 - Objectif: Accélérer la fiabilité des développements en permettant au développeur de tester le code avec la vision métier



2.1 Introduction: Solutions équivalentes

- Fitnesse est un des pionniers dans l'approche des spécifications exécutables

cucumber



Concordion

GreenPepper



2.2 Principes: Spécification des tests

- Fitnessse complète l'architecture FIT avec une approche boîte noire
 - Tests d'acceptation
 - Test de non régression

Dynamic Decision Table

ddt: dynamic decision table									
# description	1p	5p	10p	25p	50p	Rs.1	total paise?	Rs. total?	
addition of Coin values	5	3	6	0	0	0	80	0.80	
saving total value in a symbol	37	1	1	0	2	23	\$totalPaise<-[2452]	24.52	
Using the total Paise value Stored before	\$totalPaise->[2452]						0	0	0
						10	3452	34.52≈=34.5	

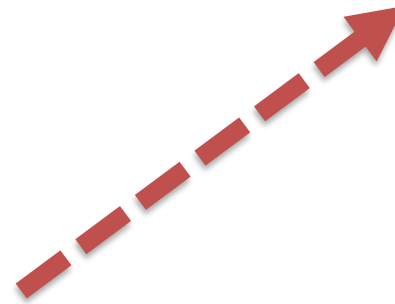
- Fitnessse est un outil de gestion des tests
 - Permet la documentation des tests
 - permet de spécifier les tests à l'aide des techniques classiques de conception des tests
 - **Structure en tableaux**

2.2 Principes: Outil de test

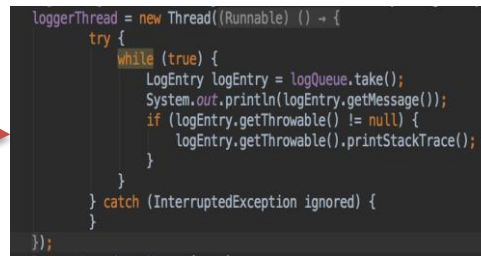
- Approche ATDD
 - Conception des tests
 - Exécution des tests



