THE
DEVELOPER'S
CONFERENCE

Arquitetura Java

Design patterns e Tecnologias para Modularização em Java

@filipeportes

Globalcode

Mim tarzan...

THE DEVELOPER'S CONFERENCE

- Graduado em Ciência da Computação
- Um dos Coordenadores do Grupo de Usuários Java de Goiás - @gojava
- Mais de 7 anos de experiência com desenvolvimento e arquitetura Java Web
- Atualmente trabalha para a Cast Tecnologia em Brasília – DF
- Sósia oficial do Brad Pitt

Globalcode

THE DEVELOPER'S CONFERENCE

Globalcode

THE DEVELOPER'S CONFERENCE

Globalcode

Modularização

THE DEVELOPER'S CONFERENCE

- Definindo um módulo:

"A Software Module is a **deployable, manageable, natively reusable, composable, stateless** unit of software that **provides a concise interface** to consumers"

  - Instalável
  - Gerenciável
  - Reutilizável
  - Combinável
  - Não guarda estado
  - Oferece uma Interface clara

Globalcode

Fonte: Java Application Architecture: Modularity Patterns with Examples Using OSGi

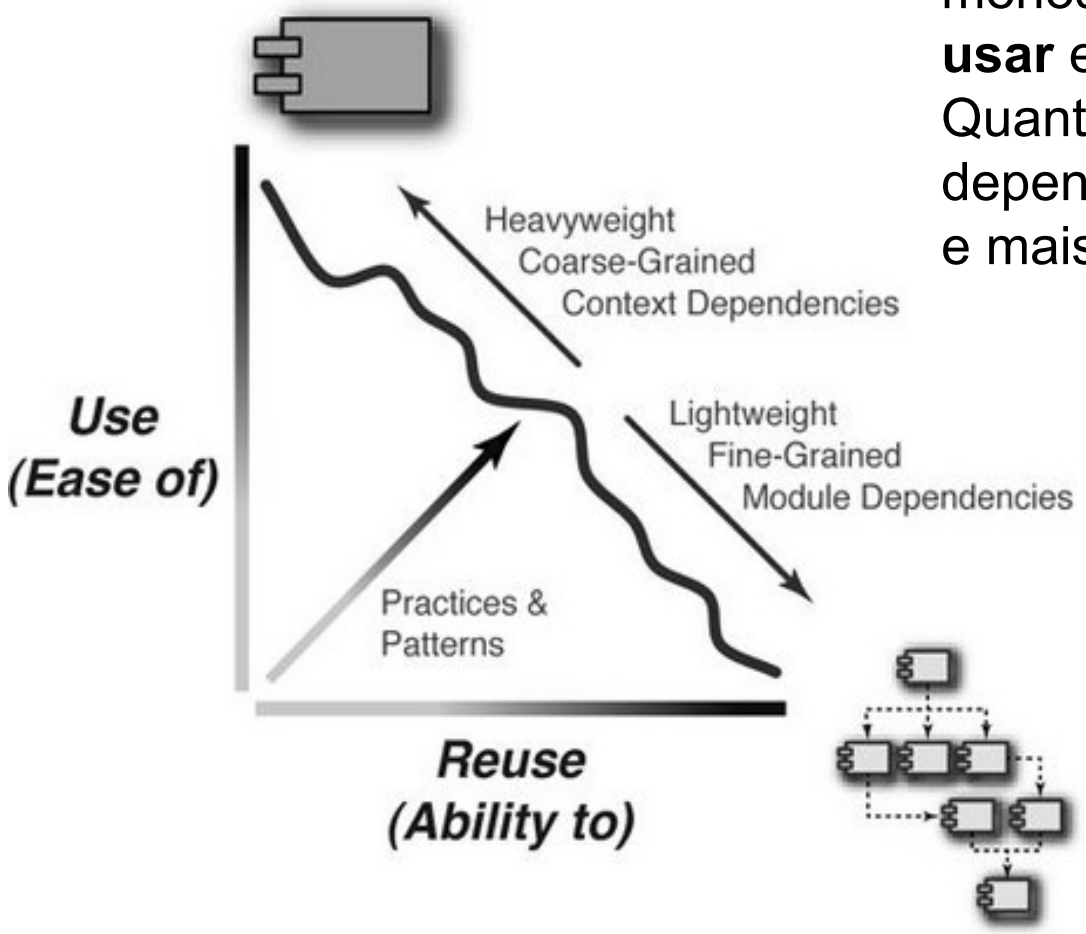
Facetas da Modularização

THE DEVELOPER'S CONFERENCE

- Modelo de Desenvolvimento
  - Formas de construir arquiteturas modulares, e tratar os problemas comuns nesse cenário.
  - Design Patterns
- Modelo de Execução
  - Foca em como gerenciador sistemas modulares em tempo de execução, ou seja, Plataformas que suportem um Eco-sistema Modular oferecendo recursos que facilitem e potencializem a modularização
  - OSGi, jigsaw, php synfony, CDI Extensions, etc...

