

Génie logiciel avancé

Spécification, Conception, Développement et Gestion du cycle de vie d'une solution informatique

2015-2016



Plan des cours

- 1 Modalités du cours,
Présentation du **génie logiciel**,
Cycles de développement,
- 2 Cycles de développement (suite),
Spécifications
- 3 Planification de projet,
Conception
- 4 Techniques de conception, UML
- 5 Design patterns
- 6 Environnement de développement :
IDE, SCM, Environnement de test, Intégration continue, etc.
- 7 Suivi de projet,
Tests et revues de code
- 8 Documentation
- 9 Recettes, présentation client
- 10 Déploiement,
Gestion des évolutions fonctionnelles
- 11 Présentation des projets par les étudiants
- 12 Séance de questions / réponses

Conception




Génie Logiciel Avancé Cours 2 — Conception

Stefano Zacchioli
zack@pps.univ-paris-diderot.fr

Laboratoire PPS, Université Paris Diderot

2014-2015

URL <http://upsilon.cc/zack/teaching/1415/gla/>
Copyright © 2011-2015 Stefano Zacchioli
© 2010 Yann Régis-Gianas
License Creative Commons Attribution-ShareAlike 4.0 International License
http://creativecommons.org/licenses/by-sa/4.0/deed.en_US


Présentation complète : <https://upsilon.cc/~zack/teaching/1415/gla/cours-02-design.pdf>

Auteurs : Stefano Zacchioli & Yann Régis-Gianas

Planification d'un projet

1. Spécifications

- Clarifier le plus possible les fonctionnalités demandées
- Lever toutes les ambiguïtés
- Définir les limites du système à réaliser

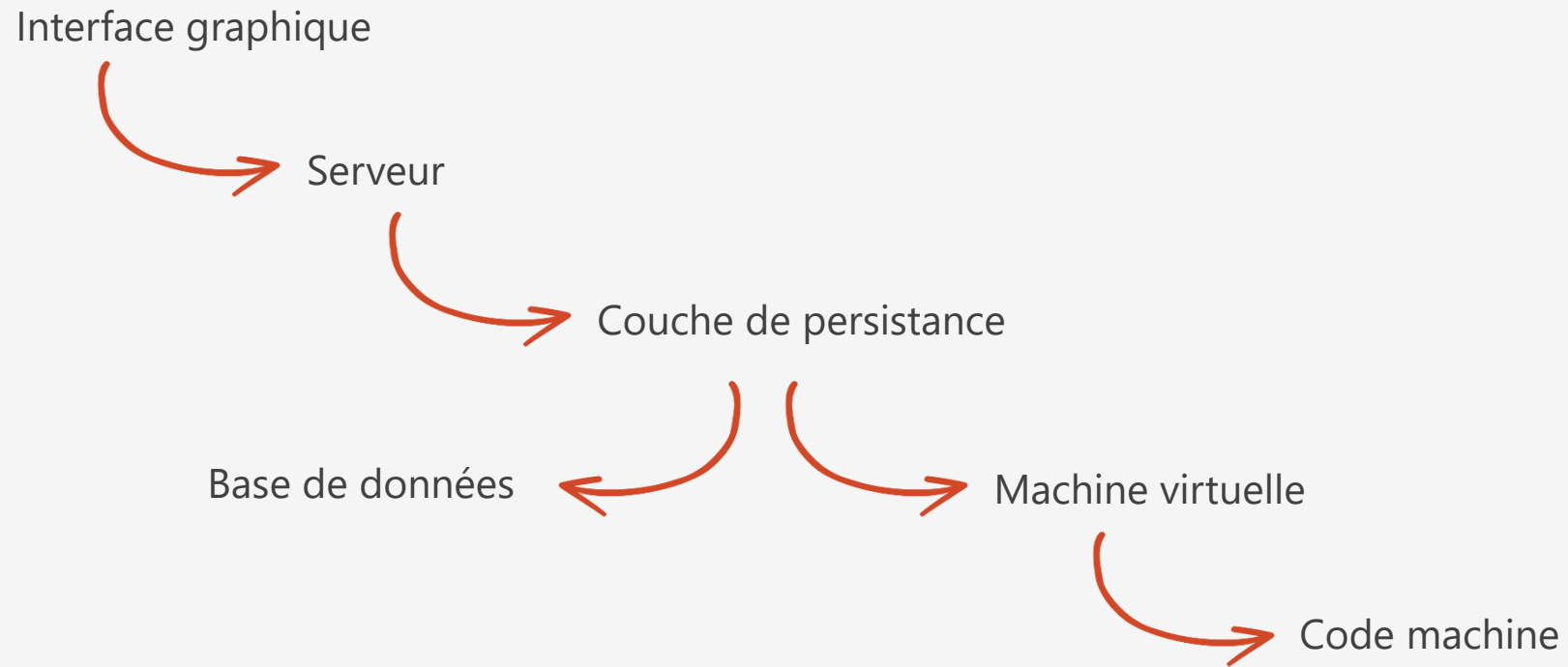
2. Conception

1. Définir les composants du système et son architecture globale
2. Définir toutes les interfaces
3. Concevoir chaque composant

3. Réalisation

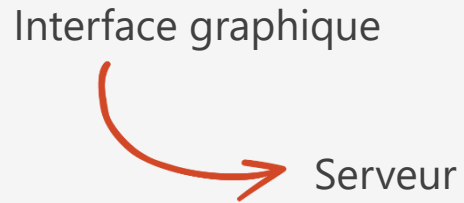
.... le plus tard possible... C'est une **fausse « bonne idée »** que de vouloir développer très tôt !