Code applicatif

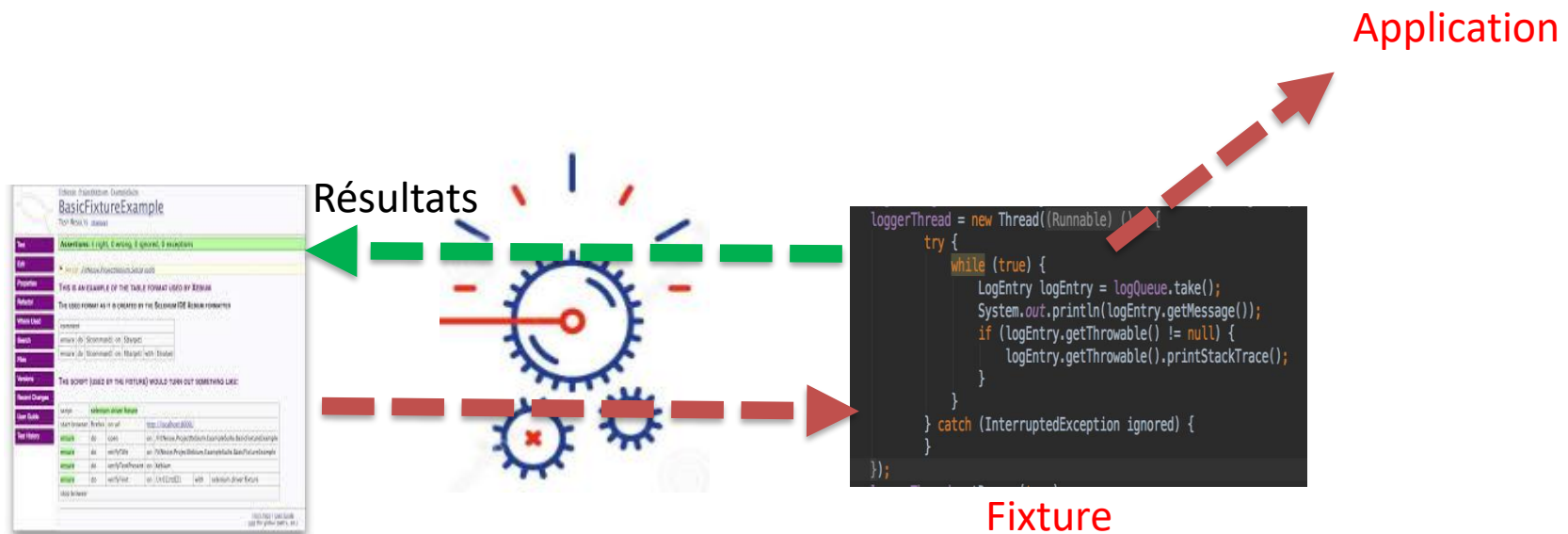


Code de test



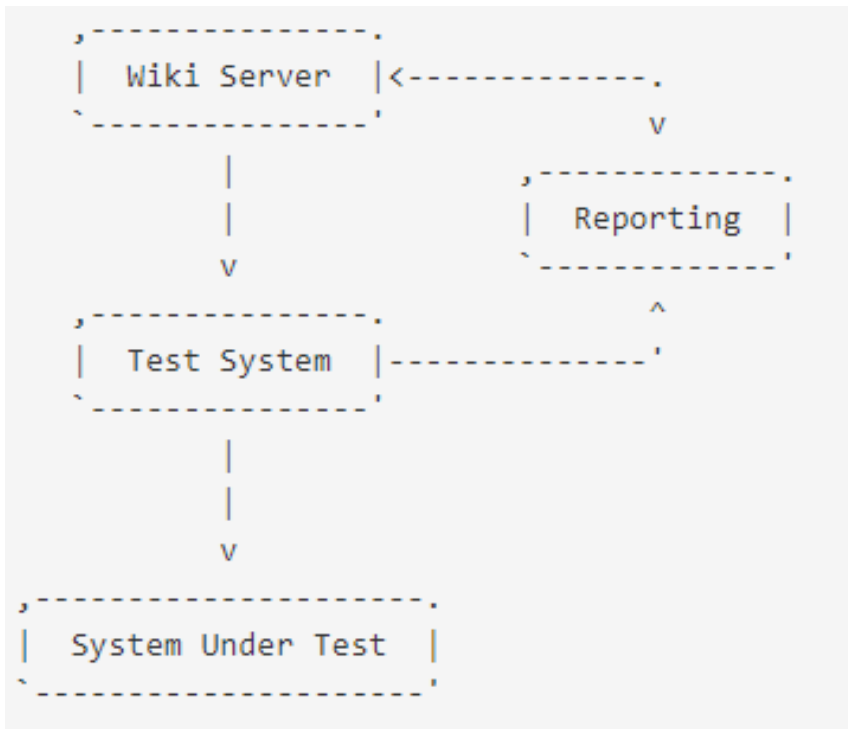
2.3 Architecture: Le moteur

- Framework de test

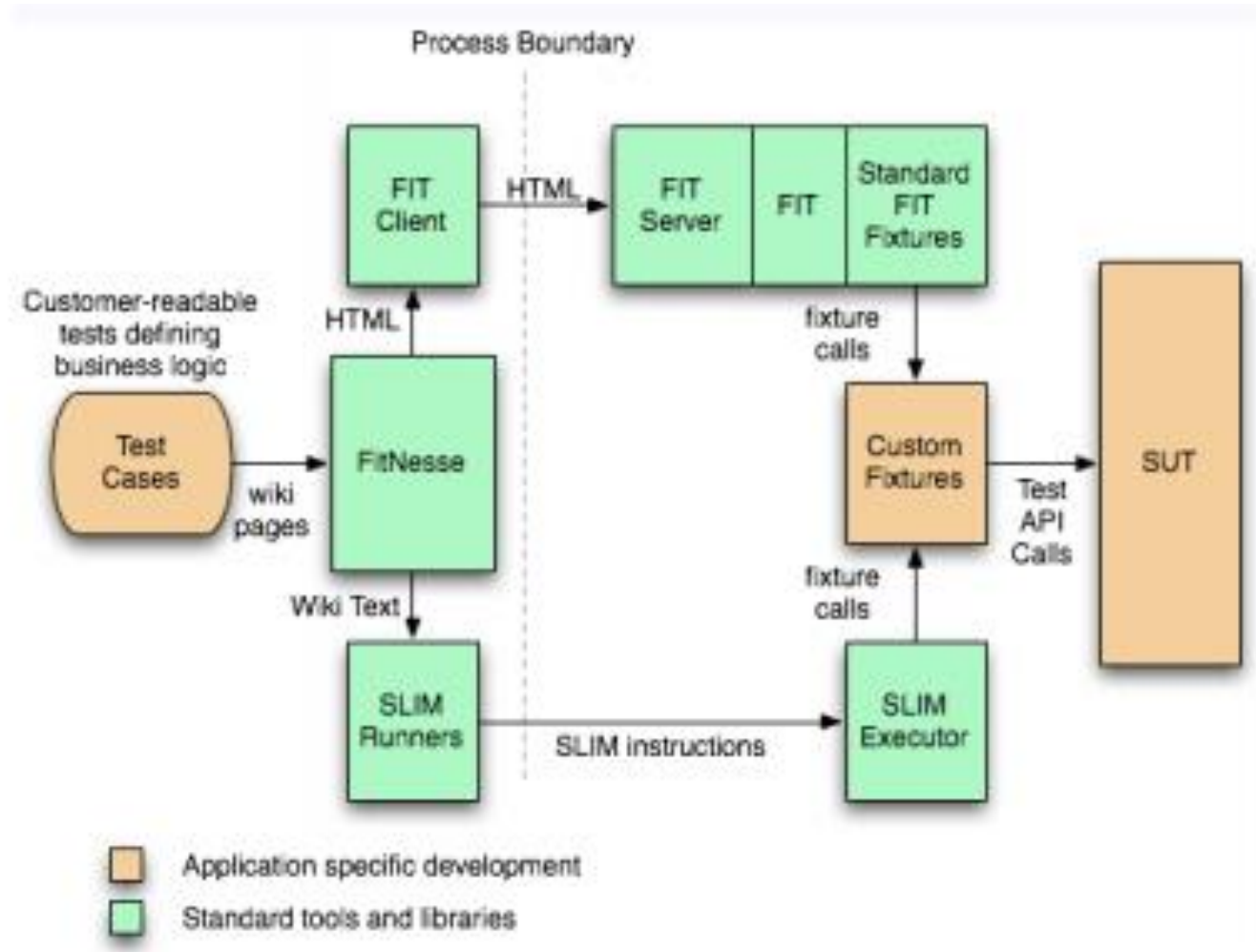


2.3 Architecture: les briques

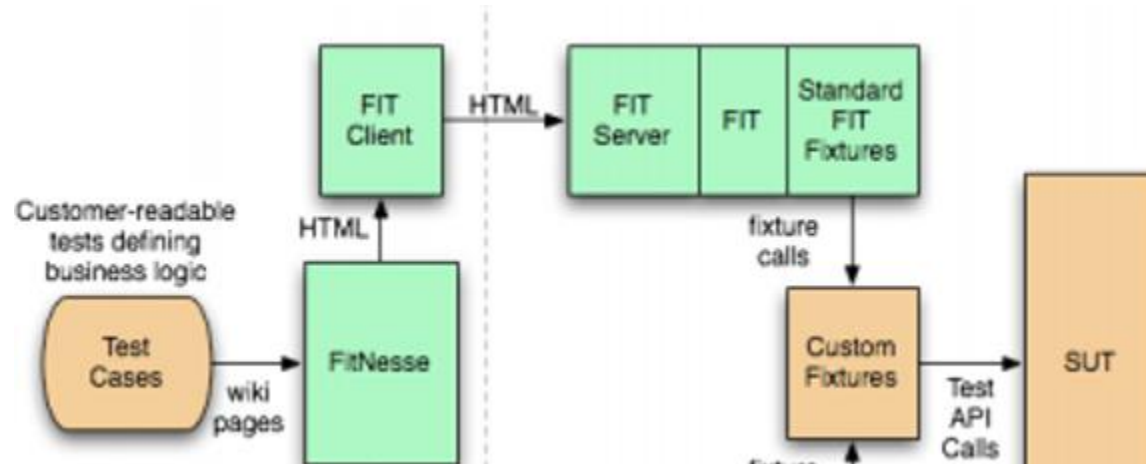
- Composants



2.3 Architecture: les composants

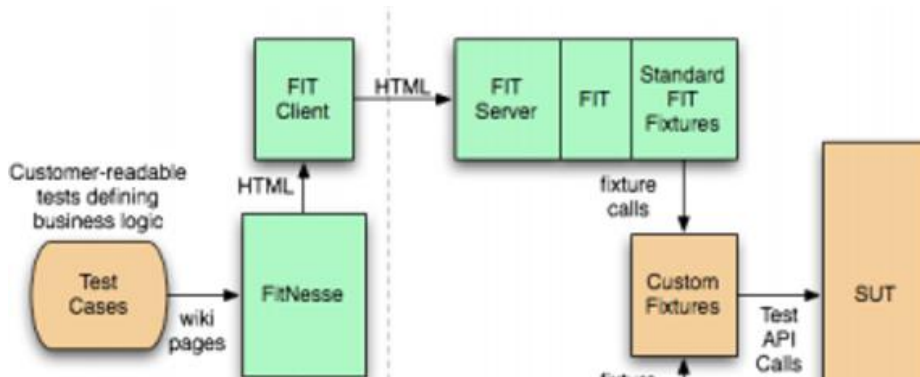


2.3 Test engine: FIT



- Ancien système (base du framework)
- Protocole client server pour analyser les pages

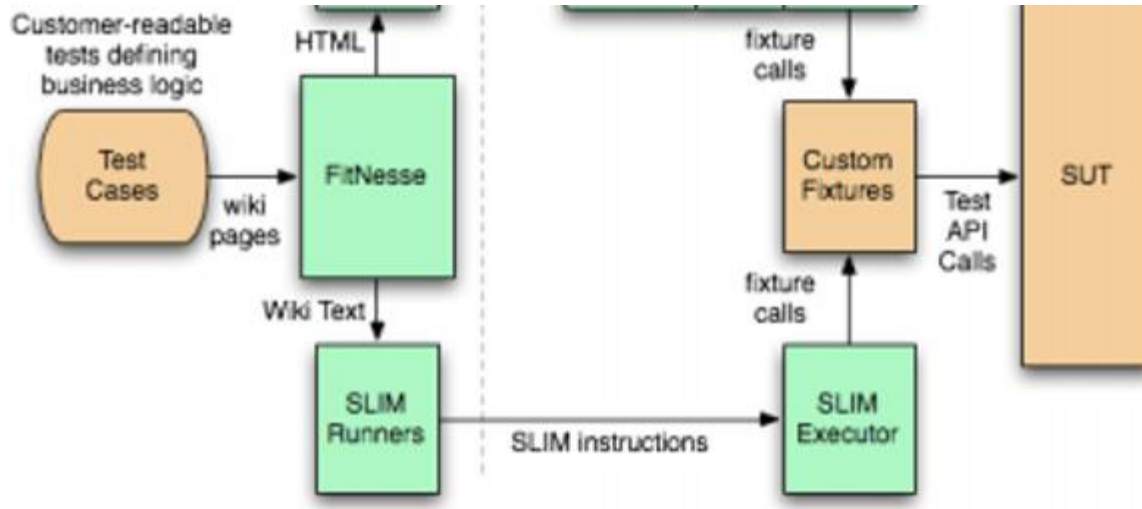
2.3 Test engine: FIT



- Gestion des fixtures plus lourdes
 - Héritage nécessaire entre les Custom Fixtures et les classes Standard

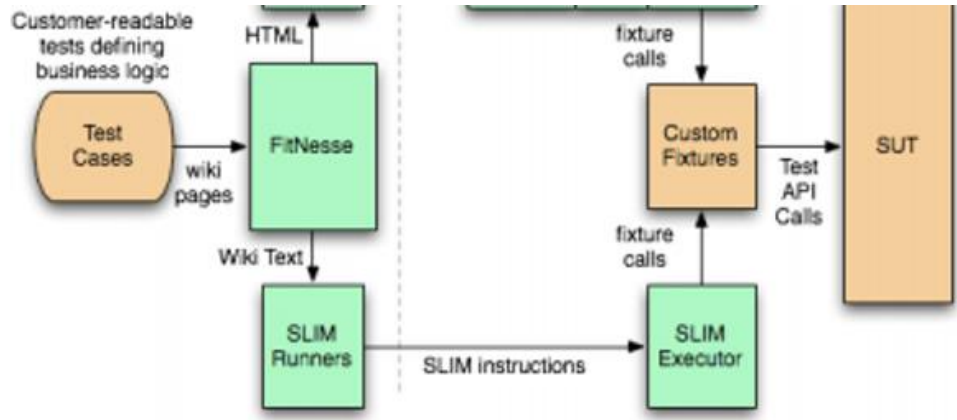
```
public class ColumnFixtureTest extends ColumnFixture {  
    public String firstPart;  
    public String secondPart;  
    private int length;  
    public String together(){  
        length=firstPart.length()+secondPart.length();  
        return firstPart+ ", "+secondPart;  
    }  
    public int totalLength(){  
        return length;  
    }  
}
```

2.3 Test engine: SLIM



- Simple List Invocation Method
- Protocole plus léger
 - L'analyse des pages et la génération des résultats sont gérées par le server Wiki

2.3 Test engine: SLIM



- Gestion des fixtures
 - Simple POJO
 - Héritage est toujours possible mais pas obligatoire

should I buy milk			
cash in wallet	credit card	pints of milk remaining	go to store?
0	no	0	no
10	no	0	yes
0	yes	0	yes

```
public class ShouldIBuyMilk {
    private int dollars;
    private int pints;
    private boolean creditCard;

    public void setCashInWallet(int dollars) {
        this.dollars = dollars;
    }

    public void setPintsOfMilkRemaining(int pints) {
        this.pints = pints;
    }

    public void setCreditCard(String valid) {
        creditCard = "yes".equals(valid);
    }
}
```

2.5 Démarrons: Installation de Fitnesse

- Prérequis:
 - JDK Java
 - Binaire Fitnesse:
 - <http://fitnesse.org/FitNesseDownload>
- Nécessaire
 - Un IDE Java

- Lancer une invite de commande

java -jar fitnesse-standalone.jar

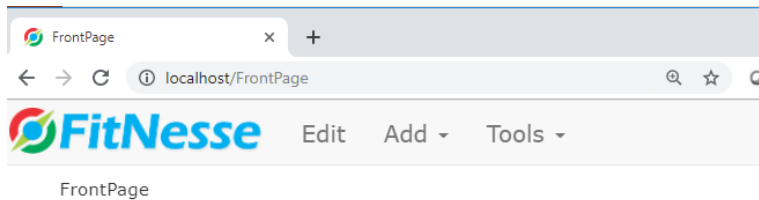
```
INFO: No configuration file found (D:\dev\fitnesse\standard\plugins
Bootstrapping FitNesse, the fully integrated standalone wiki and acc
root page: fitnesse.wiki.fs.WikiFilePage: FitNesseRoot
logger: none
authenticator: fitnesse.authentication.PromiscuousAuthenticator
page factory: fitnesse.html.template.PageFactory
page theme: bootstrap
Unpacking new version of FitNesse resources. Please be patient...
*****
Files have been updated to a new version.
Please read the release notes on
http://localhost:80/FitNesse.ReleaseNotes
to find out about the new features and fixes.
*****
Starting FitNesse on port: 80
```

Autre port:

java -jar fitnesse-standalone.jar -p 8080

2.5 Démarrons: Installation de Fitnesse

- Fitnesse WIKI:
 - Lancer votre navigateur: `http://localhost`



Welcome to **FitNesse**!

The fully integrated stand-alone acceptance test framework

To add your first "page", click the [Edit](#) button and add a [WikiWord](#) to the page.