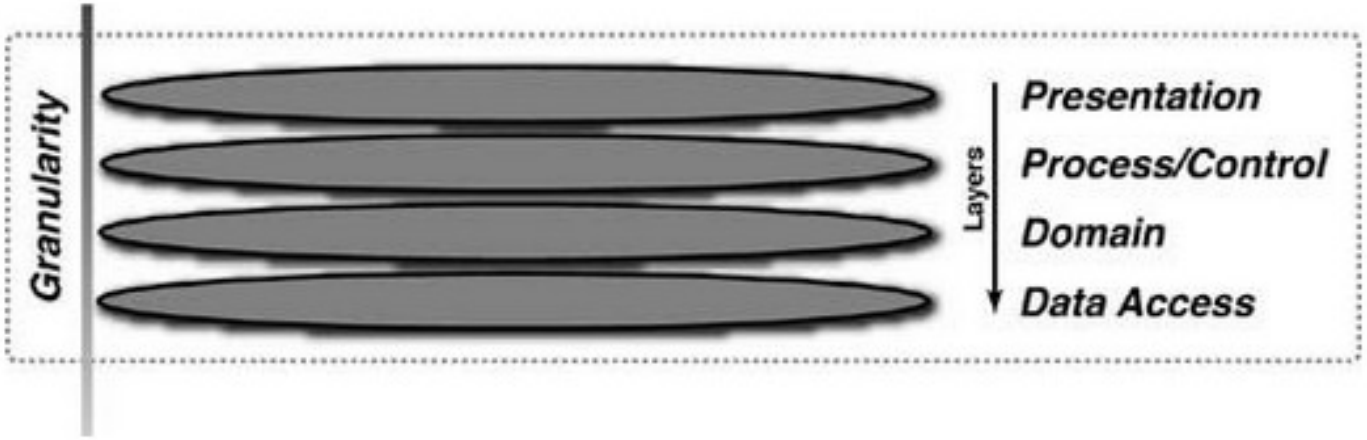
Globalcode

Uso vs Reuso



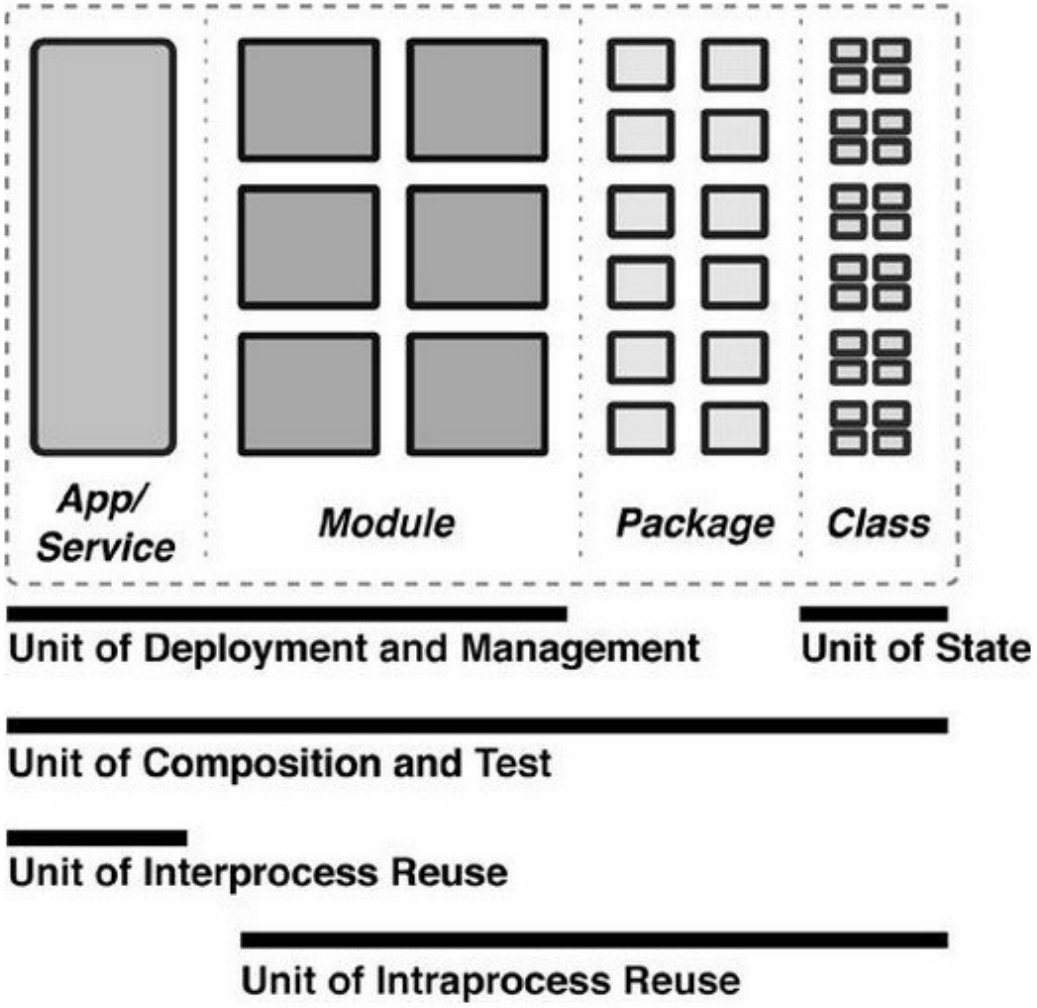
Quanto **maior a granularidade**, menos dependências, mais **fácil de usar** e mais **difícil de reutilizar**.
Quanto **menor a granularidade**, mais dependências, mais **fácil de reutilizar** e mais **difícil de usar**.

