

Composants

Julien Ponge, INSA de Lyon

Composants

Duplication de code

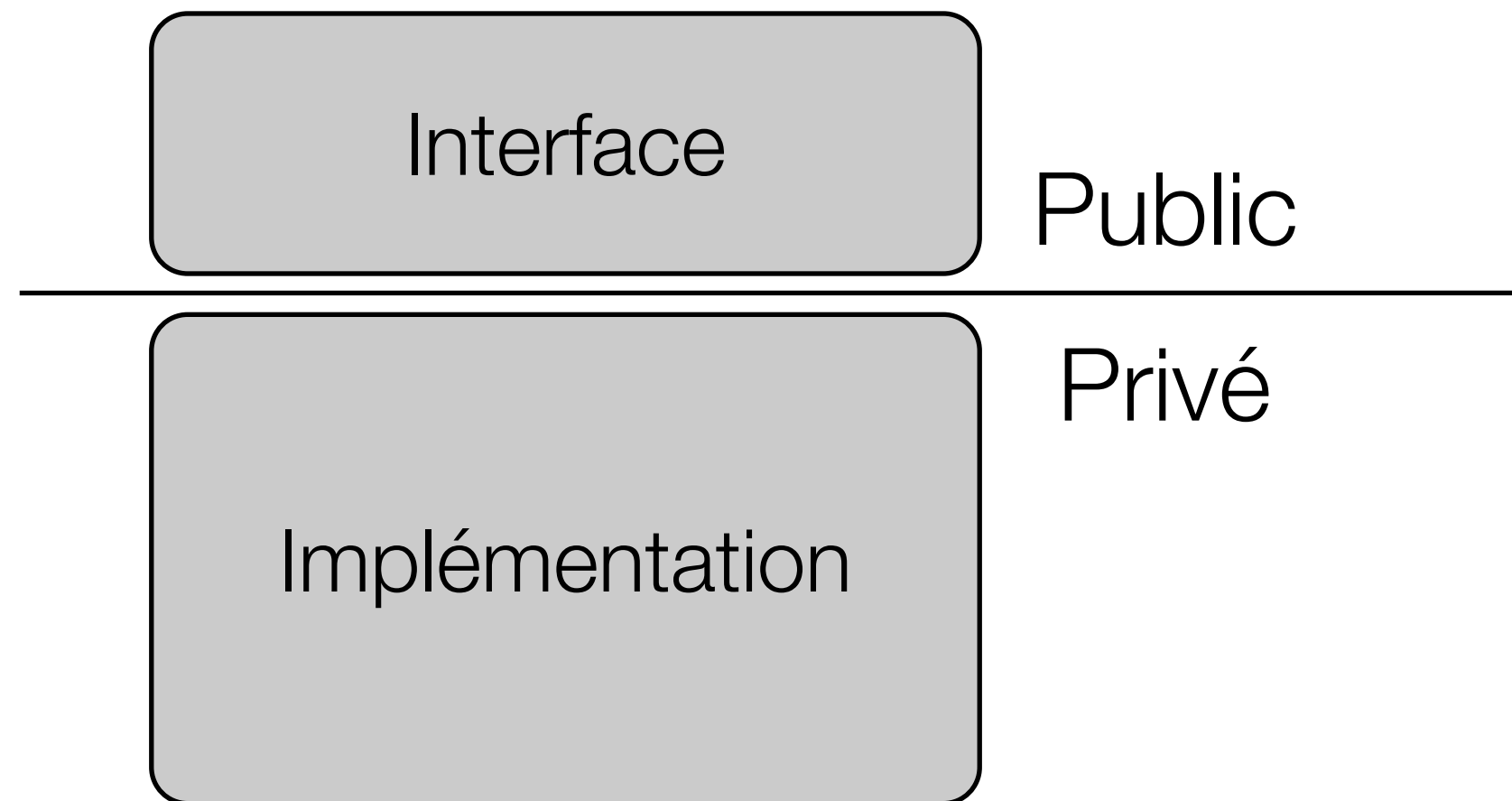
- Un gag récurrent dans le domaine du logiciel
- Aucune valeur dans la réécriture de code déjà écrit par ailleurs
- 70's : monolithes ré-écrits en permanence : base, présentation, etc
- Recodez-vous à chaque fois :
 - listes chaînées, hash tree ?
 - base de données ? persistence ?
 - widgets graphiques ?
 - HTTP ? TCP/IP ?

Composants : analogie

- Disques durs
- Écrans VGA
- USB
- Batterie de voiture
- Prises électriques (!)

Définition minimale d'un composant

- Interface publique : ce que le composant offre au code client
- Implémentation : ce qui est masqué, la partie cachée de l'iceberg
- Un composant doit être une unité fonctionnelle



Vous avez déjà écrit des composants (!)

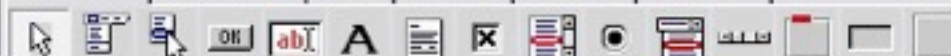
- Java : interfaces / implémentations
- C : .h vs .c vs static
- Ada, Pascal, ...
- Les langages objets se pretent naturellement au jeu, mais les langages procéduraux également

Un marché de composants (80's / 90's)

- Bibliothèques partagées
- Visual Basic
- COM / DCOM
- Delphi
- Java Beans
- Corba

Que font ces composants ?

- Composants graphiques
- Connexion client / serveur
- Logiques métier diverses et variées
- Accès à une couche de donnée, relationnelle ou non



Object Inspector

Form1: TForm1	
Properties	Events
ACTIVECONTROL	
ALIGN	alnone
AUTOSIZE	False
BORDERWIDTH	0
CAPTION	Form1
COLOR	clBtnFace
CLIENTHEIGHT	300
CLIENTWIDTH	400
CONSTRAINTS	(TSIZECONST)
ENABLED	True
FORMSTYLE	fsnormal
ICON	(TICON)

Breakpoint list

State	Filename/Address	Line/Length	Condition	Action	Pass
	project1.lpr	9		Break	0

Watch list

Expression	Value
Application	\$60f3a94

Watch Properties

Expression: Application

Repeat Count: 0 Digits: 0

☒ Enabler ☐ Allow Function Calls

Style

- ☐ Character ☐ String ☐ Decimal
☐ Hexadecimal ☐ Floating Point ☐ Pointer
☐ Record/Structure ☒ Default ☐ Memory Dump

OK

Cancel

Help

Form1

Lazarus Source Editor

```

unit1 project1.lpr
program Project1;

{$mode objfpc}{$H+}

uses
  Forms, unit1;

begin
  Application.Initialize;
  Application.CreateForm(TForm1, Form1);
  Application.Run;
end.

