

Modular Java EE
in the cloud





Paul Bakker

Architect at Luminis Technologies

 @pbakker



Trend

- Applications tend to grow bigger and more complex
- Agile development and refactoring have become more common

This leads to a number of challenges :

Dependency
management

versioning

Maintenance
(long term)

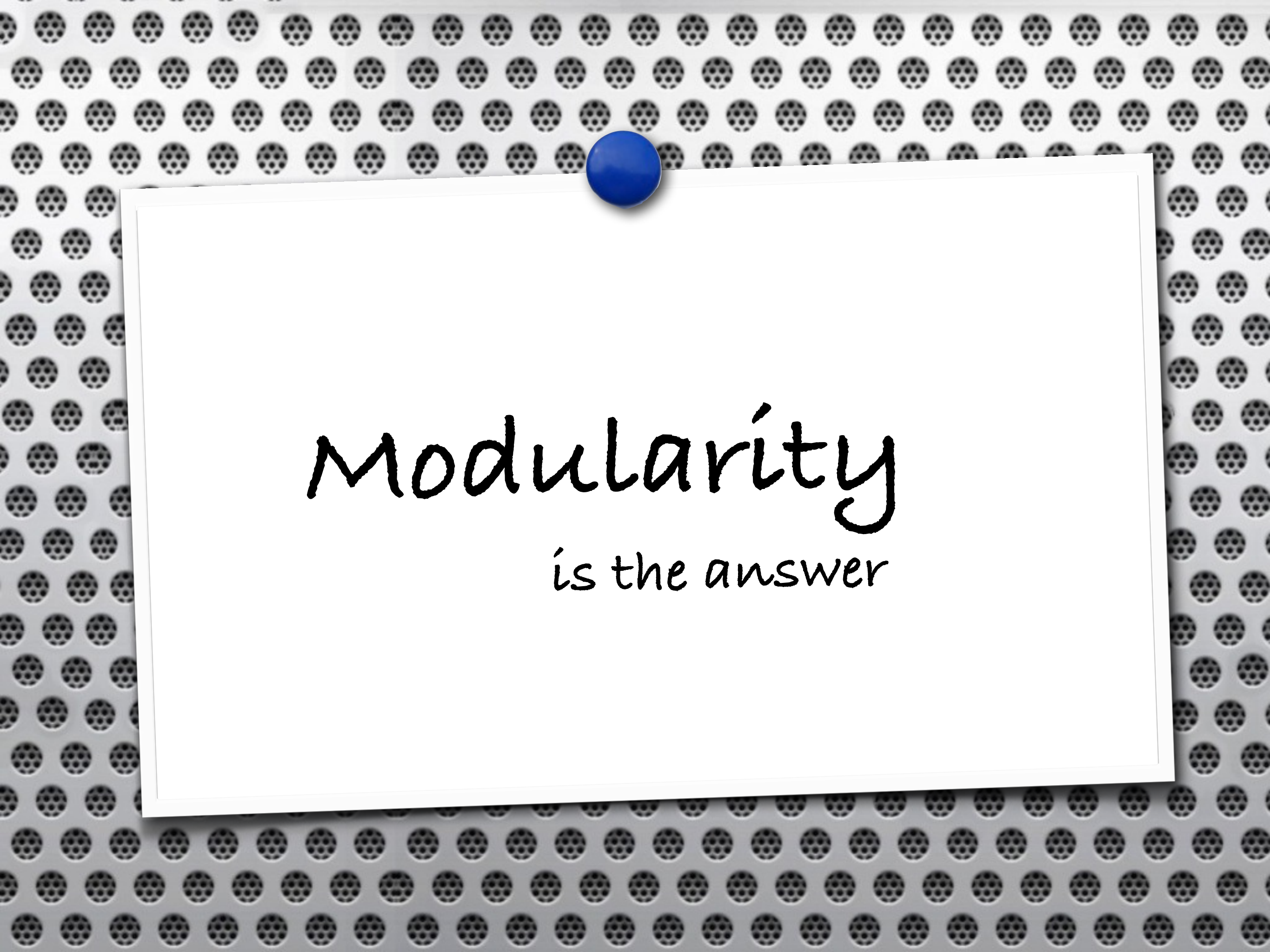
Deployment

Apps are moving to the cloud


Cloud challenges require non-trivial non-functional requirements

- Zero-downtime deployments
- Modular deployments
- Customer specific extensions (SaaS)