Design em camadas

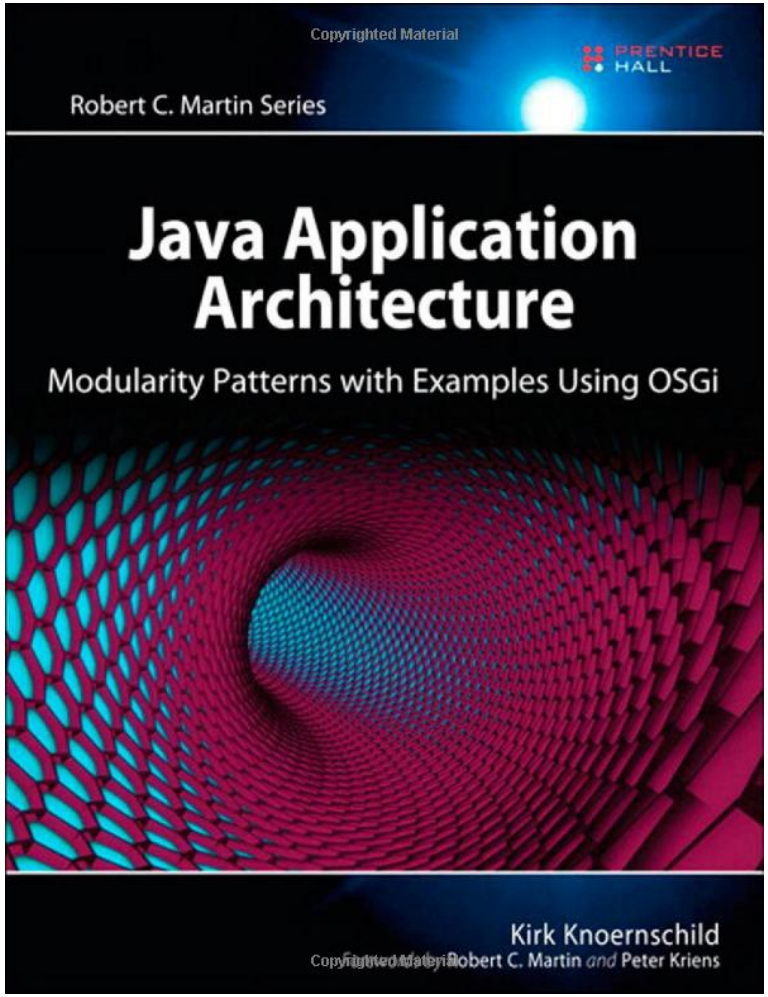


Design comum encontrado em diversos sistemas, nas camadas superiores observa-se uma **granularidade maior**, ou seja, entidades mais **fácil de se usar**, a medida que se desce para as camadas inferiores a **granularidade diminui**, ou seja, entidades menores e mais **fáceis de reutilizar**.

Design Modular



Patterns Modularização



Excelente Livro sobre Modularização de Software e Design Patterns com exemplos práticos em Java e OSGI

"Esse livro non Ecxsisssteee" – Padre Quevedo sobre esse livro

"Meu Precioooossooo" – Gollum sobre esse Livro

"Aiiii que Linnndo, Aiiiii que Luuuxo, Maara, que D+ +, Liiindoo" – Narcisa sobre esse livro

"Aaaahôôôoo trem que pula" – típico goiâno sobre esse livro

http://www.amazon.com/Java-Application-Architecture-Modularity-Patterns/dp/0321247132

Principais Patterns

THE DEVELOPER'S CONFERENCE

- 18 patterns descritos no padrão GOF
- Base Patterns
  - Manage relationships
  - Module Reuse
  - Cohesive Modules
- Depency Patterns
  - Acyclic Relationships
  - Physical Layers
- Usability Patterns
  - Published Interface
- Extensibility Patterns
- Utility Patterns

Globalcode

Manage Relationships

THE DEVELOPER'S CONFERENCE

- O relacionamento entre dois módulos existe quando uma classe em um módulo, importa ao menos uma classe de outro módulo. Em outras palavras:

  "Se mudar o conteúdo de um módulo, m2, pode causar impacto em outro módulo, m1, podemos dizer que m1 tem uma dependência física em m2 (knoerschild, 2001)"

- Dependências diretas e indiretas
- Evitando as dependências cíclicas
- Invertendo os relacionamentos
- Eliminando as dependências físicas entre módulos
- Módulos devem permitir a instalação totalmente independente.

Globalcode

Module Reuse

THE DEVELOPER'S CONFERENCE

- Um dos benefícios mais citados da Orientação a Objetos é a Reutilização de código, nós falhamos miseravelmente em conseguir isso apenas utilizando objetos.

- Módulos horizontais e verticais
- Entafizar a reusabilidade em módulos, não em Classes, isso aumentara suas chances de sucesso com a reutilização
- Os produtos OpenSource bem sucedidos e largamente adotados são arquivos JAR, que podem ser facilmente incorporados em um projeto
- Interfaces!! suas lindas

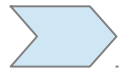
Globalcode

Modulos Coesos

THE DEVELOPER'S CONFERENCE

- O Comportamento de um módulo deve atender a um único propósito

- Um dos principais problemas de um módulo pouco coeso, é a dificuldade em entender o que ele faz.
- Um módulo com Alta Coesão é mais fácil de entender, manter e reutilizar.