```

CallStack

Source	Line	Function
heap.inc	317	SYSGETMEM(12, (HEAPERRORPROC) \$4015a0
heap.inc	197	ASMGETMEM((void_pointer) \$0, 12)
except.inc	64	PUSHEXCEPTADDR(1, (PJMP_BUF) \$0, (PEXCE
application.inc	175	TAPPLICATION_INITIALIZE(^TAPPLICATION) s
project1.lpr	9	main

Locals

Name	Value
SIZE	12
AGAIN	true (64)
I	-1073744104
PBEST	(PFREERECORD) \$403f4740
PCURR	(PFREERECORD) \$408
PROC	(HEAPERRORPROC) \$4015a0ec <__DTOR_END__+4>
RESULT	(void_pointer) \$400
S1	1077157964
S	-1073744024
SIZELEFT	1077888832
SYSGETMEM	(void_pointer) \$400

Ok ... et aujourd'hui ?

- Les composants existent dans toute pile logicielle rencontrée
- Direct X ? Composants COM
- Java EE ? EJBs



DESKTOP

MOBILE

RELEASES

ADD-ONS

SUPPORT

ABOUT



Firefox Free Download

[Systems & Languages](#) | [Release Notes](#) | [Privacy](#)

Firefox FEATURES

Bringing together all kinds of awesomeness to make browsing better for you.



*Browsing
Made Easy*



*High
Performance*



*Advanced
Security*



*Powerful
Personalization*



*The Cutting
Edge*



*Universal
Access*

Top FEATURES

(rollover for details)

[Super Speed](#)

[Stay in Sync](#)

[Easy Customization](#)

[Awesome Bar](#)

[Intuitive Interface](#)

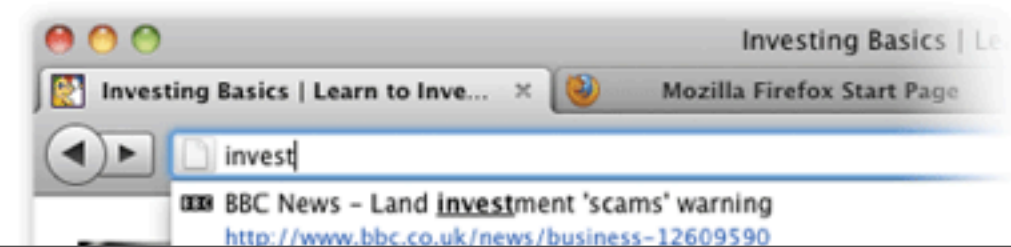
[Return to top](#)

Browsing Made EASY

Awesome Bar

Get to your favorite sites quickly – even if you don't remember the URLs. Type your term into the location bar (aka the Awesome Bar) and the autocomplete function will include possible matches from your browsing history, bookmarked sites and open tabs.

[The Awesome Bar](#) learns as you use it—over time, it adapts to your preferences and



XUL

JavaScript

XPCOM : JavaScript, C, C++, Java, Python

Filesystem

TCP

HTTP

zlib

Gecko

...

Serveurs d'applications

- Java EE, Spring, Tomcat, .Net, ...
- Les applications sont déployées sur des serveurs qui offrent un environnement pour :
 - exécuter des applications
 - les monitorer
 - les intégrer