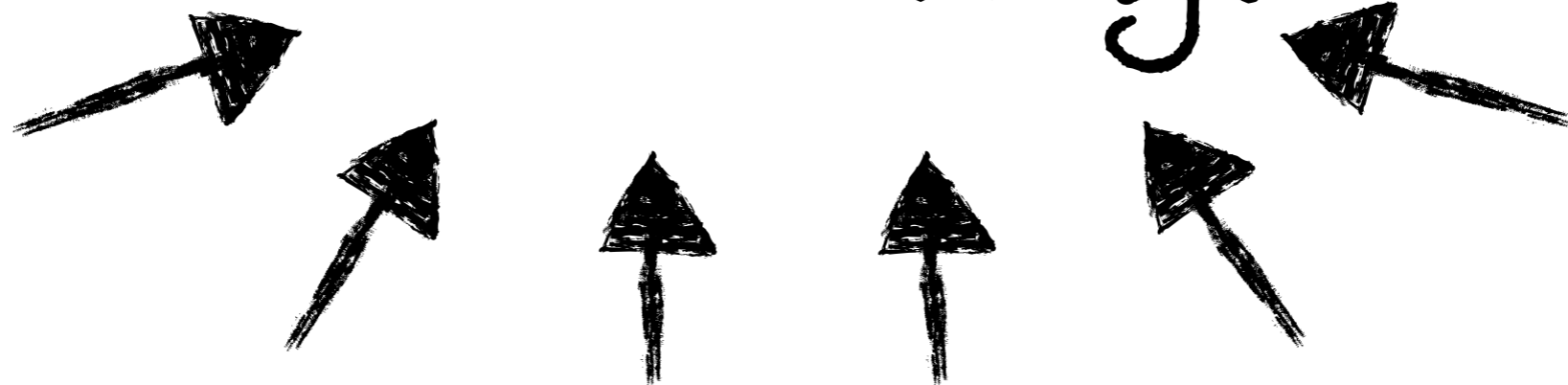




Modularity
is the answer



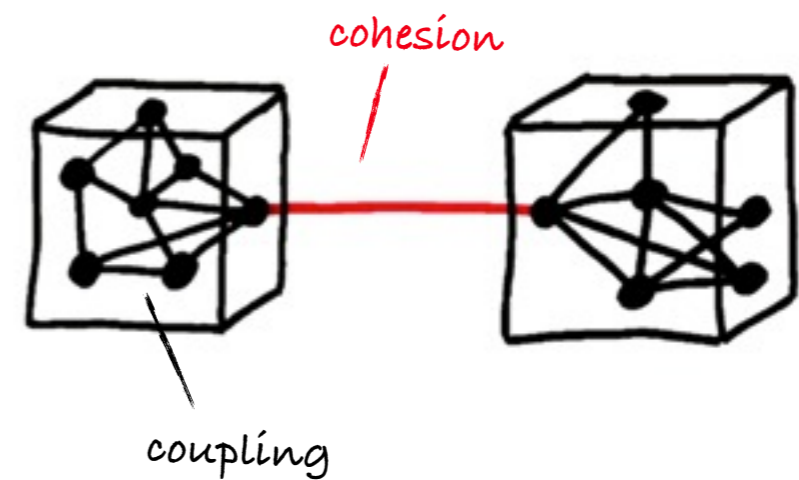
But... what exactly is
modularity?




What we learned about OO design in university :