To Learn More...	
User Guide	Answer the rest of your questions here.
A Two-Minute Example	A brief example. Read this one next.
Acceptance Tests	FitNesse's suite of Acceptance Tests
Release Notes	Find out about FitNesse's new features

Release v20190716

[Front Page](#) | [User Guide](#) | [root](#) (for global !path's, etc.) | Press '?' for keyboard shortcuts

2.5 Démarrons: Exécutons un test

- Aller sur la page:

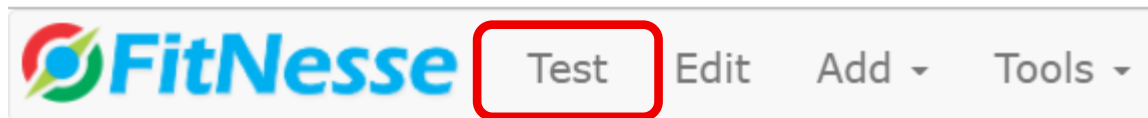
[A Two-Minute Example](#)

A brief example. Read this one next.

eg.Division		
numerator	denominator	quotient?
10	2	5.0
12.6	3	4.2
22	7	≈ 3.14
9	3	< 5
11	2	$4 < _ < 6$
100	4	33

Tableau de test

- Cliquer sur le bouton



2.5 Démarrons: Résultat

- Résumé des assertions de la page

✖ **Test Pages:** 0 right, 0 wrong, 0 ignored, 0 exceptions **Assertions:** 5 right, 1 wrong, 0 ignored, 0 exceptions (0,559 seconds)

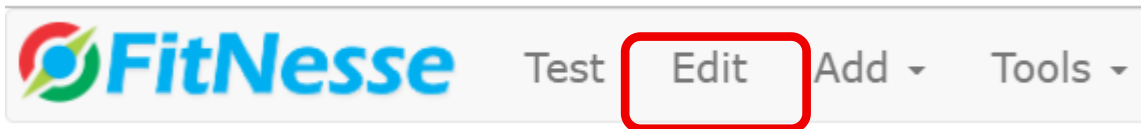
- Résultat du tableau de test

eg.Division		
numerator	denominator	quotient?
10	2	5.0
12.6	3	4.2
22	7	3.142857142857143~3.14
9	3	3.0<5
11	2	4<5.5<6
100	4	[25.0] expected [33]



2.5 Démarrons: Structure

- Editons la page



```
1 |!***< Hidden
2
3 |!define TEST_SYSTEM {slim}
4 |*!
5 |!1 An Example !-FitNesse-! Test
6 |If you were testing the division function of a calculator application, you might like to see some exa
7
8 |In !-FitNesse-!, tests are expressed as tables of '''input''' data and '''expected output''' data. He
9
10 |
11 |eg.Division |
12 |numerator | denominator | quotient? |
13 |10 | 2 | 5.0 |
14 |12.6 | 3 | 4.2 |
15 |22 | 7 | ~=3.14 |
16 |9 | 3 | <5 |
17 |11 | 2 | 4<=6 |
18 |100 | 4 | 33 |
19
20 |This style of !-FitNesse-! test table is called a [[Decision Table][<UserGuide.WritingAcceptanceTest:
21
22 |!3 Running our test table: Click the Test button
23 |If you are reading this page on a local copy of !-FitNesse-!, run this test table. See the '''Test'''
24
25 |!*** Now, if you're reading this on the web site, ...
```



2.5 Démarrons: Résultat

- Après modification

[FitNesse](#) / [UserGuide](#) / [TwoMinuteExample](#)

✓ **Test Pages:** 0 right, 0 wrong, 0 ignored, 0 exceptions **Assertion**

An Example FitNesse Test

If you were testing the division function of a calculator application, you (You might be hoping for a 5!)

In FitNesse, tests are expressed as tables of **input** data and **expected**

eg.Division		
numerator	denominator	quotient?
10	2	5.0
12.6	3	4.2
22	7	3.142857142857143~3.14
9	3	3.0<5
11	2	4<5.5<6
100	4	25.0

2.5 Démarrons: La fixture

eg.Division

numerator	denominator	quotient?
-----------	-------------	-----------

```
public class Division {  
    private double numerator, denominator;  
  
    public void setNumerator(double numerator) {  
        this.numerator = numerator;  
    }  
  
    public void setDenominator(double denominator) {  
        this.denominator = denominator;  
    }  
  
    public double quotient() {  
        return numerator/denominator;  
    }  
}
```



2.5 Démarrons: TP

- Nous voulons ajouter une nouvelle table de test dans la page actuelle.
- Cette table permet de tester la fonction `Math.max`

org.fitnessse.demo.fixtures.MathFixture		
first number	second number	the greater is?
10	2	10
2	10	10
2	2	2
152	200	200

2.5 Démarrons: TP

- Editons la table
- Créons un nouveau projet maven:
org.fitnessse.demo.fixtures
- Créons une nouvelle classe MathFixture
- Créons le jar
- Déposons le jar dans la librairie Fitnessse
- Exécutons le test

2.5 Démarrons: TP

- La classe

```
package org.fitnessse.demo.fixtures;

class MathFixture {

    Integer nb1;
    Integer nb2;

    public void setFirstNumber(Integer n){
        this.nb1=n;
    }

    public void setSecondNumber(Integer n){
        this.nb2=n;
    }

    public Integer theGreaterIs(){
        return Math.max(nb1,nb2);
    }
}
```



2.5 Démarrons: TP

- Le CamelCase fait le lien

```
package org.fitnessse.demo.fixtures;
```

```
org.fitnessse.demo.fixtures.MathFixture
```

```
class MathFixture {
```

```
    Integer nb1;  
    Integer nb2;
```

```
    public void setFirstNumber(Integer n){ first number  
        this.nb1=n;  
    }
```

```
    public void setSecondNumber(Integer n){ second number  
        this.nb2=n;  
    }
```

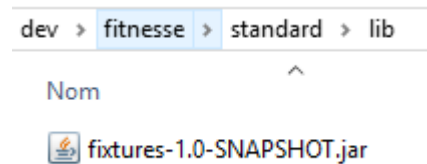
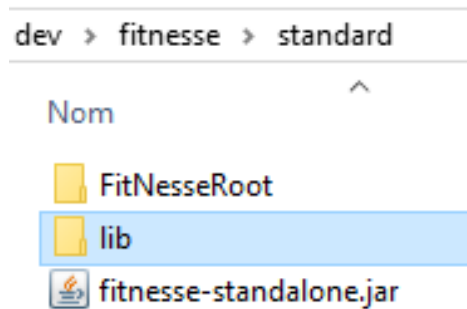
```
    public Integer theGreaterIs(){ the greater is?  
        return Math.max(nb1,nb2);  
    }
```

```
}
```

Résultat attendu

2.5 Démarrons: TP

- Commande pour le packaging: *mvn clean package*
- Déposons le jar dans un nouveau dossier lib à la racine de l'application Fitnessse



2.5 Démarrons: TP

- Exécutons le test

org.fitnessse.demo.fixtures.MathFixture		
first number	second number	the greater is?
10	2	10
2	10	10
2	2	2
152	200	200

2.5 Démarrons: TP

- Travaillons en FIT
- Editons la page avec les information suivantes

```
!define TEST_SYSTEM {fit}  
* |
```

org.fitnessse.demo.fixtures.MathFitFixture		
first number	second number	the greater is?

2.5 Démarrons: TP

- Ajoutons la classe dans notre projet

```
package org.fitnessse.demo.fixtures;  
import fit.ColumnFixture;  
  
public class MathFitFixture extends ColumnFixture{  
    Integer firstNumber;  
    Integer secondNumber;  
  
    public Integer theGreaterIs() { return Math.max(firstNumber,secondNumber); }  
}
```

- Lançons le packaging

2.5 Démarrons: TP

- Exécutons le test

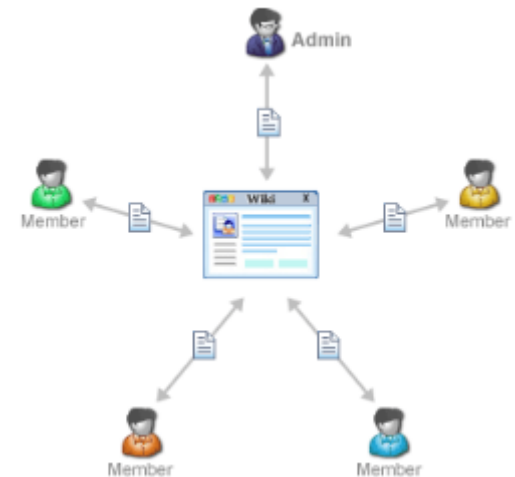
org.fitnessse.demo.fixtures.MathFitFixture		
first number	second number	the greater is?
10	2	10
2	10	10
2	2	2
152	200	200

Chap. 3: Ecrire les tests d'acceptation

- Gestion du Wiki
- Organisation des tests
- Les différents types de tableaux de tests
- Travaux Pratiques

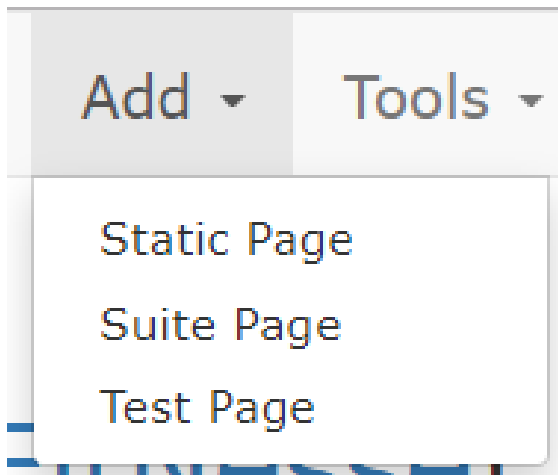
3.1: Un Wiki

- Fitnessse est d'abord un wiki
 - Même créateur (W. Cunningham)
- Gestion des contenus
- Gestion collaborative au sein d'une équipe
- Utilisation de balises pour créer les contenus (Markup)
- Gestion des liens avec des WikiWords
- Suivi de l'historique des pages



3.1: Un Wiki: Créer une nouvelle page

- Nommage des pages par Camel Case
- Ajout d'une page statique:
 - Une page non exécutable



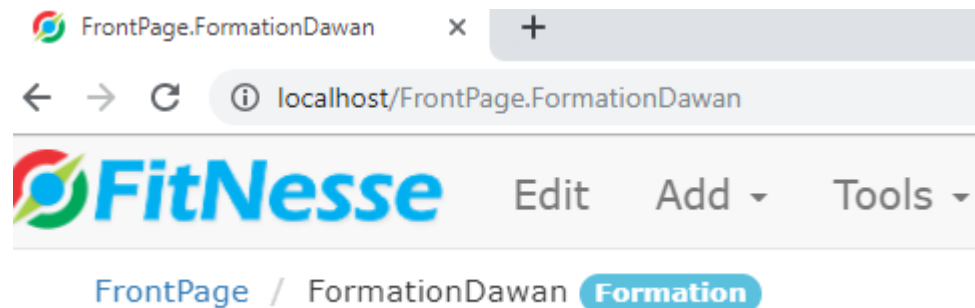
3.1: Un Wiki: Créer une nouvelle page

FrontPage

Page name: FormationDawan

Help text: Formation Fitnessse

Tags: Formation x



3.1: Un Wiki: Editer la page

- A l'aide du Rich Text Editor (Boîte à outil Word-like)

Normal **B** *I* U ~~ABC~~

Formation Tests d'acceptation automatisés avec [FitNesse](#)

Durée : 3 jours

Public : Développeurs, testeurs

Pré-requis : ~~Avoir suivi le stage "Java initiation" ou posséder les connaissances équivalentes~~

Objectifs :

- Maîtriser la rédaction des tests de recettes avec [FitNesse](#)
- Maîtriser l'automatisation des tests de recettes avec [FitNesse](#)

Sanction : Attestation de fin de stage mentionnant le résultat des acquis

Référence : [Page de présentation](#)

– *Découvrir Fitnesse*

Tests fonctionnels : intérêts, panorama des outils disponibles
Tests unitaires vs Tests fonctionnels