Niveaux d'abstraction



Chaque niveau d'abstraction a un rôle

Interfaces



Les liens entre les différentes couches définissent l'interface

Une interface permet :

1. de rendre explicite les échanges entre deux niveaux
2. de cacher ce qui n'est pas utile
3. de faire évoluer l'implémentation d'un niveau sans modifier le niveau supérieur

Interfaces – IHM / Serveur

	A	B	C	D	E	F
1	Module	View	Fonction	Method	Path Param 1	Path Param 2
143	action-followup	attachment	Return attachments	GET	X = id of the action	
144	action-followup	attachment	Add an attachment	POST	X = id of the action	
145	action-followup	attachment	Return an attachment (with the conte	GET	X = id of the action	Y = id of the attachment
146	action-followup	attachment	Delete an attachment	DELETE	X = id of the action	Y = id of the attachment
147	action-followup	note	Return a note	GET	X = id of the action	
148	action-followup	note	Modify a note	POST	X = id of the action	
149	action-followup	discussion	Return a discussion	GET	X = id of the action	
150	action-followup	discussion	Return discussion users	GET	X = id of the action	
151	action-followup	discussion	Modify discussion users	POST	X = id of the action	
152	action-followup	discussion	Return the attachment of a message	GET	X = id of the action	Y = id of the attachment
153	action-followup	discussion	Add a message	POST	X = id of the action	
154	publication	list-publication	Return the list of publications	GET		
155	publication	list-publication	Export the list of publications	GET		
156	publication	list-publication	Delete a publication	DELETE		
157	publication	create-publication	Create a publication	POST		
158	publication	view-publication	Return a publication	GET	X = id of the publication	
159	publication	view-publication-attachment	Return an attachment (with the conte	GET	X = id of the publication	Y = id of the attachment
160	publication	edit-publication	Return a publication	GET	X = id of the publication	
161	publication	edit-publication	Return publication users	GET	X = id of the publication	
162	publication	edit-publication	Modify a publication	POST	X = id of the publication	
163	user-prefrences	view-preferences	Returns user preferences	GET		
164	user-prefrences	edit-preferences	Returns user preferences	GET		
165	user-prefrences	edit-preferences	Modify user preferences	POST		
166	user-prefrences	edit-password	Modify user password	POST		
167	reporting	home	Return the list of existing filters	GET		
168	reporting	home	Modify the existing filters	POST		
169	reporting	occurrences-period	Return the occurrences divided by per	GET		