Promote
cohesion

Prevent
(tight)
coupling



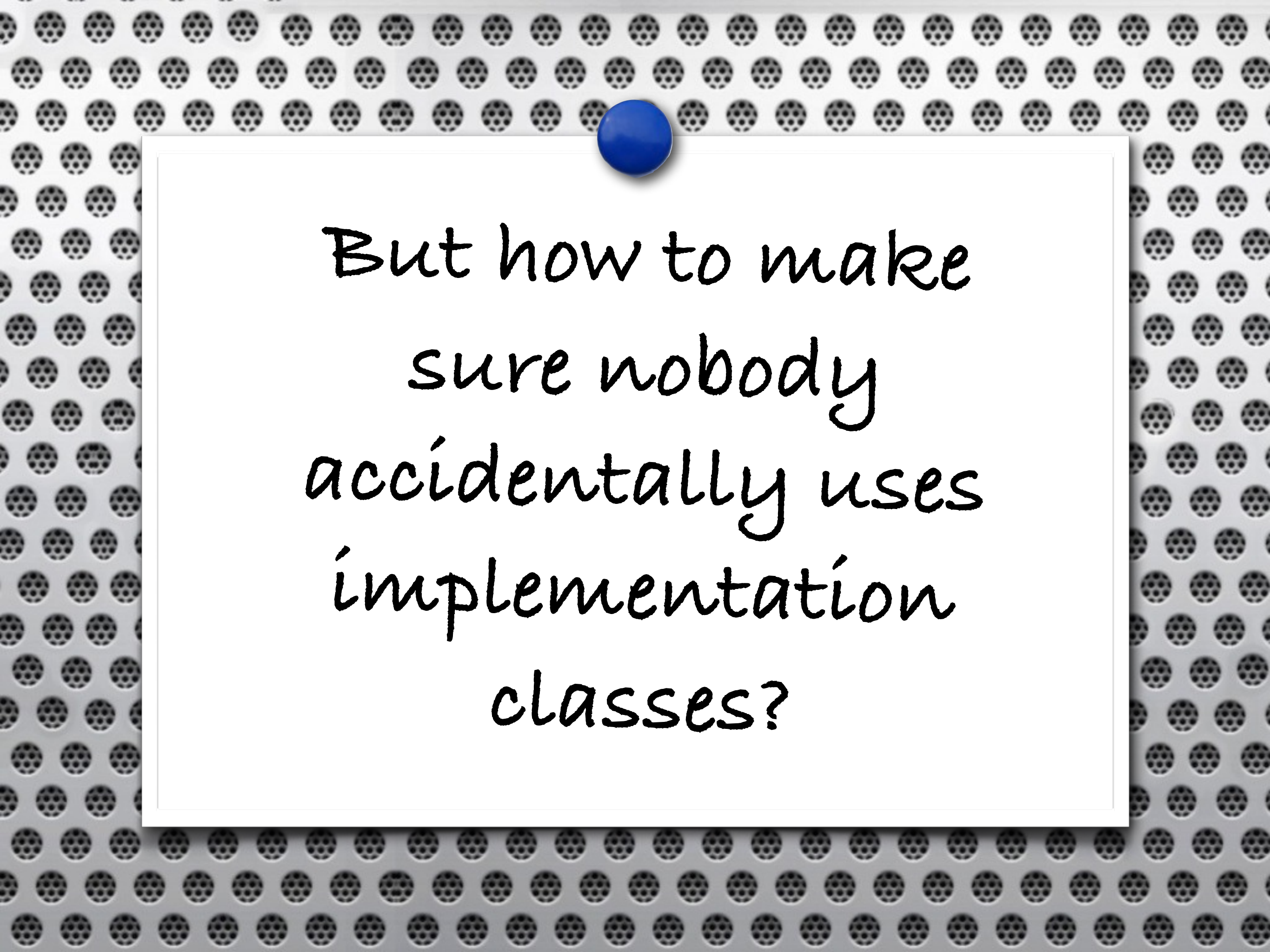


How to prevent
coupling in a code
base?



How to prevent
coupling in a code
base?

interfaces



But how to make
sure nobody
accidentally uses
implementation
classes?