[FitNesse](#) : principe, architecture, apports
Moteurs de test : FIT, SLIM

3.1: Un Wiki: Editer la page

<http://fitnesse.org/FitNesse.UserGuide.FitNesseWiki.MarkupLanguageReference>

- A l'aide des balises

```
!1 Formation Tests d'acceptation automatisés avec FitNesse
'''Durée ''' : '3 jours'

'''Public ''' : Développeurs, testeurs

--Pré-requis : Avoir suivi le stage "Java initiation" ou posséder les connaissances équivalentes--

Objectifs :

* Maîtriser la rédaction des tests de recettes avec FitNesse
* Maîtriser l'automatisation des tests de recettes avec FitNesse

Sanction : Attestation de fin de stage mentionnant le résultat des acquis

Référence: [[Page de présentation]]\[http://www.dawan.fr/formations/tests/tests-d-acceptation-automatisees-avec-fitnesse\]

!*** Découvrir FitNesse

Tests fonctionnels : intérêts, panorama des outils disponibles
Tests unitaires vs Tests fonctionnels

FitNesse : principe, architecture, apports
Moteurs de test : FIT, SLIM

*!
!***> Ecrire des tests d'acceptation

Organisation des tests :

* Test Suites
* Pages spéciales
* Historique de test
* Patterns de tests d'acceptation
```

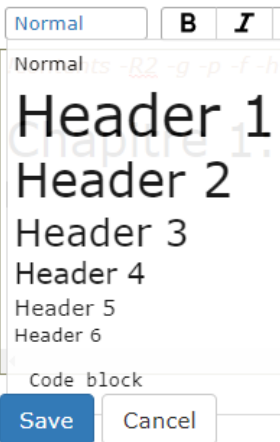


3.1: Un Wiki: Editer la page

- A l'aide du Rich Text Editor (Boîte à outil Word-like)



Entête de chapitre



Texte du lien

3.1: Un Wiki: Créer un lien vers la page

- Edition de la page pour ajouter un lien
- Rajouter la ligne à la fin du tableau

```
| [[La formation Dawan]] | ''Formation'' |
```

Diagram illustrating the structure of the link and the page access:

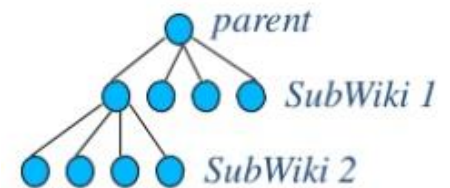
- Texte du lien**: [[La formation Dawan]]
- Accès à la page**: [.FrontPage.FormationDawan]]

To Learn More...	
User Guide	<i>Answer the rest of your questions here.</i>
A Two-Minute Example	<i>A brief example. Read this one next.</i>
Acceptance Tests	<i>FitNesse's suite of Acceptance Tests</i>
Release Notes	<i>Find out about FitNesse's new features</i>
La formation Dawan	<i>Formation</i>

3.1: Un Wiki: SubWiki



- Groupées les pages par projet dans un SubWiki
 - Hiérarchisation des pages de tout type
- Facilite l'organisation, la lisibilité et la maintenabilité du Wiki



.FrontPage.FormatonDawan.

3.1: Un Wiki: SubWiki

- Créer la page Parent
 - Cliquer sur **root**
 - Ajouter une page statique: Dolibarr
- Rajouter des pages
 - Cliquer sur Dolibarr
 - Ajouter une page Connexion
 - Ajouter une page Gestion du profil



root

Contents:

- [Dolibarr](#)

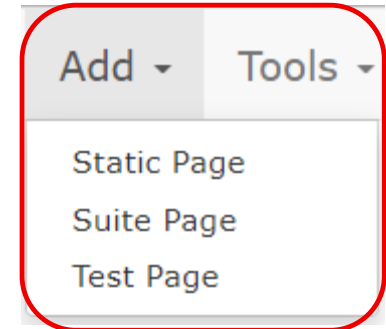
Dolibarr

Contents:

- [Connexion](#)
- [Gestion Du Profil](#)

3.1: Un Wiki: Type de page

- Static Page
 - Page du contenu du wiki
- Test Page
 - Page géré en tant que test
- Suite page
 - Suite de pages de test



Page dont le nom commence par Test => Test page
Page dont le nom commence par Suite => Suite page

3.1: Un Wiki: Propriété de page

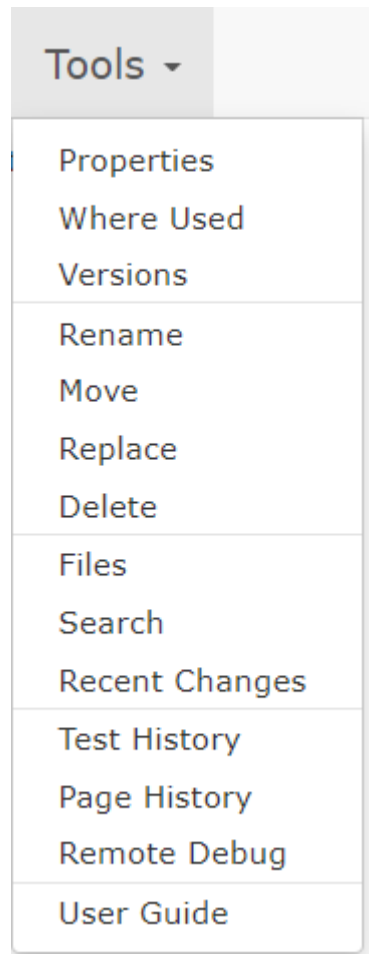
Page properties

Page type:	Actions:	Navigation:	Security:
<input type="radio"/> Static	<input checked="" type="checkbox"/> Edit	<input checked="" type="checkbox"/> RecentChanges	<input type="checkbox"/> secure-read
<input checked="" type="radio"/> Test	<input checked="" type="checkbox"/> Versions	<input checked="" type="checkbox"/> Files	<input type="checkbox"/> secure-write
<input type="radio"/> Suite	<input checked="" type="checkbox"/> Properties	<input checked="" type="checkbox"/> Search	<input type="checkbox"/> secure-test
<input type="checkbox"/> Skip (Recursive)	<input checked="" type="checkbox"/> Refactor		
	<input checked="" type="checkbox"/> WhereUsed		

Help text:

Tags:

3.1: Un Wiki: Propriété de page



3.1: Un Wiki: Page de test

- Une page de test (Test Page)
 - Page qui exécute un scénario FitNesse

[Dolibarr](#) / [Connexion](#)

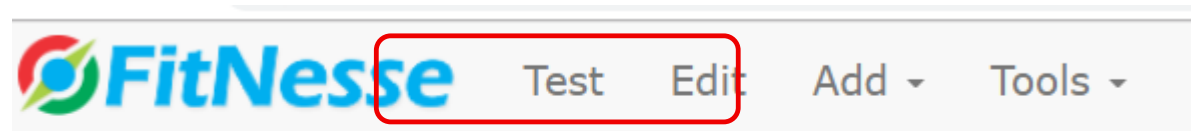
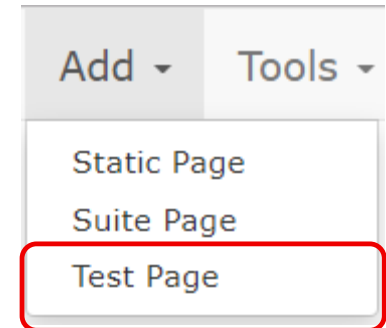
Page name:

Help text:

Tags:

☐

1 `<test page>`



[Dolibarr](#) / [Connexion](#) / [TestConnexionAvecIdentifiantsValides](#)

3.1: Un Wiki: Page de test

- Exécution du test



The screenshot displays the FitNesse web application interface. At the top, there is a navigation bar with the FitNesse logo, links for 'Test', 'Edit', 'Add', and 'Tools', and a green 'Execution Log' button. Below the navigation bar, the breadcrumb path 'Dolibarr / Connexion / TestConnexionAvecIdentifiantsValides' is shown. A summary bar indicates 'Test Pages: 0 right, 0 wrong, 0 ignored, 0 exceptions' and 'Assertions: 0 right, 0 wrong, 0 ignored, 1'. A section titled 'Exception occurred:' contains a detailed stack trace of a `fit.exception.FitParseException`.

```
fit.exception.FitParseException: Can't find tag: table
    at fit.Parse.findMatchingEndTag(Parse.java:89)
    at fit.Parse.((Parse.java:49))
    at fit.Parse.((Parse.java:37))
    at fit.FitServer.process(FitServer.java:80)
    at fit.FitServer.run(FitServer.java:56)
    at fit.FitServer.main(FitServer.java:41))
```

3.1: Un Wiki: Page spéciale

- Page avec des objectifs définis par le serveur
Fitnesse

- *FitNesse*
- *FrontPage*
- *PageFooter*
- *PageHeader*
- *PlugIns*
- *RecentChanges*
- ***RerunLastFailures***
- *TemplateLibrary*

Pied de page

Entête de page

Modèle de page

Insert Template

StaticPage
SuitePage
TestPage

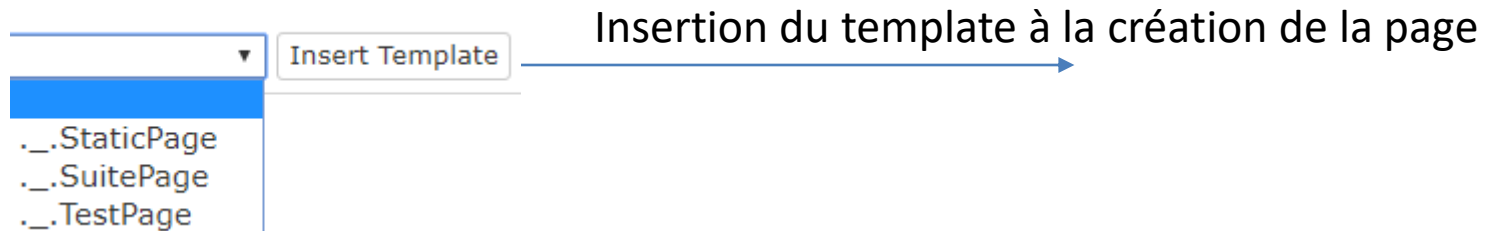
root (for global !path's, etc.)