Application type

Infra

Présentation

Bases de données

Logique métier

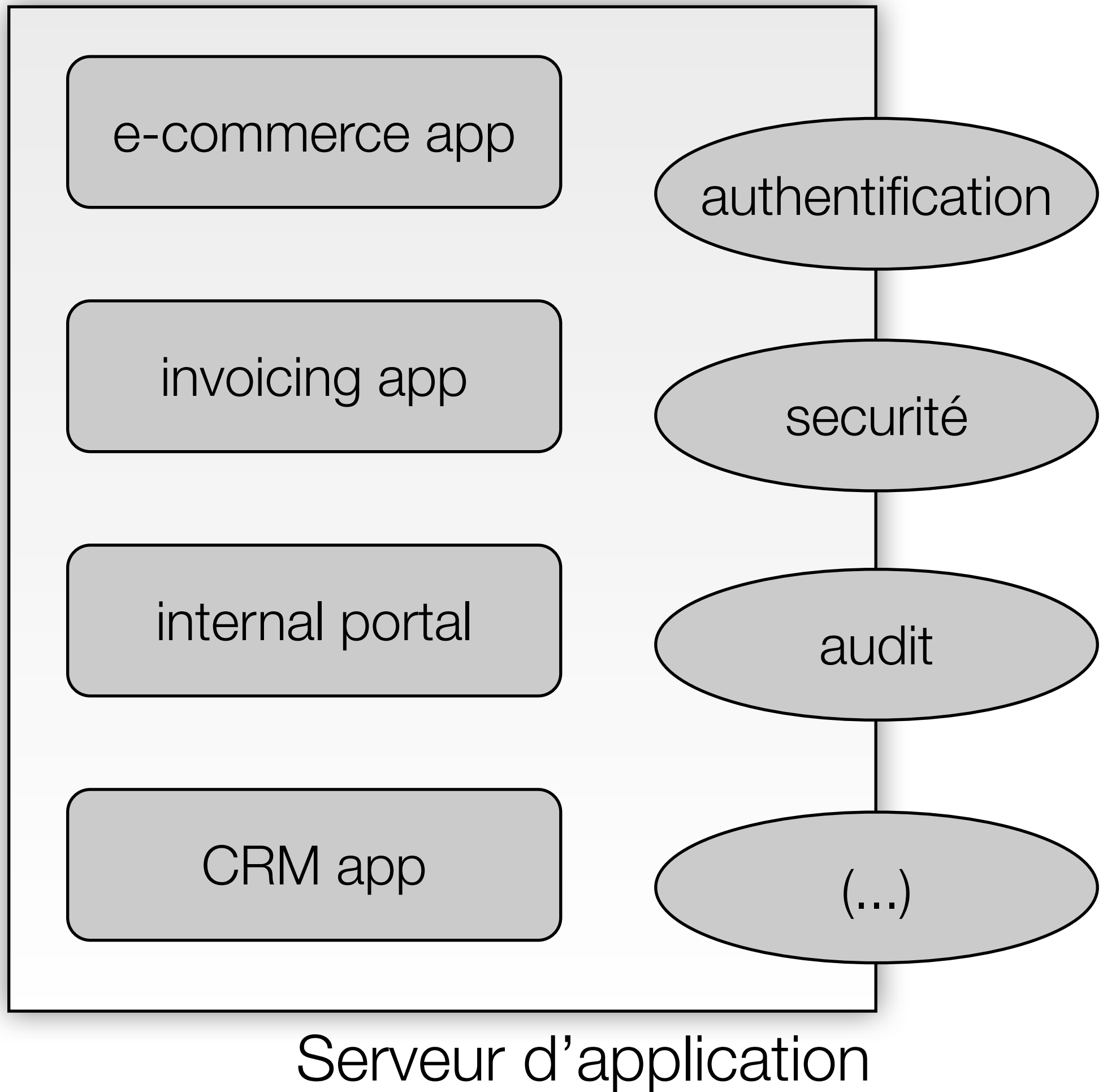
Messaging

SMTP

Données

LDAP

(...)



Le cas des serveurs Java EE

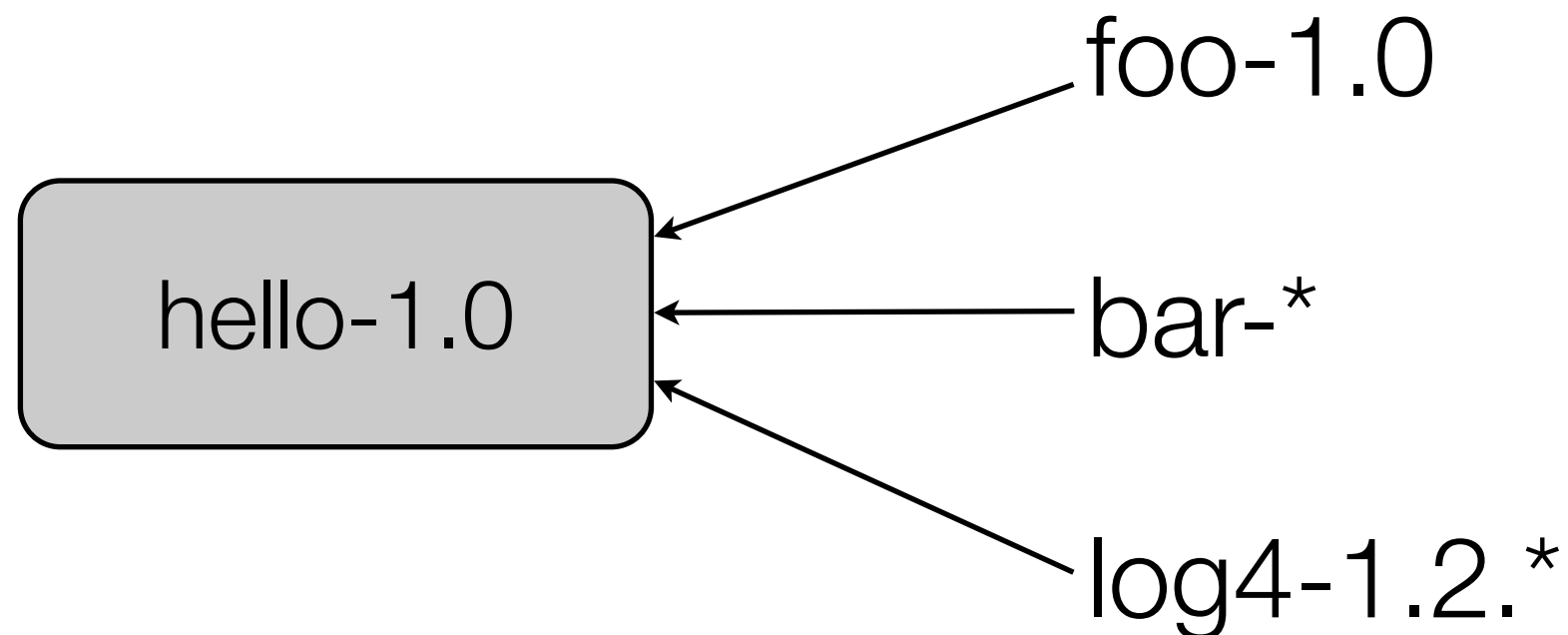
- Une spécification, de multiples implémentations
- De nombreuses briques à assembler pour implémenter les spécifications de Java EE : data, composants, transactions, web services, web, messaging, ...
- Certaines briques (opensource) sont partagées entre des serveurs concurrents
- 1 brique = 1 composant

Services d'un serveur d'applications

- Il n'y a pas que l'infrastructure qui propose des services
- Classiques :
 - logging
 - transactions
 - validation
 - (...)