imp

lemmen

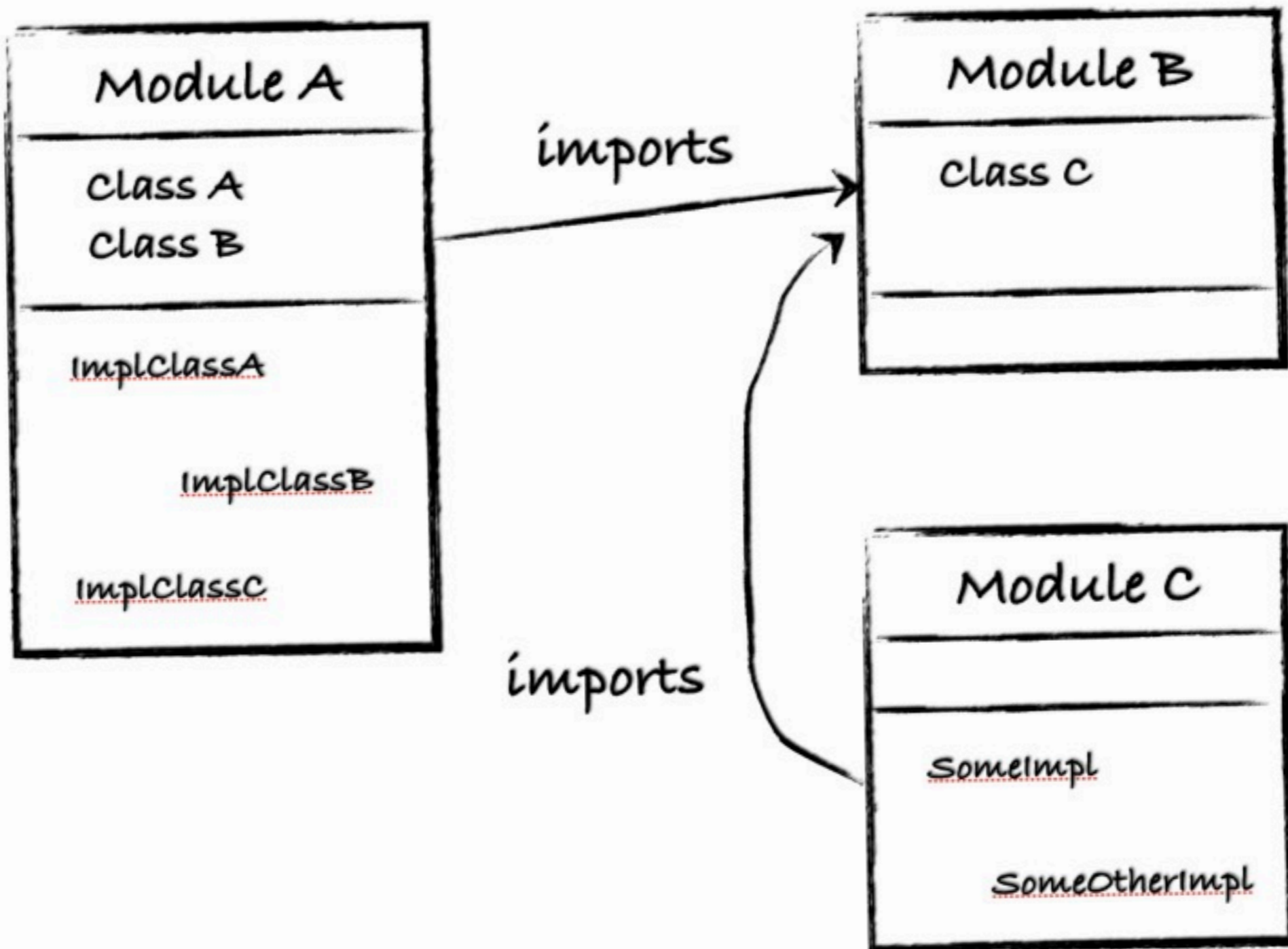
tation

hi

di

ng

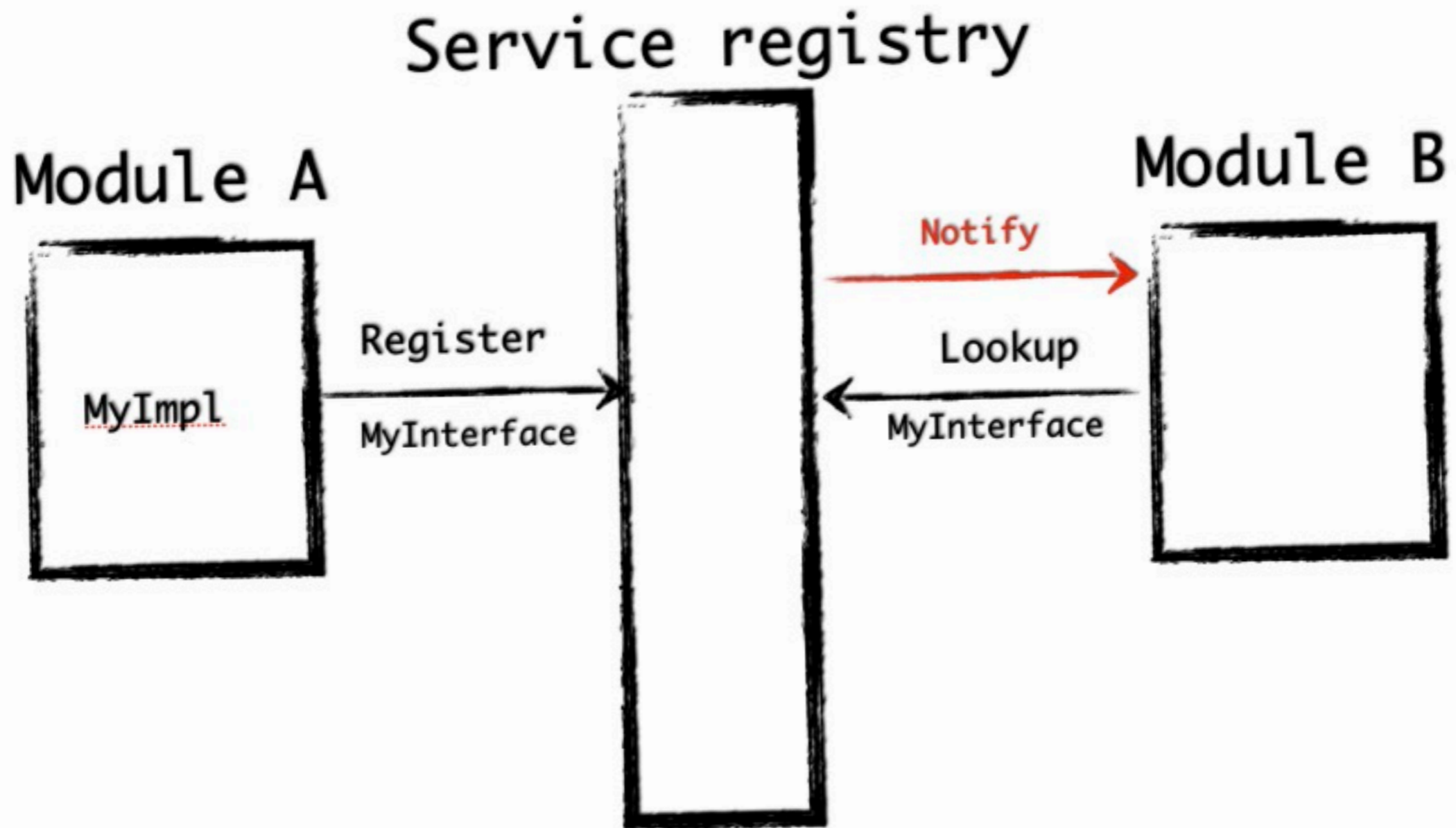
Modules




Ok, but how to create an instance of a hidden class?

```
MyInterface myI = new  
MyImplementation();
```


Services





Design time modularity

- Divide and conquer
- Forces separation of concerns
- Prevents spaghetti



The jar file is the unit of
runtime modularity
on the java platform

what about
classpath
hell?