3.1: Un Wiki: Page spéciale

- Template

Contents:

- *Static Page*
- **Suite Page** *
- Test Page +



3.2: Organisation des tests

- FitNesse est aussi un outil d'automatisation des tests d'acceptation
 - Le wiki permet l'écriture des tests
 - Le framework permet l'automatisation de ces tests
 - Le wiki permet d'exécuter les tests automatisés
- L'écriture des tests est basée sur une structure tabulaire

3.2: Ecriture d'un test

Dolibarr / Connexion / TestConnexionAvecIdentifiantsValides

Connexion avec des identifiants valides

Règles de gestion:

- Un utilisateur peut se connecter à l'application avec des identifiants valides.
- la date de dernière connexion est sauvegardée et affichée pour l'utilisateur

Prérequis: Aucun

Cas de test:

identifiant	mot de passe	estValide?
admin	Selenium&2018	O
jsmith	Selenium&2018	O
lsmith	Selenium&2018	O

3.2: Ecriture d'un test

- Table de décision:
 - les entrées et sorties du test sont les colonnes
 - Chaque ligne correspond à un cas de test
 - *Table de vérité*

Connexion	entrées		sortie
#Cas	identifiant	mot de passe	estValide?
administrateur	admin	Selenium&2018	O
commercial	jsmith	Selenium&2018	O
chef de projet	lsmith	Selenium&2018	O
Commentaire non traité			

3 tests

3.2: Ecriture d'un test

- Baseline Table de décision:
 - Regroupement des cas de test
 - Meilleur lisibilité pour les grands tableaux

		entrées	sortie
Cas de test:			
baseline: Connexion non valide			
#Cas	identifiant	mot de passe	estValide?
identifiant non valide	simpson 25		N
	1é&&!;		
	test@mail.fr		
mot de passe non valide	jsmith	123	N
	&&&&		
	ete rere		
login correct, mot de passe incorrect	lsmith	Selenium&2019	N

estValide=N pour les 3 tests

3.2: Ecriture d'un test

- Script Table:
 - Série d'actions permettant d'exécuter une procédure de test

Script de test web:

script	Connexion			Nom du script
lancer Chrome avec l'adresse	https://demo.glpi-project.org/			
saisir la valeur	admin	pour le champ	Login	Action d'opération avec des données en paramètre
saisir la valeur	admin	pour le champ	Password	
cliquer sur le bouton	Post			
check	est connecté	Oui		Action de vérification

3.2: Ecriture d'un test

- Script Table: Mot spécifique du table en première cellule
- **check**: opération de comparaison de l'instruction qui suit avec la valeur de la dernière cellule
 - **check not**: idem mais en négatif
- **ensure**: opération de comparaison de l'instruction qui suit avec la valeur booléenne de la dernière cellule (en vert si vrai, sinon rouge)
 - **reject** (en vert si faux, sinon rouge)
- **note**: instruction de commentaire, la ligne est ignorée
- **show**: la valeur retournée de l'instruction est affichée dans le rapport d'exécution
- \$variable: assigne la valeur de l'instruction à la variable

3.2: Ecriture d'un test

- Query Table:
 - Table fournissant les données d'une requête à vérifier
 - Pratique pour valider les données de prérequis

Query:Liste des identifiants		
#Cas	identifiant	mot de passe
administrateur	admin	Selenium&2018
commercial	jsmith	Selenium&2018
chef de projet	lsmith	Selenium&2018

Nom de la requête

Résultat de la requête

- Il existe aussi:
 - Subset Query
 - Ordered Query

3.2: Ecriture d'un test

- Table:
 - Table libre

Table: Bowling																				
3	5	4	/	X		X		3	4	6	/	7	2	3	4	9	-	4	/	3
	8		28		51		68		75		92		101		108		117			130

- Comments:
 - Tableau pour commenter (non exécuter durant le test)

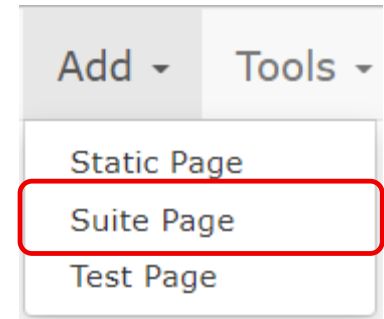
Détails:

Comment

L'environnement de test doit être déployé sur l'espace X

3.2: Suite de test

- Une suite de test
 - SubWiki composé
 - de pages de test
 - de pages static

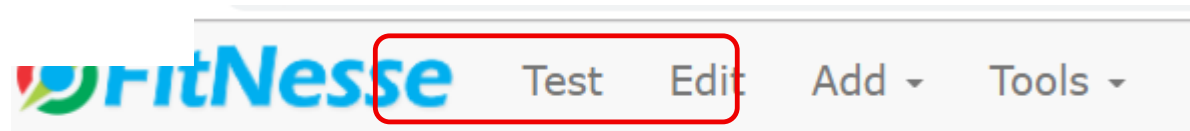


Page name:

Help text:

Tags:

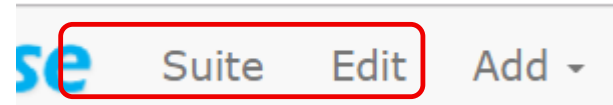
```
1 ▾ !1 Test suite X
2 !contents -R2 -g -p -f -h
```



Dolibarr / Connexion / TestConnexionAvecIdentifiantsValides

3.2: Suite de test

- Exécution d'une suite de test
 - Toutes les pages de test de la suite sont exécutées
 - Un rapport résumant le statut de tous les tests



Dolibarr / Version3

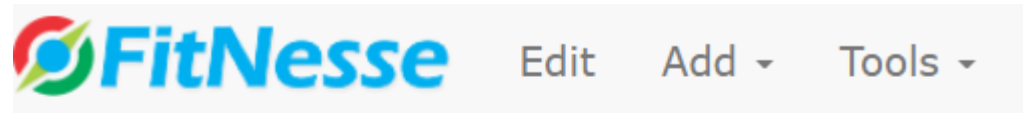
✓ **Test Pages:** 2 right, 0 wrong, 0 ignored, 0 exceptions **Assertions:** 7 right, 0 wrong

Test Summaries

✓ 5 right, 0 wrong, 0 ignored, 0 exceptions TestCaseA (0,092 seconds)
✓ 2 right, 0 wrong, 0 ignored, 0 exceptions TestCaseB (0,010 seconds)

3.2: Suite de test

- Particularité
 - Tous les tests héritent du moteur défini au niveau de la suite (page parent)
 - Tous les tests héritent de la Page Header et page Footer de la suite



Dolibarr / Version3 / PageFooter

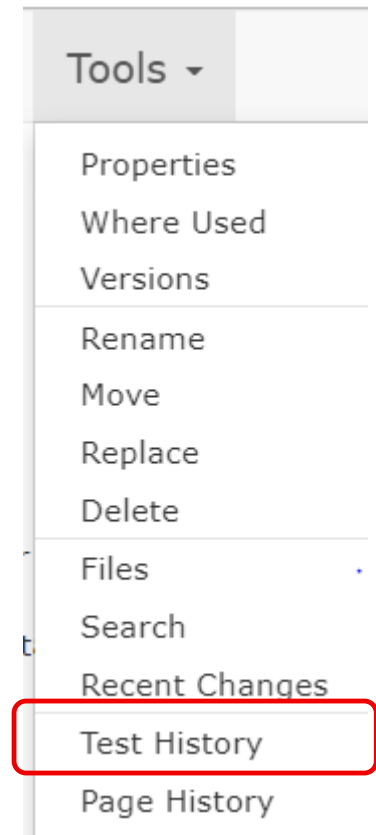
Version 3 Dolibarr

3.2: Historique

- Historique de l'exécution des tests

Test history

View as Overview		Purge > 30 days		Purge > 7 days		Purge all		Cancel		<input type="checkbox"/> Hide passed tests								
Page				Pass	Fail	Latest		Last 20 Results										
FitNesse.UserGuide.TwoMinuteExample				9	54	08 Oct 19, 20:09		-	-	-	-	+	-	-	+	+	-	-



3.3 Ecrivons les tests: TP

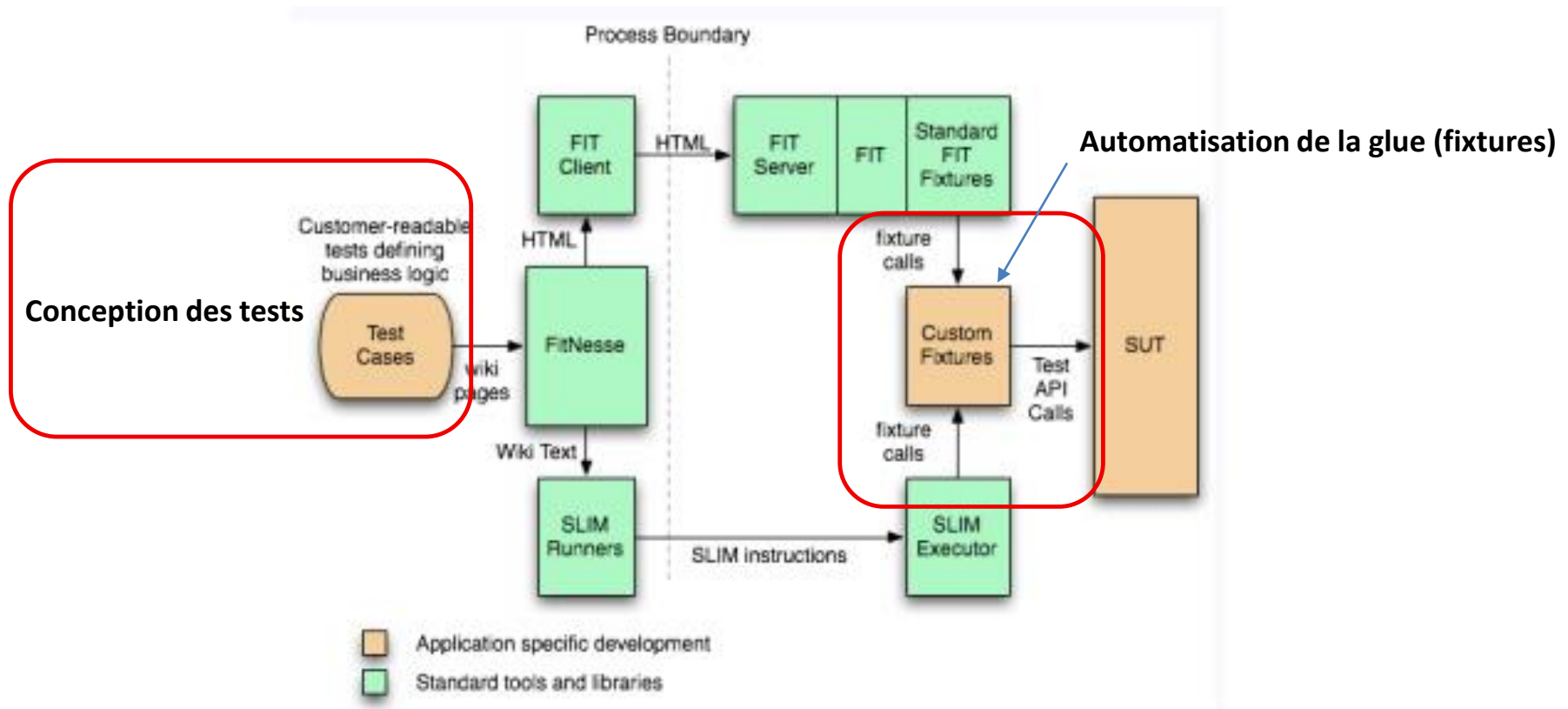
- Spécification des tests

Chap. 4: Implémentation des tests

- Automatisation des scénarii
 - Fixtures
 - Gestion des librairies
 - Gestion des variables
- Exécution des test
- Mavenisation d'un projet fitnessse
 - Gestion du projet maven
 - Exécution par Junit
- Plugins