Composants ++

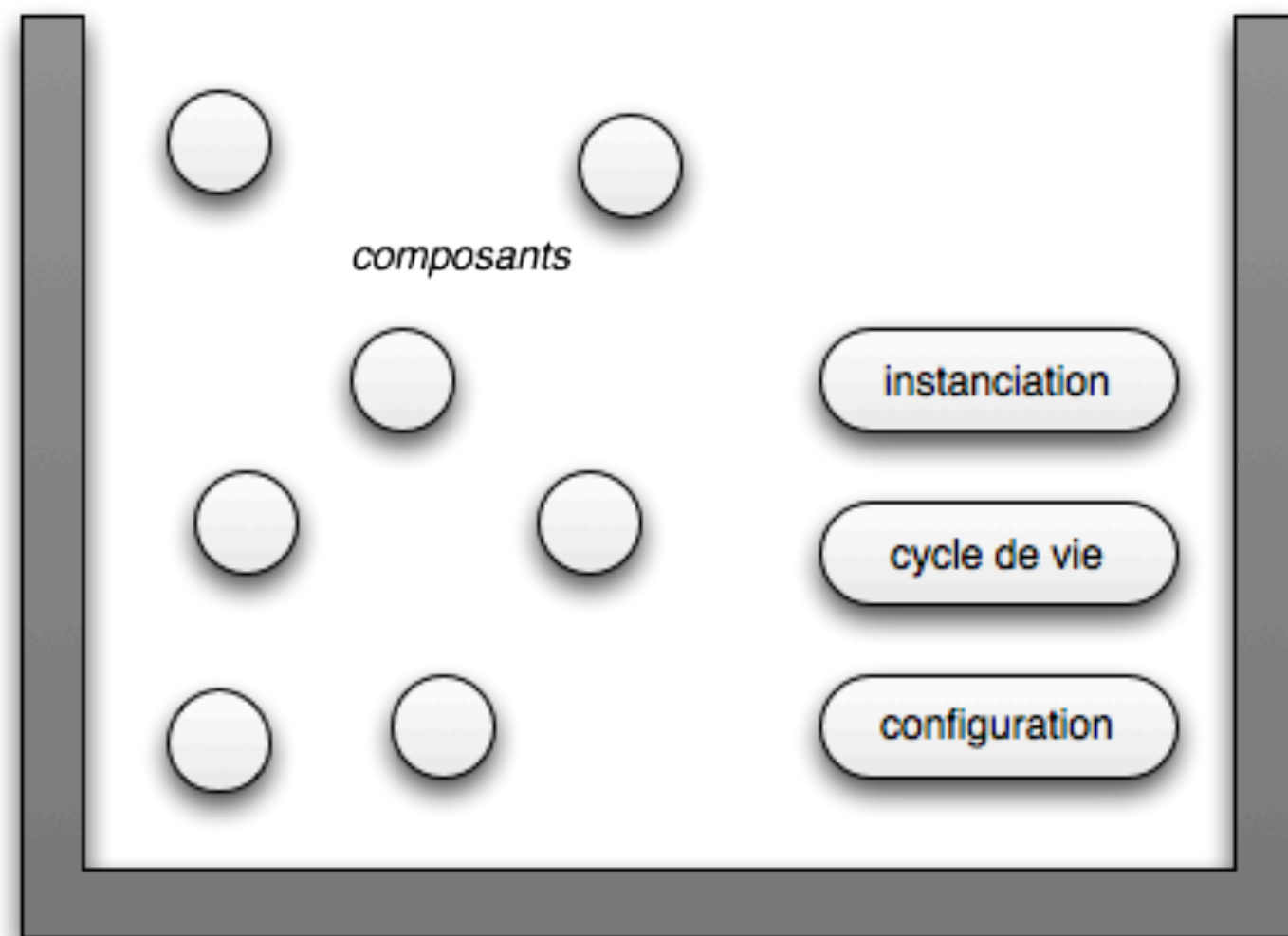
- Certaines technologies de composants ne se limitent pas à interface + implémentation
- Versioning, e.g. “1.2.0-beta1”
- Dépendances, e.g. “foo-1.0” + “bar-[1.*]”



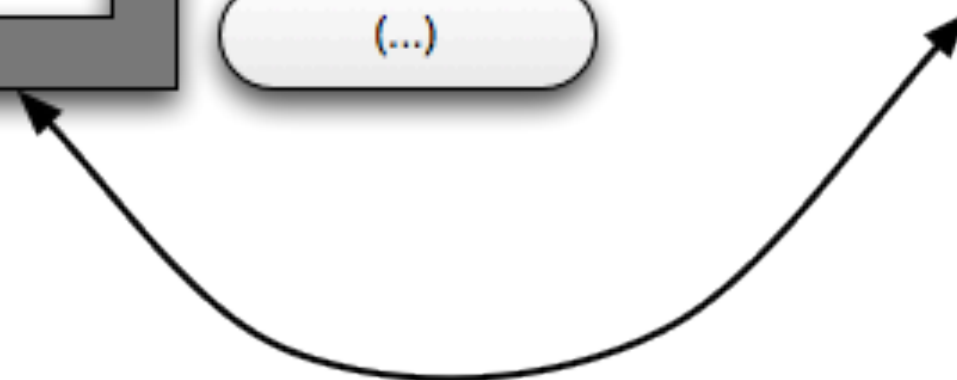
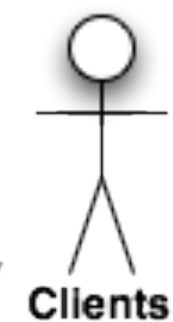
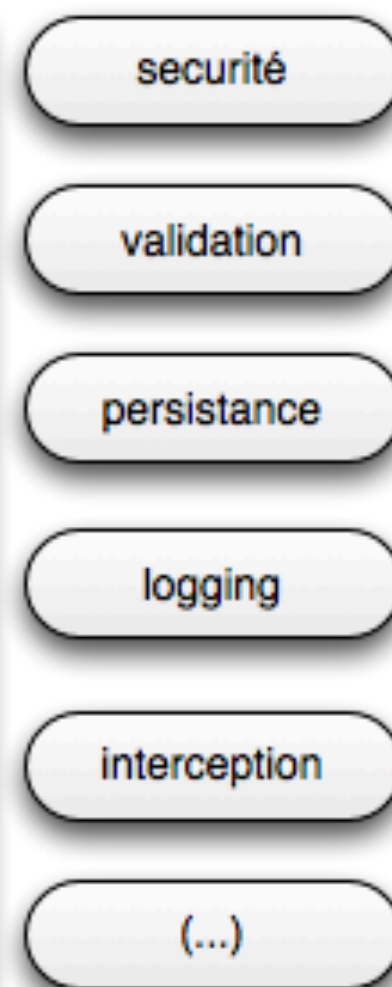
Conteneurs

- On déploie les composants dans un middleware dédié
- Notion de conteneur de composants :
 - gestion du cycle de vie (start/stop/...)
 - configuration : paramètres, services
 - dépendances : autres composants
 - abstraire la localisation

Conteneur



Services



Annuaire

- Lors du déploiement d'un composant, on le référence dans un annuaire
- L'annuaire permet d'obtenir une référence via le conteneur
- Un annuaire référence aussi les services offerts par le conteneur
- Accès uniforme pour un composant :
 - ses dépendances
 - les services dont il a besoin

Spring, Guice, ...