How to deal with

Runtime dynamics

- Module versioning
- updating single modules
 - Intra-module dependency management

What do we need?

design →
consequences


Architectural
focus on
modularity

Runtime
dynamic
module
framework

High-level
enterprise APIs

← Let's not
reinvent the
wheel


↑
Right now,
OSGi is the
only option



OSGi is the de-facto
standard module
system for Java

OSGi != modularity

What about Jigsaw?




OSGI is the de-facto
standard module
system for Java

remember!

Modularity is
an architectural
principle

What about Jigsaw?



OSGI is the de-facto
standard module
system for Java

remember!
Modularity is
an architectural
principle

It does not come
for free with a
framework

at abo



Demo

OSGI vs. JAVA EE



OSGi vs. Java EE : opposite ends?

Java EE:
high level
APIs

OSGi:
low-level, mix
and match
yourself

OSGi vs. Java EE : opposite ends?

Java EE:
high level
APIs

They don't
have to be
either / or

OSGi:
low-level, mix
and match
yourself

How? - 3 options

OSGi as core
architecture with
EE APIs published
as services

Hybrid solution
using an injection
bridge

OSGi a la carte



Option 1 - OSGi in Java EE appservers

OSGi bundles as
your core
programming
model

Use capabilities of
your app server
(clustering, data
sources...)

Java EE APIs
available from
within bundles

The (near) future!

Web Application Bundle (WAB)

Enterprise
OSGi
standard

WAR with a manifest

Easy registration of Servlets
and JAX-RS components

Web Application Bundle (WAB)

mywab.jar

```
META-INF/manifest.mf
index.html
WEB-INF/classes/MyResource.class
WEB-INF/faces-config.xml
WEB-INF/web.xml
```

/myweb/index.html

/myweb/agenda

manifest.mf

```
Bundle-Name: agenda.web.ui
Web-ContextPath: /myweb
Import-Package: javax.servlet
```

```
@Path("agenda")
public class MyResource {
    @GET
    public String demo() {
        return "hello";
    }
}
```

EJB Bundles

Not a standard yet!

Use transactions and JPA
like you normally would

Publish EJBs
as OSGi
services



EJB bundles

```
@Local(AgendaStorage.class)
@Stateless
public class EjbAgenda implements AgendaStorage {

    @PersistenceContext EntityManager em;

    //...
```

manifest.mf

Bundle-Name: agenda.storage.ejb

Export-EJB:ALL


OSGi CDI integration

RFP 146

Prototyped in
WELD

Use CDI
within
bundles

Use CDI to
produce /
consume
services



How are the app servers doing?

App server	Deploy bundles	WAB support	EJB/JPA support
Glassfish	✓	✓	✓
WebSphere	✓	✓	✓
WebLogic	✓	✓	✓
JBoss AS	✓	✓	✓

Option 2 : Hybrid solution

Best of both worlds?

- Your app is effectively split into two parts:
 - 'Administrative' enterprise part -> Java EE
 - 'Dynamic' part where services can be replaced without downtime -> OSGi
- Glue the two together using an 'injection bridge'