4.1: Rappel

- Architecture



4.1: Automatisation

- La fixture
 - Fitnessse est une solution d'automatisation des tests
 - Le moteur Fitnessse analyse la spécification décrite dans le wiki pour exécuter les actions sur l'application à tester
 - La fixture (glue) est le code de test automatisant ces actions
 - En mappant le texte de la spécification avec une méthode de tests
- Technologie




4.1 Automatisation: une fixture

org.fitnessse.demo.fixtures.MathFixture

first number	second number	the greater is?
10	2	10
2	10	10
2	2	2
152	200	200

Le tableau de test

La librairie

dev	>	fitnessse	>	standard	>	lib
Nom						
 fixtures-1.0-SNAPSHOT.jar						

La fixture

```
package org.fitnessse.demo.fixtures;
import fit.ColumnFixture;

public class MathFitFixture extends ColumnFixture{

    Integer firstNumber;
    Integer secondNumber;

    public Integer theGreaterIs() { return Math.max(firstNumber, secondNumber); }
}
```

4.2: Fixture d'une table de décision

entrées			sortie
Connexion			
# Cas	identifiant	mot de passe	estValide?
administrateur	admin	Selenium&2018	O
commercial	jsmith	Selenium&2018	O
chef de projet	lsmith	Selenium&2018	O

} 3 tests

4.2: Fixture d'une table de décision

```
public class Connexion {  
    private String identifiant;  
    private String motDePasse;  
  
    private Login login = new Login();  
  
    public void setIdentifiant(String identifiant) { this.identifiant = identifiant; }  
    public void setMotDePasse(String password) { this.motDePasse = password; }  
  
    public boolean estValide(){  
        return false;  
    }  
}
```

entrées

sortie



Et l'application à tester?

4.2: Fixture d'une table de décision

- AUT*

```
package fr.dawan.formation.test.fixtures;

import fr.dawan.formation.common.Login;

public class Connexion {
    private String identifiant;
    private String motDePasse;

    private Login login = new Login();





    public void setIdentifiant(String identifiant) { this.identifiant = identifiant; }
    public void setMotDePasse(String password) { this.motDePasse = password; }

    public boolean estValide(){
        return login.connect(identifiant,motDePasse);
    }
}
```

4.2: Fixture d'une table de décision

- *Exécution: la fixture n'est pas importée dans Fitnessse*

Cas de test:

Connexion  Could not invoke constructor for Connexion[0]		
#Cas	identifiant	nr
administrateur	admin  The instance decisionTable_1.setIdentifiant. does not exist	S
commercial	jsmith  The instance decisionTable_1.setIdentifiant. does not exist	S
chef de projet	lsmith  The instance decisionTable_1.setIdentifiant. does not exist	S

4.2: Fixture: Import et Classpath

- *Packaging des classes: mvn clean package*
 - *Utilisation d'un plugin pour gérer les classes de test*
- *Edition de la page:*
 - *Classpath: répertoire des librairies des fixtures*
 - *Import: permettant d'importer un package pendant l'exécution d'un test*
 - *Library: permettant d'importer toutes les pages de test enfants*

```
|!define TEST_SYSTEM {slim}  
|!path lib/*.jar
```

```
|import  
|fr.dawan.formation.test.fixtures|
```

variable defined: TEST_SYSTEM=slim
classpath: lib/.jar*

import

fr.dawan.formation.test.fixtures

4.2: Fixture: Import et Classpath

- *Exécution*

Connexion			
#Cas	identifiant	mot de passe	estValide?
administrateur	admin	Selenium&2018	[false] expected [0]
commercial	jsmith	Selenium&2018	[false] expected [0]
chef de projet	lsmith	Selenium&2018	[false] expected [0]

Modifions la classe de l'application pour réussir le test

Connexion			
#Cas	identifiant	mot de passe	estValide?
administrateur	admin	Selenium&2018	true
commercial	jsmith	Selenium&2018	true
chef de projet	lsmith	Selenium&2018	true

4.2: Fixture d'une table de décision

- Les méthodes SLIM

```
public boolean estValide(){
    System.out.println("je fais le test");
    return login.connect(identifiant,motDePasse);
}

//Méthodes optionnelles
public void table(List<List<String>> table) {
    // est exécuté à la création de la table pendant l'exécution du test
    // renvoie les données du tableau
    System.out.println("La table contient " + table.size() + " lignes");
}

public void beginTable() {
    // utile avant de lancer l'exécution de toute la table
    // permet d'initier les données de prequis pour toute la table
    // équivalent d'un before Table
    System.out.println("On lance les tests");
}

/**** Pour chaque ligne *****/
int row=0;
public void reset() {
    // est exécuté avant le traitement de la ligne
    // équivalent d'un before Row
    row ++;
    System.out.println("Ligne: " + row);
}

public void execute() {
    // exécution de la ligne en affectant les données et avant le premier output
    System.out.println("identifiant: " + identifiant);
}

/***** */

public void endTable() {
    // permet de nettoyer les données
    // équivalent d'un After Table
    System.out.println("Fin des tests");
}
```

4.2: Fixture d'une table de décision

- Console*

slim:fitnesse.slim.SlimService

Test page:	Dolibarr.Connexion.TestConnexionAvecIdentifiantsValides
Command:	"C:\Program Files\Java\jdk1.8.0_121\bin\java" -cp D:\dev\fit\notifications\SNAPSHOT.jar fitnesse.slim.SlimService 1
Exit code:	0
Time elapsed:	0 seconds

Standard Output:

```
La table contient 4 lignes
On lance les tests
Ligne: 1
identifiant: admin
je fais le test
Ligne: 2
identifiant: jsmith
je fais le test
Ligne: 3
identifiant: lsmith
je fais le test
Fin des tests
```

4.2: Fixture d'une baseline

- *Identique à la table de décision*

Cas de test:

baseline: Connexion			
#Cas	identifiant	mot de passe	estValide?
identifiant non valide	simpson 25		false
	1é&&!;		
	test@mail.fr		
mot de passe non valide	jsmith	123	false
	&&&&		
	ete rere		
login correct, mot de passe incorrect	lsmith	Selenium&2019	false

baseline: Connexion			
#Cas	identifiant	mot de passe	estValide?
identifiant non valide	simpson 25		false
	1é&&!;		false
	test@mail.fr		false
mot de passe non valide	jsmith	123	false
	&&&&		false
	ete rere		false
login correct, mot de passe incorrect	lsmith	Selenium&2019	false

4.2: Fixture d'une Query

- *Vérification d'une liste de données d'une requête*

Query:Liste des identifiants		
#Cas	identifiant	mot de passe
administrateur	admin	Selenium&2018
commercial	jsmith	Selenium&2018
chef de projet	lsmith	Selenium&2018

4.2: Fixture d'une Query

```
package fr.dawan.formation.test.fixtures;
```

```
import java.util.List;
```

```
import static java.util.Arrays.asList;
```

```
public class ListeDesIdentifiants {
```

Query:Liste des identifiants

```
    public ListeDesIdentifiants(){}
```

```
    public List<List<List<String>>> query() {
```

```
        return
```

```
            asList( // //Table
```

```
                asList( // Ligne
```

```
                    asList("identifiant", "admin"), // Cellules
```

```
                    asList("mot de passe", "Selenium&2018")
```

```
                ),
```

```
                asList(
```

```
                    asList("identifiant", "jsmith"), // Cellules
```

```
                    asList("mot de passe", "Selenium&2018")
```

```
                ),
```

```
                asList(
```

```
                    asList("identifiant", "lsmith"), // Cellules
```

```
                    asList("mot de passe", "Selenium&2018")
```

```
                )
```

```
            );
```

```
        }
```

```
    }
```

```
public class ListeDesIdentifiants {  
    private String role;  
    public ListeDesIdentifiants(String role){  
        this.role=role;  
    }  
}
```

4.2: Fixture d'une Query

Prérequis:

Query: Liste des identifiants		
#Cas	identifiant	mot de passe
administrateur	admin	Selenium&2018
commercial	jsmith	Selenium&2018
chef de projet	lsmith	Selenium&2018

4.2: Fixture d'une Query

- *Gestion des résultats par la fixture Query*

Prérequis:

Query:Liste des identifiants		
#Cas	identifiant	mot de passe
administrateur	admin	[Selenium&2018] expected [Selenium&2019]
[commercial] missing	dsmith	Selenium&2018
chef de projet	lsmith	Selenium&2018
[chef de projet] missing	bsmith	Selenium&2018
field #Cas not present	jsmith	Selenium&2018

4.2: Fixture d'une Query

- *Subset*

Prérequis:

Query: Liste des identifiants		
#Cas	identifiant	mot de passe
administrateur	admin	Selenium&2018
commercial	jsmith	Selenium&2018
chef de projet	lsmith	Selenium&2018
chef de projet	bsmith	Selenium&2018
field #Cas not present	dsmith	Selenium&2018

Prérequis:






Subset Query: Liste des identifiants

#Cas	identifiant	mot de passe
administrateur	admin	Selenium&2018
commercial	jsmith	Selenium&2018
chef de projet	lsmith	Selenium&2018
chef de projet	bsmith	Selenium&2018

4.2: Fixture d'un Script

- *Classe de script*

Script de test web:

script	Connexion		
lancer Chrome avec l'adresse  Method lancerChromeAvecLAdresse[1] not found in fr.dawan.formation.test.fixtures.Connexion.	https://demo.glpi-project.org/		
saisir la valeur  Method saisirLaValeurPourLeChamp[2] not found in fr.dawan.formation.test.fixtures.Connexion.	admin	pour le champ	Login
saisir la valeur  Method saisirLaValeurPourLeChamp[2] not found in fr.dawan.formation.test.fixtures.Connexion.	admin	pour le champ	Password
cliquer sur le bouton  Method cliquerSurLeBouton[1] not found in fr.dawan.formation.test.fixtures.Connexion.	Post		
check	est connecté	Oui  Method estConnecté[0] not found in fr.dawan.formation.test.fixtures.Connexion.	