- Les conteneurs à injection de dépendance sont la base de nombreux frameworks industriels modernes
- Composant = interface + implémentation
- Configuration : dépendances
- Cycle de vie : supporté par certains (Spring, PicoContainer)

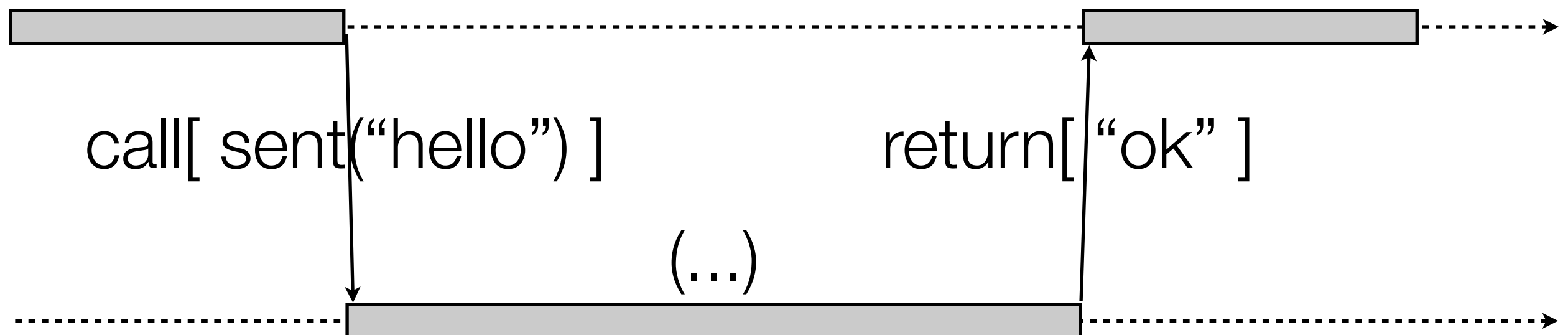
Composants distribués

Local ou distant

- Un conteneur offre accès à des composants avec une isolation de leur localisation
- Invoquer un composant distant ou local ne change pas grand chose
- C'est essentiel pour des applications d'entreprise

RPC/RMI

- Qui aime ouvrir des connexions TCP/IP ? Écrire un protocole ? Le mettre au point ?
- Masque le distribué par un appel “normal” de fonction méthode
- À la base de nombreuses communications (e.g., FaceBook, Google, ...)



Principe d'un appel RPC

- Coté client :
 - encodage des paramètres
 - connexion TCP, envoi d'un message sur un protocole ad-hoc pour demander l'appel de fonction
- Coté serveur :
 - serveur TCP avec boucle de réception
 - décodage du message d'appel et des paramètres
 - appel de la fonction
 - encodage de la valeur de retour pour un message de réponse

Thrift

Software minus logo.

Home »

[About](#)

[Mailing Lists](#)

[Developers](#)

[Download](#)

[Tutorial](#)

[Thrift Wiki](#)

Thrift is a software framework for scalable cross-language services development. It combines a software stack with a code generation engine to build services that work efficiently and seamlessly between C++, Java, Python, PHP, Ruby, Erlang, Perl, Haskell, C#, Cocoa, JavaScript, Node.js, Smalltalk, and OCaml.

Originally developed at Facebook, Thrift was open sourced in April 2007 and entered the Apache Incubator in May, 2008.

Quick Links

- [Thrift Wiki](#)
- [JIRA page](#)

Getting Started

- [Download Thrift](#)
- [Thrift Tutorial](#)
- [Mailing Lists](#)

An Example

Thrift allows you to define data types and service interfaces in a simple definition file. Taking that file as input, the compiler generates code to be used to easily build RPC clients and servers that communicate seamlessly across programming languages.

For instance, say you would like to write a service to store user objects for your web frontend. You could write a Thrift file as follows:



protobuf

Protocol Buffers - Google's data interchange format

Search projects

Project Home

[Downloads](#)

[Wiki](#)

[Issues](#)

[Source](#)

Summary

[Updates](#)

[People](#)

Project Information

★ Starred by 2219 users

[Activity](#)  High
[Project feeds](#)

Code license

[New BSD License](#)

Labels

google, ProtocolBuffers, protobuf

Members

kenton@google.com,
jas...@google.com,
tempo...@gmail.com,
liuj...@google.com,
wen...@google.com
[6 committers](#)

Featured

Downloads

[protobuf-2.4.1.tar.bz2](#)
[protobuf-2.4.1.tar.gz](#)
[protobuf-2.4.1.zip](#)
[protoc-2.4.1-win32.zip](#)
[Show all »](#)

Links

External links

[Documentation](#)

[Tutorials](#)

Protocol Buffers

What is it?

Protocol Buffers are a way of encoding structured data in an efficient yet extensible format. Google uses Protocol Buffers for almost all of its internal RPC protocols and file formats.

Latest Updates

<http://protobuf.googlecode.com/svn/trunk/CHANGES.txt>

Documentation

[Read the documentation.](#)

Discussion

[Visit the discussion group.](#)

Quick Example

You write a .proto file like this:

```
message Person {  
  required int32 id = 1;  
  required string name = 2;  
  optional string email = 3;  
}
```

Then you compile it with protoc, the protocol buffer compiler, to produce code in C++, Java, or Python.