Example

```
@Component
public class HelloServiceImpl implements HelloService, Serializable {
    @ServiceDependency
    private volatile LogService logger;

    @Override
    public String hello(String name) {
        logger.log(LogService.LOG_INFO, "Hello " + name + ".");
        return "Hello, " + name + ".";
    }
}
```

```
@WebServlet(urlPatterns={"/worker"})
public class WorkerServlet extends HttpServlet {
    @Resource BundleContext bundleContext;

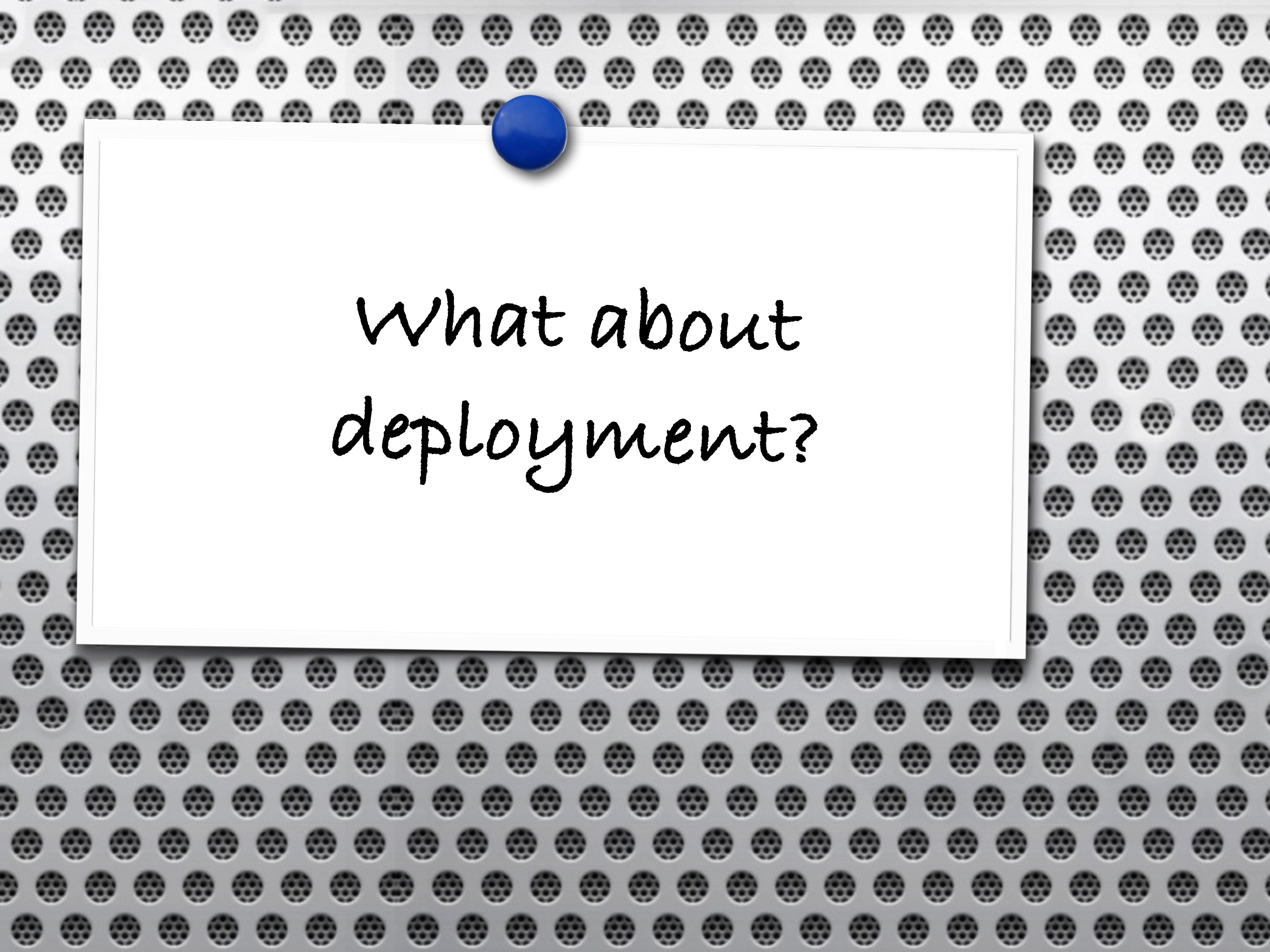
    public WorkerServlet() {
        super();
    }

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        PrintWriter out = response.getWriter();


        ServiceReference serviceReference = bundleContext.getServiceReference(HelloService.class.getName());
        if (serviceReference != null) {
            HelloService helloService = (HelloService) bundleContext.getService(serviceReference);
            out.println(helloService.hello("World"));
        }
    }
}
```

Option 3 - OSGi a la carte

Demo

A blue pushpin is pinned to the top center of a white rectangular note. The note is placed on a background of a perforated metal surface, which consists of a grid of small circular holes. The text on the note is written in a casual, handwritten style.

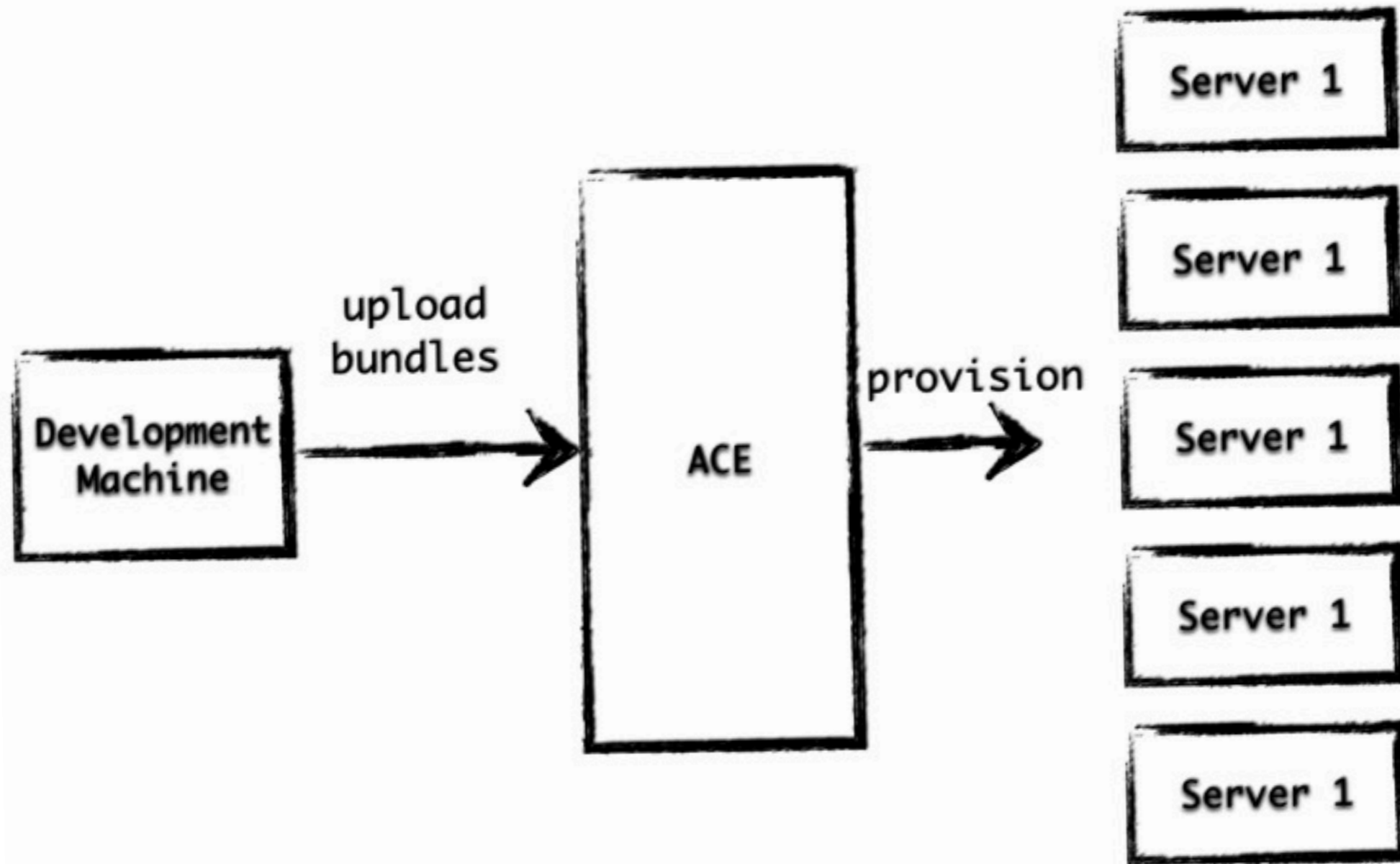
What about
deployment?