4.2: Fixture d'un Script

- *Classe de script*

```
/*Pour la fixture Script*/
```

```
public void lancerChromeAvecLAdresse(String url){  
    System.out.println("Je lance l'application Chrome");  
    System.out.println("Je navigue à l'adresse: " + url );  
}  
  
public void saisirLaValeurPourLeChamp(String valeur, String champ){  
    System.out.println("Je renseigne dans le champ " + champ + " la valeur: " + valeur);  
}  
  
public void cliquerSurLeBouton(String bouton){  
    System.out.println("Je clique sur le bouton " + bouton);  
}  
  
public boolean estConnecté(){  
    System.out.println("Je suis connecté");  
    return true;  
}
```

4.2: Scenario Table

- *Permet de réutiliser dans un tableau des scripts et des fixtures*
- *Les arguments sont possibles avec des paramètres @param*
- *Un scenario peut être utilisé pour exécuter un jeu de données sur un même script*

scenario	Connexion	username	password
lancer Chrome avec l'adresse	https://demo.glpi-project.org/		
saisir la valeur	@username	pour le champ	Login
saisir la valeur	@password	pour le champ	Password
cliquer sur le bouton	Post		

4.2: Scenario Table

- *Permet de réutiliser dans un tableau des scripts et des fixtures*
- *Les arguments sont possibles avec des paramètres @param*
- *Un scenario peut être utilisé pour exécuter un jeu de données sur un même script*

scenario	Connexion	avec	username,password
lancer Chrome avec l'adresse	https://demo.glpi-project.org/		
saisir la valeur	@username	pour le champ	Login
saisir la valeur	@password	pour le champ	Password
cliquer sur le bouton	Post		

4.2: Scenario Table

script	Connexion
--------	-----------

scenario	Connexion _ avec _		username,password
lancer Chrome avec l'adresse	https://demo.glpi-project.org/		
saisir la valeur	@username	pour le champ	Login
saisir la valeur	@password	pour le champ	Password
cliquer sur le bouton	Post		

Connexion avec	
username	password
admin	admin
normal	normal



Connexion avec			
username		password	
-admin		admin	
scenario	Connexion _ avec _	username,password	
lancer Chrome avec l'adresse	https://demo.glpi-project.org/		
saisir la valeur	admin	pour le champ	Login
saisir la valeur	admin	pour le champ	Password
cliquer sur le bouton	Post		
+normal		normal	

4.3: Variable

- *Définir une variable*

lancer Chrome avec l'adresse *undefined variable: URL*

```
!define URL {https://demo.glpi-project.org/}
```

- *Utiliser une variable*

```
script
lancer Chrome avec l'adresse ${URL}
saisir la valeur admin pour le champ Login .
```

- *Scope*
 - *Toutes les pages de tests descendantes*

Dolibarr / Connexion

variable defined: URL=https://demo.glpi-project.org/

Contents:

- **Test Connexion Avec Identifiants Valides +** : Connexion avec des identifiants valides

4.3: Setup & Tear Down

- *Utilisation des pages spéciales pour définir les Hooks*
 - *Page Header: pour un setup*
 - *Page Footer: pour un tear down*

En définissant au niveau d'un subwiki (Test Suites)

- *Tous les pages de test de la suite hériteront du Setup et Tear Down*

- *Utilisation de la commande: `!include`*

`!include .Dolibarr.SetUpDolibarr`

[Dolibarr](#) / [SetUpDolibarr](#)

Help text:

Tags:

[Spreadsheet to FitNesse](#)

[FitNesse to Spreadsheet](#)

[Format](#)

```
1 |!define TEST_SYSTEM {slim}
2 |!define URL {https://demo.glpi-project.org/}
3 |!path lib/*.jar
4 |import
5 |fr.dawan.formation.test.fixtures
6 |script |Connexion
7 |
```

4.3: Setup & Tear Down

– Included page: [.Dolibarr.SetupDolibarr \(edit\)](#)

variable defined: TEST_SYSTEM=slim
variable defined: URL=https://demo.glpi-project.org/
classpath: lib/.jar*

import

fr.dawan.formation.test.fixtures

script

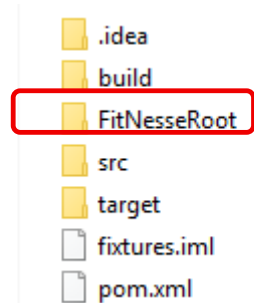
Connexion

4.3 Automatisons les tests: TP

- Spécification des tests

4.5: Junit Runner

- *Utilisation d'un Runner pour exécuter les tests à partir de Junit*
 - *Mode IDE développeur*
 - *Pas besoin de démarrer le serveur*
 - *Mettre le répertoire FitnessseRoot à la racine du projet Java*



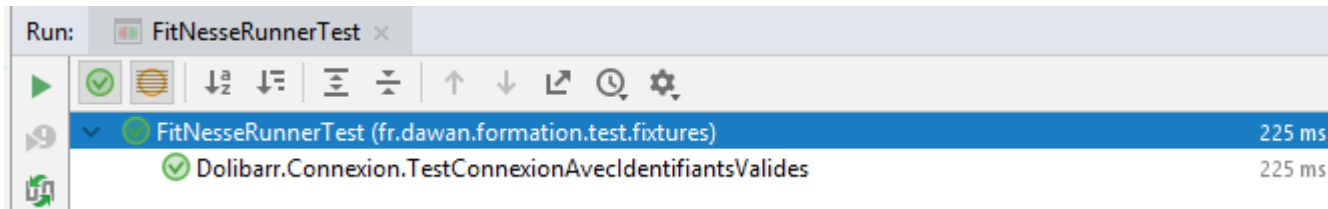
```
package fr.dawan.formation.test.fixtures;

import fitnessse.junit.FitNesseRunner;
import org.junit.runner.RunWith;

@RunWith(FitNesseRunner.class)
@FitNesseRunner.Suite("Dolibarr.Connexion.TestConnexionAvecIdentifiantsValides")
@FitNesseRunner.FitnessseDir(".")
@FitNesseRunner.OutputDir("./build/fitnessse-results")
public class FitNesseRunnerTest {
}
```

4.5: Junit Runner

- *Exécution*



slim:in-process

La table contient 4 lignes

On lance les tests

Ligne: 1

identifiant: admin

je fais le test

Ligne: 2

identifiant: jsmith

je fais le test

Ligne: 3

identifiant: lsmith

je fais le test

Fin des tests

La table contient 8 lignes

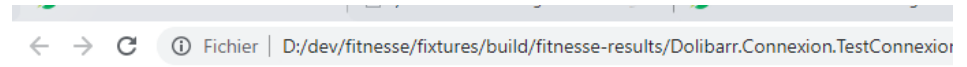
On lance les tests

Ligne: 1

identifiant: simpson 25

je fais le test

27/10/2017



Dolibarr.Connexion.TestConnexionAvecIdentifiantsValides

variable defined: TEST_SYSTEM=slim

classpath: lib/*.jar

import

fr.dawan.formation.test.fixtures

Règles de gestion:

- Un utilisateur peut se connecter à l'application avec des identifiants valides.
- la date de dernière connexion est sauvegardée et affichée pour l'utilisateur



4.5: Junit & Maven

- *Exécution avec maven par un mvn test*
- *Utilisation de plugin maven existant*
 - *fitnesse-launcher-maven-plugin*

4.6: Debug des fixtures et des tables

- *Exécution Logs*
 - Utilisation de la console pour afficher les valeurs
 - Remonte les erreurs de tableaux dans la console
- *Utilisation de l'exécution par Junit*
- *Utilisation de plugin Fitnessse (Intellij)*



```
29 // utile avant de lancer l execution de toute la table
30 // permet d'initier les données de prequis pour toute la
31 // équivalent d'un before Table
32 System.out.println("On lance les tests");
33 }
34
35 /***** Pour chaque ligne *****/
36 int row=0; row: 0
37 public void reset() {
38 // est exécuté avant le traitement de la ligne
39 // équivalent d'un before Row
40 row ++;
41 System.out.println("Ligne: " + row);
```

4.7: WikiImport

- *Importer des subwikis dans un wiki centralisé*
 - *Chaque développeur réalise ses tests*
 - *Intégration de l'ensemble des subwikis sur un serveur d'intégration*

Wiki Import

Supply the URL for the wiki you'd like to import.

Remote Wiki

URL:



Automatically update imported content when executing tests

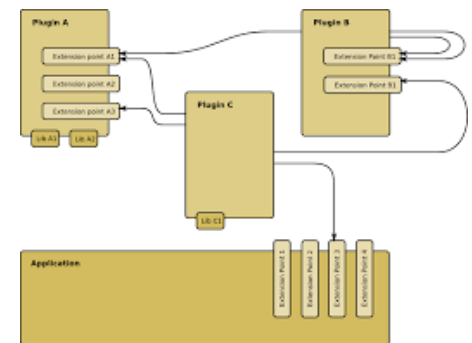
Import

4.8 Automatisons les tests: TP

- Spécification des tests

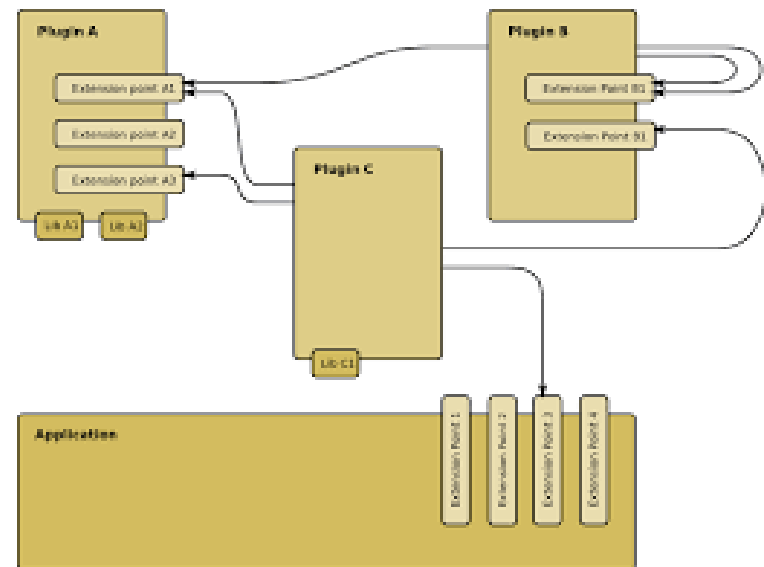
Chap. 5: Gestion des plugins

- Plugins
 - Généralités
 - Langage supporté
 - Gestion des variables



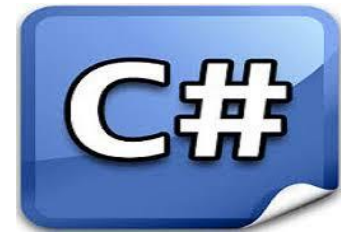
5.1 Généralités

- *Permettre de rajouter des fonctionnalités à Fitnessse*
 - Framework
 - Librairies
 - Tables