RPC et inter-opérabilité

- RMI est un exemple de technologie RPC non-portable
- Des solutions comme Thrift (FaceBook) ou Protocol Buffers proposent des protocoles binaires avec des implémentations dans plusieurs langages
- Besoin réel : 1 seule technologie / langage est rarement utilisée pour implémenter l'ensemble d'un système d'information !
- Tous les langages ne partagent pas la même sémantique :
 - appel
 - types de données
 - “null”

Les IDL

- Interface definition language
- Définissent une interface de manière abstraite
- Définissent les types de données
- Permettent de générer des adaptateurs pour chaque techno / langage
- Exemple : XPCOM de Mozilla utilise OMG IDL pour décrire chaque composant et le rendre disponible pour les autres composants, potentiellement implémentés dans un autre langage

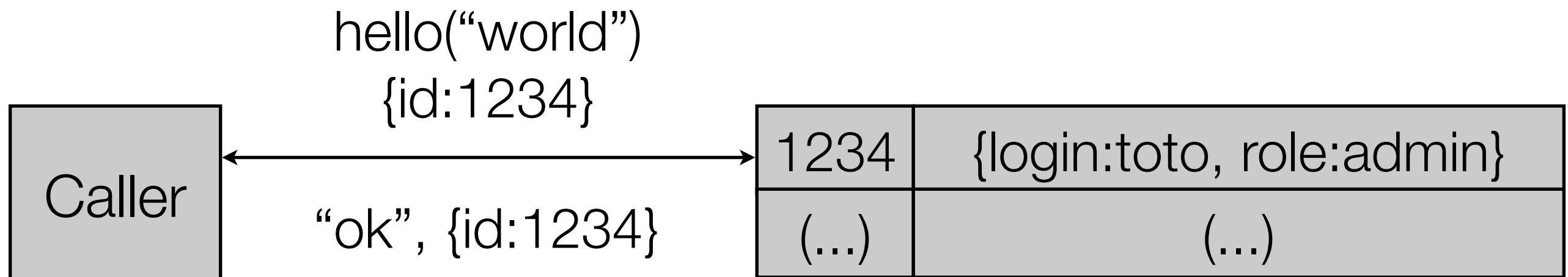
```
struct UserProfile {  
    1: i32 uid,  
    2: string name,  
    3: string blurb  
}
```

```
service UserStorage {  
    void store(1: UserProfile user),  
    UserProfile retrieve(1: i32 uid)  
}
```

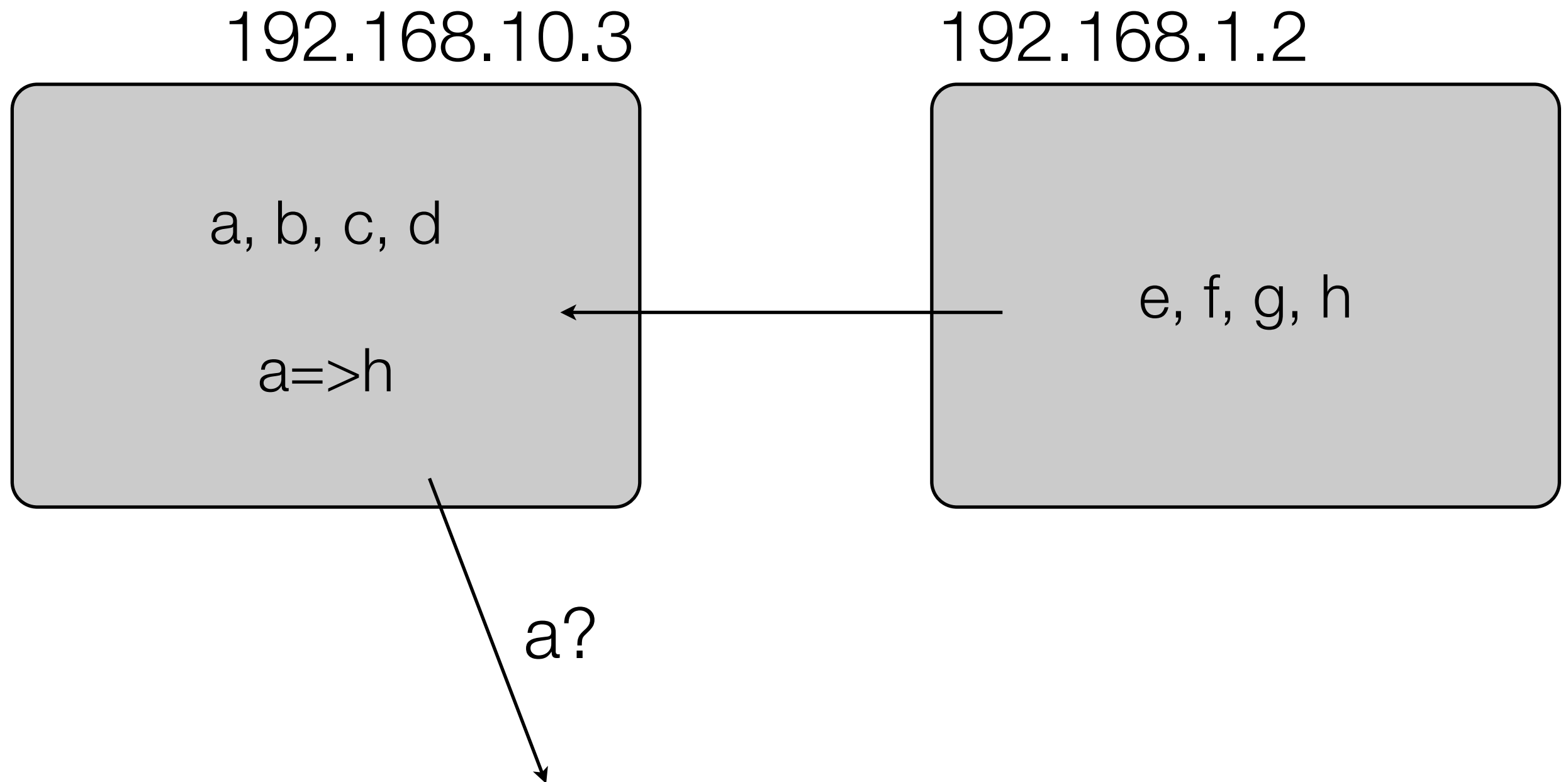
<http://thrift.apache.org/>

Stateful vs Stateless coté serveur

- Analogie : panier d'achat *versus* calculatrice sans mémoire
- Comment garder les informations de contexte pour chaque client ?
 - 1 connexion TCP persistante / client => ne passe pas à l'échelle
 - 1 identifiant de "session" ... comme les sessions web !
- Penser à l'expiration ...