Expliciter tous les web-services nécessaires

Interfaces – Serveur / Persistance

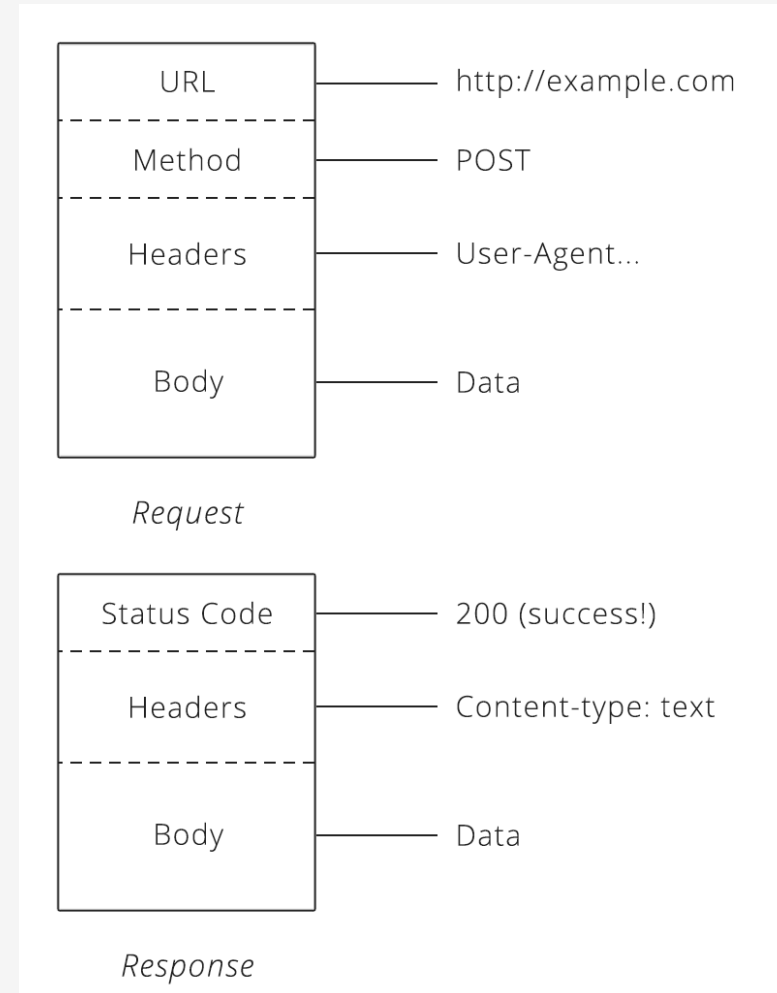
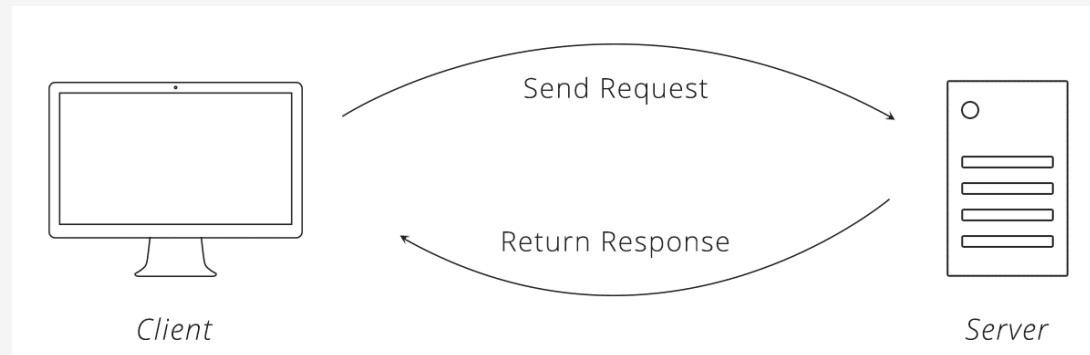
```
public class Action {  
    public enum State {  
        OPEN, CLOSED, VALIDATED;  
    };  
  
    public enum Type {  
        SAFETY, FINDING, OTHER;  
    };  
  
    public enum Mode {  
        CORRECTIVE, PREVENTIVE;  
    };  
  
    public static class ActionDelay {  
        public Integer value;  
        public String color;  
  
        public ActionDelay(Integer value, String color) {  
            super();  
            this.value = value;  
            this.color = color;  
        }  
    }  
  
    @PrimaryKey  
    @Persistent(valueStrategy = IdGeneratorStrategy.NATIVE)  
    protected Long id;
```

```
public interface ActionDao {  
  
    /**  
     * Find the actions related to this defense.  
     *  
     * @param defense  
     *           a Bow-Tie defense  
     * @return list of safety actions  
     */  
    List<Action> getSafetyActions(Defense defense);  
  
    /**  
     * Find the actions related to this impact assessment.  
     *  
     * @param assessment  
     *           an impact assessment  
     * @return list of safety actions  
     */  
    List<Action> getSafetyActions(ImpactAssessment assessment);
```

Définir les objets métiers et les méthodes d'accès
Bien garder en tête qu'il s'agit d'interfaces !

Annexe – Web-services REST

Les web-services REST – Un exemple parmi d'autres...



Annexe – Web-services REST

La spécification Java : JAX-RS

Un exemple d'implémentation : **Jersey**

```
7
8  /**
9   * Root resource (exposed at "myresource" path)
10  */
11  @Path("myresource")
12  public class MyResource {
13
14      /**
15       * Method handling HTTP GET requests. The returned object will be sent
16       * to the client as "text/plain" media type.
17       *
18       * @return String that will be returned as a text/plain response.
19       */
20      @GET
21      @Produces(MediaType.TEXT_PLAIN)
22      public String getIt() {
23          return "Got it!";
24      }
25  }
```

Annexe – Web-services REST

```
package com.mkyong.json;


import java.util.List;

public class User {

    private String name;
    private int age;
    private List<String> messages;

    //getters and setters
}
```

Conversion Java -> JSON



```
{
  "name" : "mkyong",
  "age" : 33,
  "messages" : [ "hello jackson 1", "hello jackson 2", "hello jackson 3" ]
}
```

Annexe – Web-services REST

Un outil de test
Plugin Chrome et Firefox
REST Client

[-] Request

Method

POST

▼

URL

http://95.142.173.170:8448/ws/safety-occurrence/list-occurrence?save=true

☆ ▼

SEND

Headers

Remove All

Content-Type: application/json ×

Body

zaeaze

[-] Response

Response Headers

Response Body (Raw)

Response Body (Highlight)

Response Body (Preview)

1. Status Code : 403 Page requires login.

2. Cache-Control : must-revalidate,no-cache,no-store

3. Content-Length : 325

4. Content-Type : text/html; charset=ISO-8859-1

5. Date : Tue, 02 Feb 2016 21:18:00 GMT

6. Server : Jetty(9.2.14.v20151106)

7. WWW-Authenticate : Basic realm="realm"

Quelques pointeurs pour la réalisation du projet

RESTClient	https://addons.mozilla.org/fr/firefox/addon/restclient/
Jersey	http://jersey.java.net
Apache CXF	https://cxf.apache.org/
Jackson	https://github.com/FasterXML/jackson
Jettison	https://github.com/codehaus/jettison