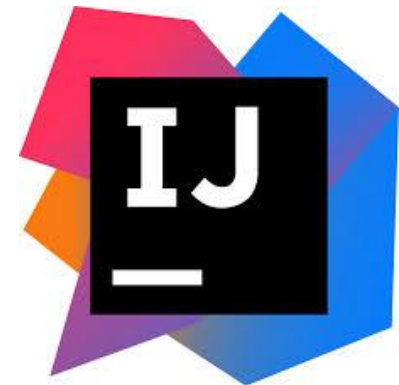
5.2 Langage

- *Extension à d'autres langages d'écriture des fixtures*



5.3 SCM plugin

- *Gestion du code dans un SCM et autres*

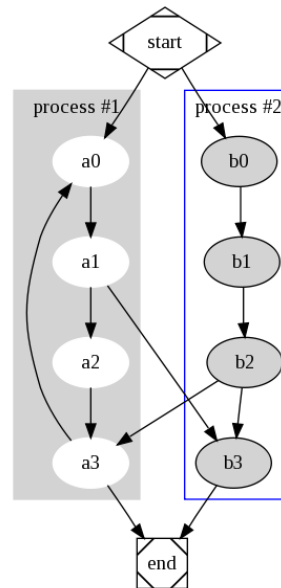


5.3 Ajout de table (Markup)

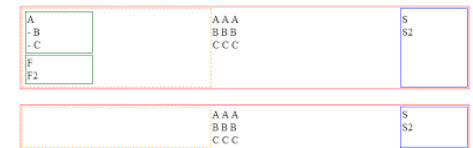
- *Plugins ajoutant des fonctionnalités de fixtures tables dans Fitness*



PlantUML



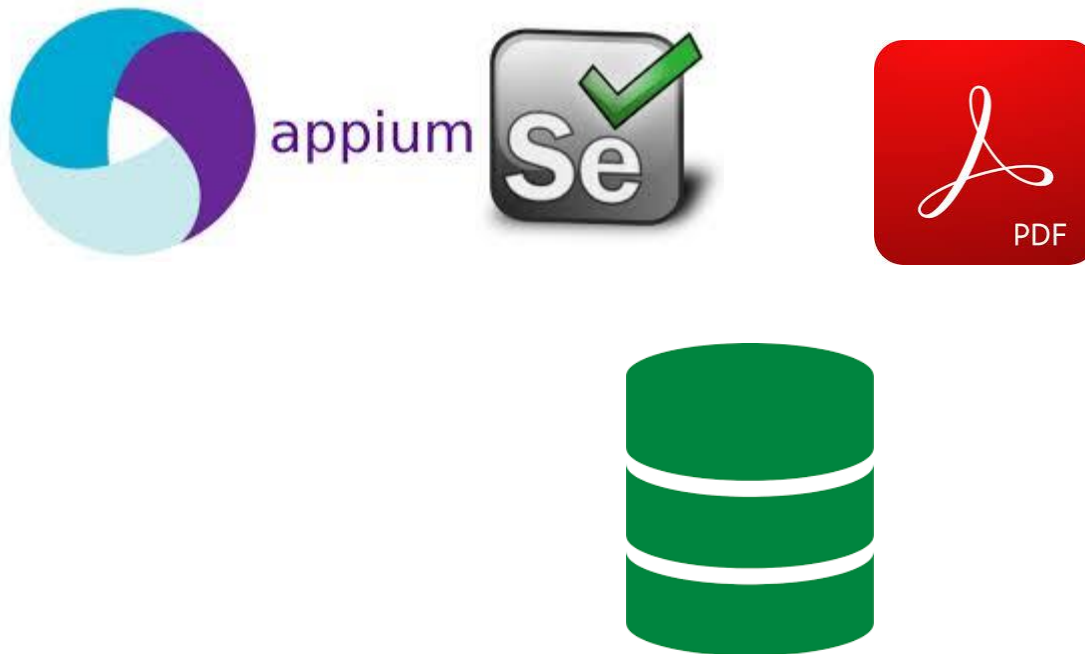
GraphViz



HSAC

5.3 Fixtures

- *Plugins ajoutant des fixtures de script dans Fitnesse*



5.4 TP

- *Installation du plugin HSAC*
 - <https://github.com/fhoeben/hsac-fitnessse-plugin>

Chap. 6: Fitnesse & Selenium

- Selenium
- Ecriture de fixtures pour driver Selenium
- Utilisation de plugins: BrowserTest



6.1: Selenium

- Driver pour piloter les actions sur des navigateurs Web
- Bibliothèques multiplateformes et langages
 - Java
 - C#
 - Ruby
 - Python

```
<!-- https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java -->  
<dependency>  
  <groupId>org.seleniumhq.selenium</groupId>  
  <artifactId>selenium-java</artifactId>  
  <version>3.141.59</version>  
</dependency>
```

6.2: Ecriture de la fixture

- Ajout de la dépendance selenium dans le projet de développement

```
<dependency>
  <groupId>org.seleniumhq.selenium</groupId>
  <artifactId>selenium-java</artifactId>
  <version>3.141.59</version>
</dependency>

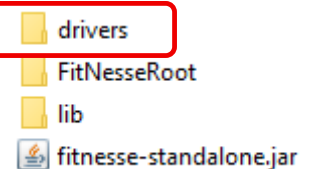
<dependency>
  <groupId>junit</groupId>
  <artifactId>junit</artifactId>
  <version>4.12</version>
  <scope>test</scope>
</dependency>
```

- Utilisation des librairies Selenium pour compléter les actions du Script

6.2: Ecriture de la fixture

- Initialisation du driver Selenium

```
private ChromeDriver driver;  
public Connexion(){  
    System.setProperty("webdriver.chrome.driver", "drivers/chromedriver.exe");  
}  
public void beginTable() {  
    // utile avant de lancer l'exécution de toute la table  
    // permet d'initier les données de prequis pour toute la table  
    // équivalent d'un before Table  
    driver = new ChromeDriver();  
}
```



```
public void endTable() {  
    // permet de nettoyer les données  
    // équivalent d'un After Table  
    driver.close();  
    driver.quit();  
}
```

6.2: Ecriture de la fixture

- Automatiser la page de test Connexion



6.2: Ecriture de la fixture

- Problème URL

lancer Chrome avec l'adresse <https://demo.glpi-project.org>

```
<a href="https://demo.glpi-project.org">https://demo.glpi-project.org</a>
```



6.2: Ecriture de la fixture

- Problème URL: Nettoyage de l'url avec une expression régulière

```
private static final Pattern LINKPATTERN = Pattern.compile( regex: "<a(\\s+.*?)?\\s+href=\"(.*?)\".*?>(.*?)</a>(.*?)", Pattern.CASE_INSENSITIVE);

/**
 * Gets a URL from a wiki page value.
 * @param htmlLink link as present on wiki page.
 * @return address the link points to (if it is an 'a'), the original link otherwise.
 */
public String getUrl(String htmlLink) {
    String result = htmlLink;
    if (htmlLink != null) {
        Matcher linkMatcher = LINKPATTERN.matcher(htmlLink);
        if (linkMatcher.matches()) {
            String href = linkMatcher.group(2);
            href = StringEscapeUtils.unescapeHtml4(href);
            result = href + linkMatcher.group(4);
        }
    }
    return result;
}
```

6.2: Ecriture de la fixture

- Problème Fermeture du browser
 - Revenir aux conditions initiales
 - A quel moment fermer le driver?



6.2: TP

- Automatiser le scénario fourni



6.3: Plugin

- Existence de plusieurs plugins Selenium



Selenese

Automated testing with FitNesse and HSAC

6.3: BrowserTest

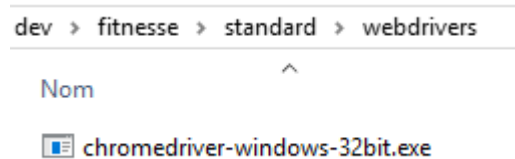
- <https://github.com/fhoeben/hsac-fitness-fixtures/blob/master/wiki/FitNesseRoot/HsacExamples/SlimTests/BrowserTests.wiki>

```
'''Command'''
|open <url>
|click <place>
|click if available <place>
|double click <place>
|right click <place>
|shift click <place>
|control click <place>
|enter <value> as <input>
|enter <value> for <input>
|select <value> as <input>
|select <value> for <input>
|enter date <value> as <place>
|available options for <input>
|normalized available options for <input>
|type <text>
|press <key>
|clear <input>
|take screenshot <name>
|page source
|save page source
|value of <input>
|normalized value of <input>
|values of <place>
|normalized values of <place>
|number for <list item text>
```

```
'''Description'''
|opens the specified URL
|performs mouse click at the designated place (e.g. link/button/
|performs mouse click at the designated place (e.g. link/button/
|performs mouse double click at the designated place (e.g. link/
|performs right mouse button click (a.k.a. context click) at the
|performs mouse click while holding shift key at the designated
|performs mouse click while holding control key at the designate
|clears the current value of an input and types the value
|types the value, appending to the current value of the input
|selects a value in the drop down box (or radio button), clearin
|selects a value in the drop down box (or radio button), appendi
|replaces the current value of an input of type 'date' (i.e. usi
|retrieves all possible values for the drop down box
|retrieves all possible normalized values for the drop down box.
|enters the supplied text in the currently active element
|sends press of a [[key]][http://seleniumhq.github.io/selenium/do
|removes the content in the input
|stores the current browser's content as <name><unique>.png (use
|gets the current browser's content (used in combination with 's
|stores the current browser's content as .html file (used in com
|retrieves the current value of the input (for use with 'check'
|retrieves the normalized current value of the input (for use wi
|retrieves all values of place, which is expected to be a 'selec
|retrieves all normalized values of place, which is expected to
|retrieves the number of the item (in a numbered list) with the
```

6.3: BrowserTest

- Librairies
- Configuration du driver



6.3: BrowserTest

- Exemple

import
nl.hsac.fitnessse.fixture.slim.web

script	selenium driver setup
start driver for	chrome

script	browser test		
open	https://demo.glpi-project.org/		
set Browser Size To Maximum			
enter	admin	as	Login
enter	admin	as	Password
click	Post		
click	Logout		

script	selenium driver setup
stop driver	



6.3: BrowserTest

- Exemple

script	browser test		
open	https://demo.glpi-project.org/		
set Browser Size To Maximum			
enter	admin	as	Login
enter	admin	as	Password
click	Post		
click	Assets		
click	Computers		
click	Add		
enter	AXT-156	as	Name
click	xpath=//label[text()='Location']/following::span[@class='select2-selection__arrow']		
click	location 0		
click	name=add		
click	AXT-156		
check	value of	Name	AXT-156
click	Logout		

6.3: TP

- Automatiser le scénario fourni



6.4: Construire son framework

- Orienté comportement 😊