Conteneurs à composants distribués



Modèles de programmation

Approche historique

- 1 technologie de composants = 1 API à utiliser de façon très explicite
- Modèle de programmation lié à la technologie
- Exemple parlant : les EJB < 3

```
package fibo;
```

```
import javax.ejb.EJBObject;
```

```
import java.rmi.RemoteException;
```

```
public interface FibonacciRemote extends EJBObject {  
    public int get(int n) throws RemoteException;  
}
```

```
package fibo;
```

```
import javax.ejb.EJBHome;
```

```
import java.rmi.CreateException;
```

```
import java.rmi.RemoteException;
```

```
public interface FibonacciHome extends EJBHome {  
    public FibonacciRemote create()  
        throws CreateException, RemoteException;  
}
```

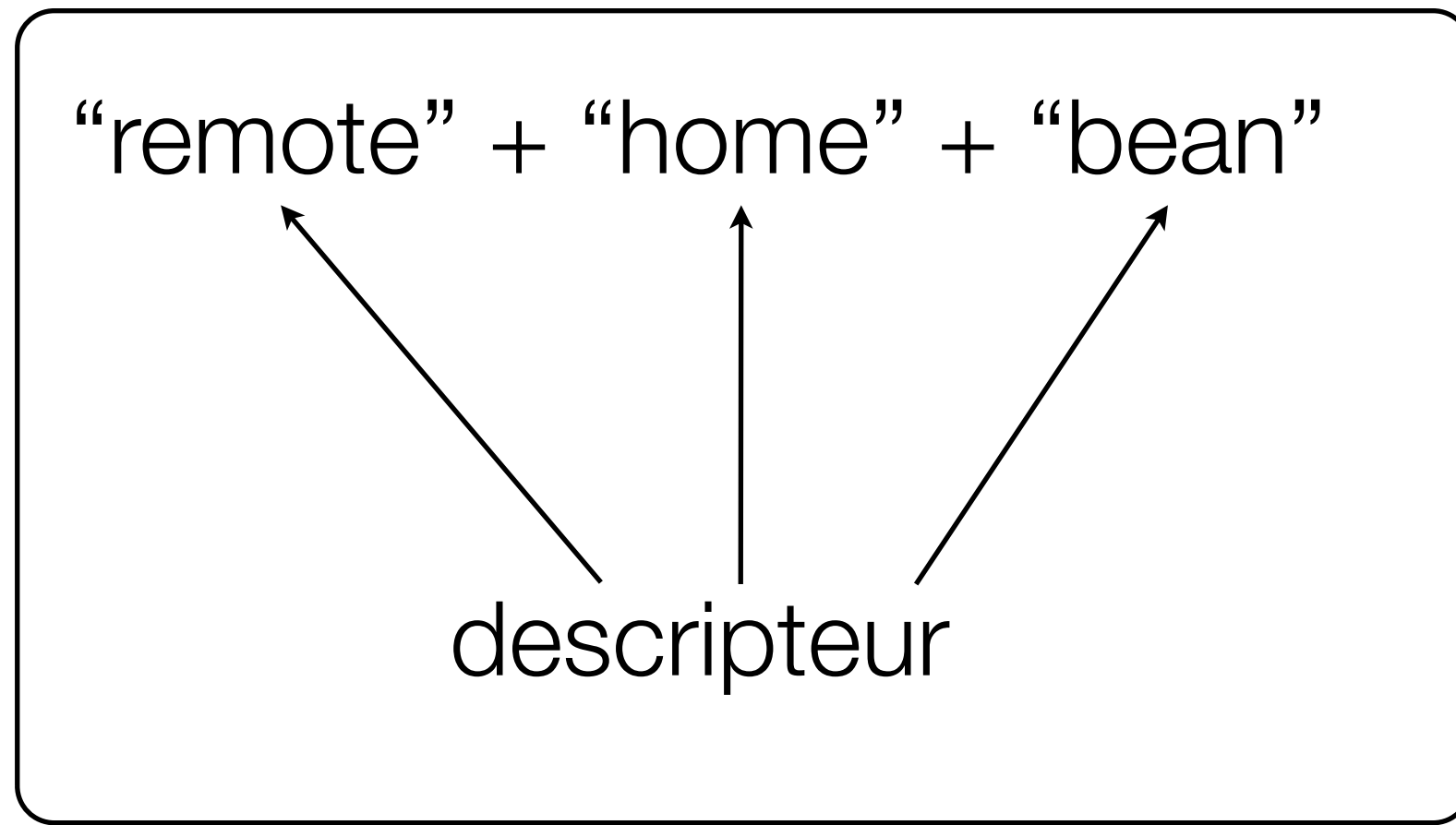
```
package fibo;

import java.rmi.*;
import javax.ejb.*;

public class FibonacciBean implements SessionBean {
    public void ejbPassivate() {}
    public void ejbActivate() {}
    public void ejbRemove() {}
    public void setSessionContext(SessionContext context) {}
    public void ejbCreate() {}

    public int get(int n) {
        // LE CODE !!!
    }
}
```

```
<?xml version="1.0"?>
<!DOCTYPE ejb-jar PUBLIC
    "-//Sun Microsystems, Inc.//DTD Enterprise JavaBeans 2.0//EN"
    "http://java.sun.com/dtd/ejb-jar_2_0.dtd">
<ejb-jar>
  <enterprise-beans>
    <session>
      <ejb-name>fibonacci</ejb-name>
      <home>fibo.FibonacciHome</home>
      <remote>fibo.FibonacciRemote</remote>
      <ejb-class>fibo.FibonacciBean</ejb-class>
      <session-type>Stateless</session-type>
      <transaction-type>Container</transaction-type>
    </session>
  </enterprise-beans>
</ejb-jar>
```

Composant distribué

+ conventions, + ...