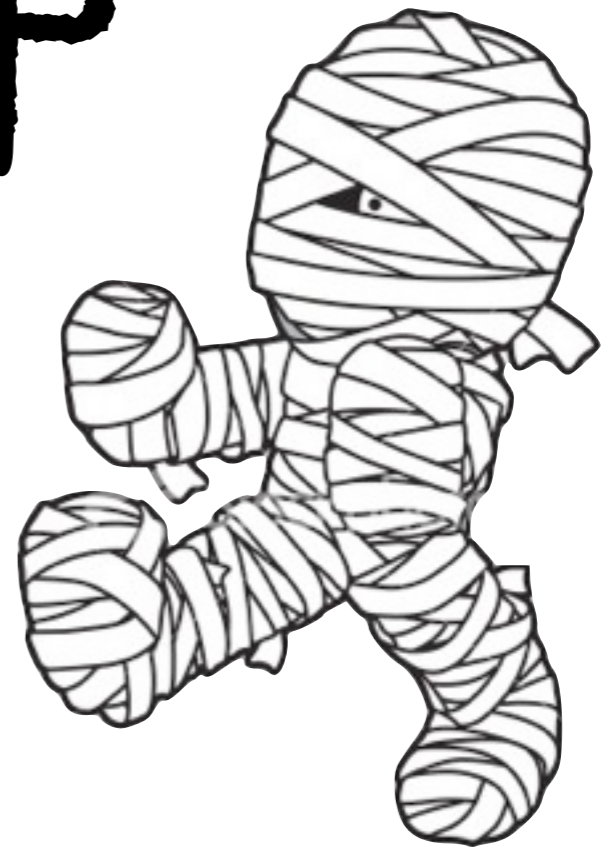
What about
deployment?