Annuaire

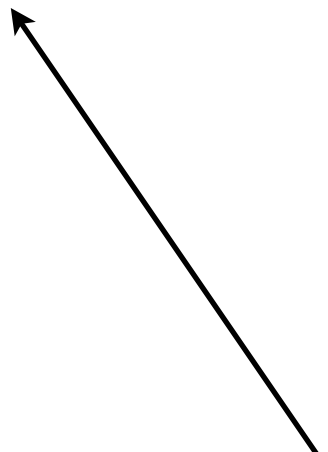
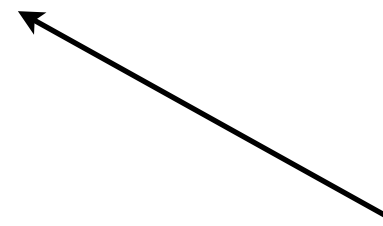
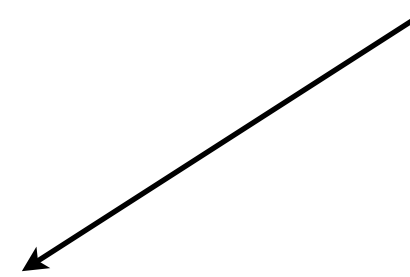
```
Context ctx = new InitialContext();  
FibonacciHome home = (FibonacciHome) ctx.lookup(  
    "java:comp/env/ejb/fibo/FibonacciHome"  
);
```

```
FibonacciRemote fibo = home.create();
```

```
System.out.println(fibo.get(10));
```

Référence

Usage



```
package fibo;
```

```
import javax.ejb.Remote;
```

```
@Remote
```

```
public interface Fibonacci {  
    public int get(int n);  
}
```

EJB 3 ...

```
package fibo;
```

```
import javax.ejb.Stateless;
```

```
@Stateless
```

```
public class FibonacciImpl implements Fibonacci {  
    public int get(int n) {  
        // Code  
    }  
}
```

```
package fibo;

import javax.ejb.Stateless;
import javax.annotations.PostConstruct;

@Stateless
public class FibonacciImpl implements Fibonacci {

    public int get(int n) {
        // Code
    }

    @PostConstruct
    public void jeSuisPret() {
        System.out.println("Youpie !");
    } // replace ejbCreate()

}
```

Cycle de vie optionnel ...

Plus d'appel à l'annuaire

```
package fibo;
```

```
import javax.ejb.Stateless;
```

```
import javax.inject.Inject;
```

```
@Stateless
```

```
public class SomeComponent implements PlopComponent {
```

```
    @Inject
```

```
    Fibonacci fibo;
```

```
    public void work() {
```

```
        System.out.println(fibo.get(10));
```

```
    }
```

```
}
```

Simplification des modèles de programmation

- Tendence de fond de l'industrie du middleware
- Écrire du code métier le plus détaché possible de frameworks et APIs : POJO (*plain old Java objects*) et de considérations non-fonctionnelles
- Favoriser les conventions sur la configuration
- Approches déclaratives plutôt qu'explicites :
 - on spécifie le paramétrage de sécurité, transaction
 - on déclare ses dépendances

2 façons d'aborder le problème

- “C’est facile, j’utilise sans chercher à comprendre”
- “J’aimerais comprendre ce qui se passe derrière histoire d’être moins stupide en cas de problèmes”


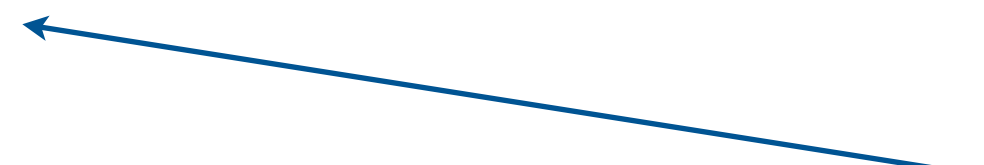
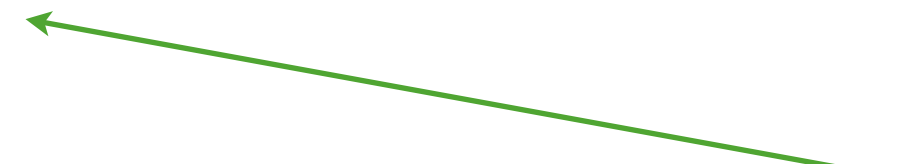

Magie = DI + AOP

- Le code que vous compilez est réellement déployé tel quel sans additifs
- C'est au moment du chargement qu'il subit de profondes transformations
- Injection de dépendances : ne plus faire d'appels à l'annuaire, principe d'inversion de contrôle auprès du conteneur qui réalise l'assemblage de classes
- Aspects : greffe du code non-fonctionnel en fonction des déclarations et conventions par défaut

Bilan

- La programmation par composants est partout, sous des formes plus ou moins élaborées
- Middleware à composants = conteneur
- Les modèles de programmation tendent à se simplifier : il faut pour cela que les middlewares absorbent la complexité en faisant du travail auparavant confié aux développeurs

4 prochains TD/TP : approche

- Vous allez implémenter des versions simplistes de middlewares courants :
 - conteneur à injection de dépendances  minimum !
 - composants distants  ok
 - cache distribué 
 - messagerie fiable  0 soucis pour l'examen
- **But 1 : éviter toute sensation de magie, les technologies utilisées dans l'industrie sont à peine plus sophistiquées que les implémentations que vous ferez en TD/TP ...**
- **But 2 : vous rendre adaptable à la technologie X, Y ou Z**