Apache
Ace
- demo -

Apache ACE



wrap up

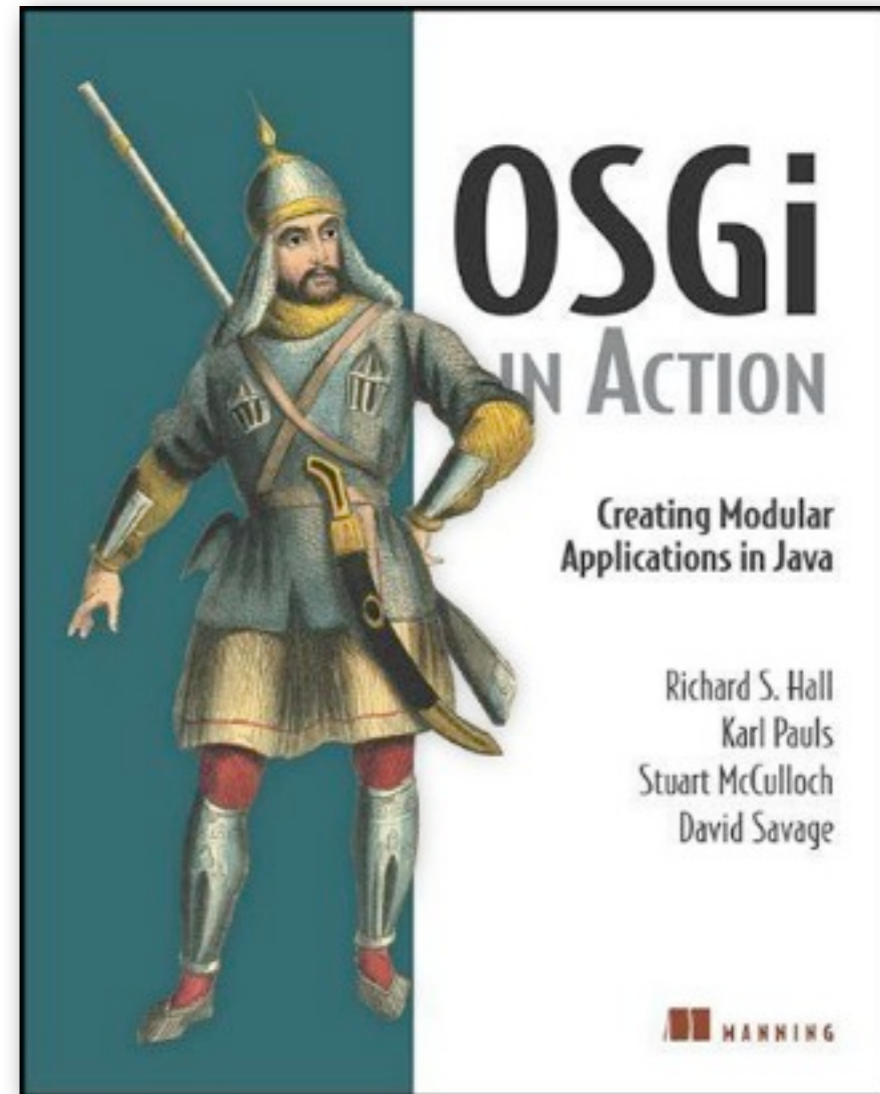
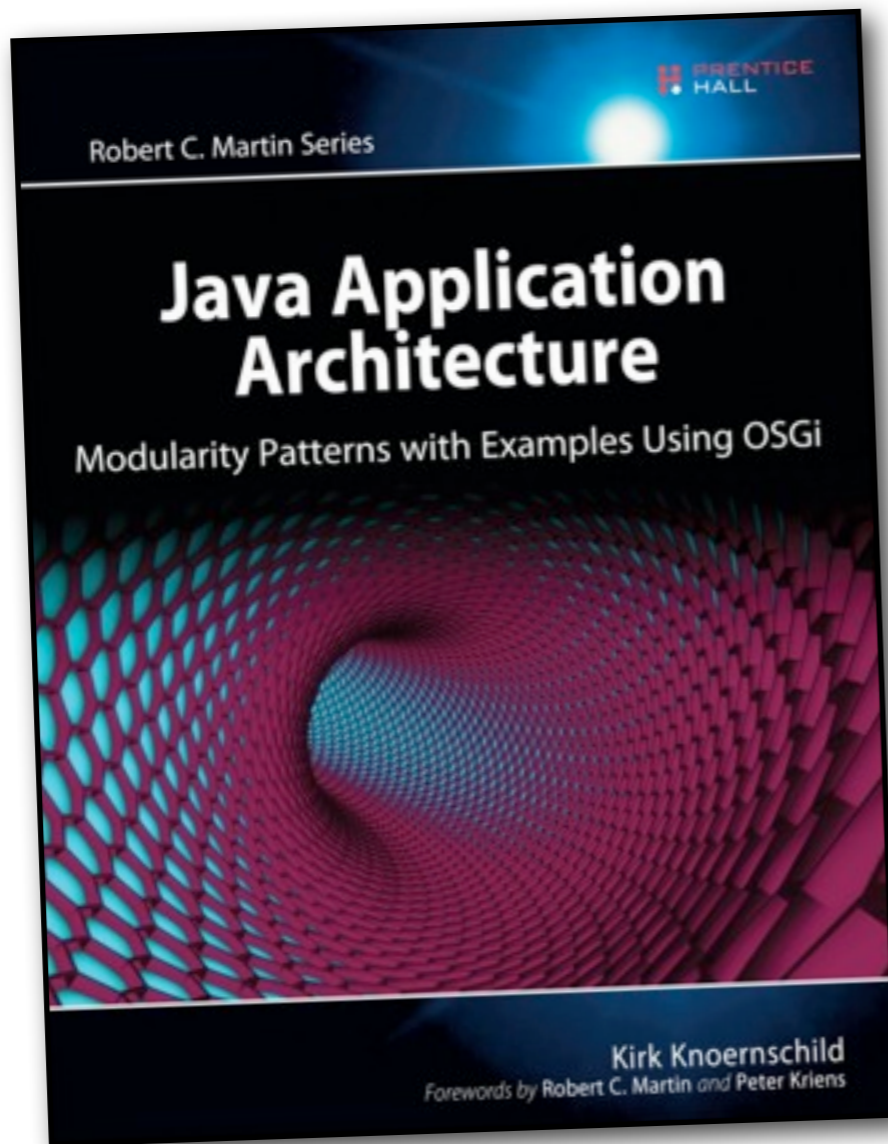




What have we learned?

- You've seen the future of enterprise development ;-)
- Why modularity is important (in the cloud)
- Practical solution for doing modularity now

Recommended reading



contact me:



Paul Bakker

paul.bakker@luminis.eu



@pbakker