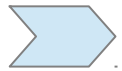
Globalcode

Dependencias Acíclicas

THE DEVELOPER'S CONFERENCE

- Quando existe um relacionamento entre dois módulos, é aumentado o acoplamento entre eles, um dependência cíclica aumenta esse acoplamento a um nível que deve ser evitado.

- O impacto de uma mudança em uma estrutura onde existem dependências cíclicas, é um loop infinito.
- As principais técnicas para quebrar uma dependência cíclica são:
    - Escalation
    - Demotion
    - CallBack



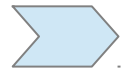
Globalcode

Camadas Físicas

THE DEVELOPER'S CONFERENCE

- O relacionamento entre módulos não deve violar as camadas conceituais.

- É comum separar de forma lógica, as camadas de um software complexo. Essa separação pode ser física.
- A separação física possibilita a reutilização de camadas, inclusive entre diferentes aplicações.
- O relacionamento entre as camadas deve ser sempre das superiores para as inferiores, nunca o contrário.



Globalcode

Interfaces Publicadas

THE DEVELOPER'S CONFERENCE

- Um módulo deve encapsular os detalhes de sua implementação, e disponibilizar uma API para que outros módulo possam acessa-lo. Essa API é a sua interface publicada, que é definida como:

  "A interface publicadas de um módulo consiste em todos os métodos, classes e pacotes que outros módulos tem a capacidade de acessar"

- A maior vantagem de ter uma boa interface publicada em seu módulo que ele se torna mais fácil de utilizar por outros desenvolvedores.
- Encapsular a implementação do seu módulo impede que outros módulo interfiram em seu funcionamento.


framework
OSGi
Platform
Service
Java
application
bundle
bundles
enterprise
Runtime
module
modules
JAR
file
class
within
modularity
many
dependencies
deployed
environment
developers
software
Using
packaged
registered
model
dynamic
capabilities
development
files
support
installed
specification
server
interface
services
manifest
deployment
system
applications
implementation
bundle
provides
include
without
must
standard
package
Eclipse
JVM
used
Web
multiple
new
registry
architecture
plug-in
management
versions
classes
packaging
information
vendors
frameworks
Figure
products
respondents
WAR
benefits

THE DEVELOPER'S CONFERENCE

- "Just about **every** software developer is an **OSGi consumer** today because just about **every platform and every IDE use OSGi**. The major platform vendors, including IBM, Oracle, and Red Hat are all using OSGi to build up their platforms. What's interesting is that **OSGi hasn't penetrated the enterprise** developer space yet. At least, it hasn't gone mainstream yet. Some people might complain that **OSGi is too complex**. But what they're really saying is that **designing modular software is really really hard**. Because it is." – Kirk Knoernschild

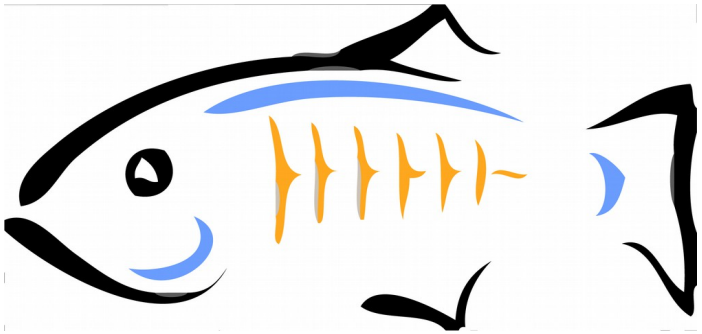
http://java.dzone.com/articles/osgi-vs-jigsaw-kirk

Globalcode

THE
DEVELOPER'S
CONFERENCE

NetBeans

OSGi™
Alliance

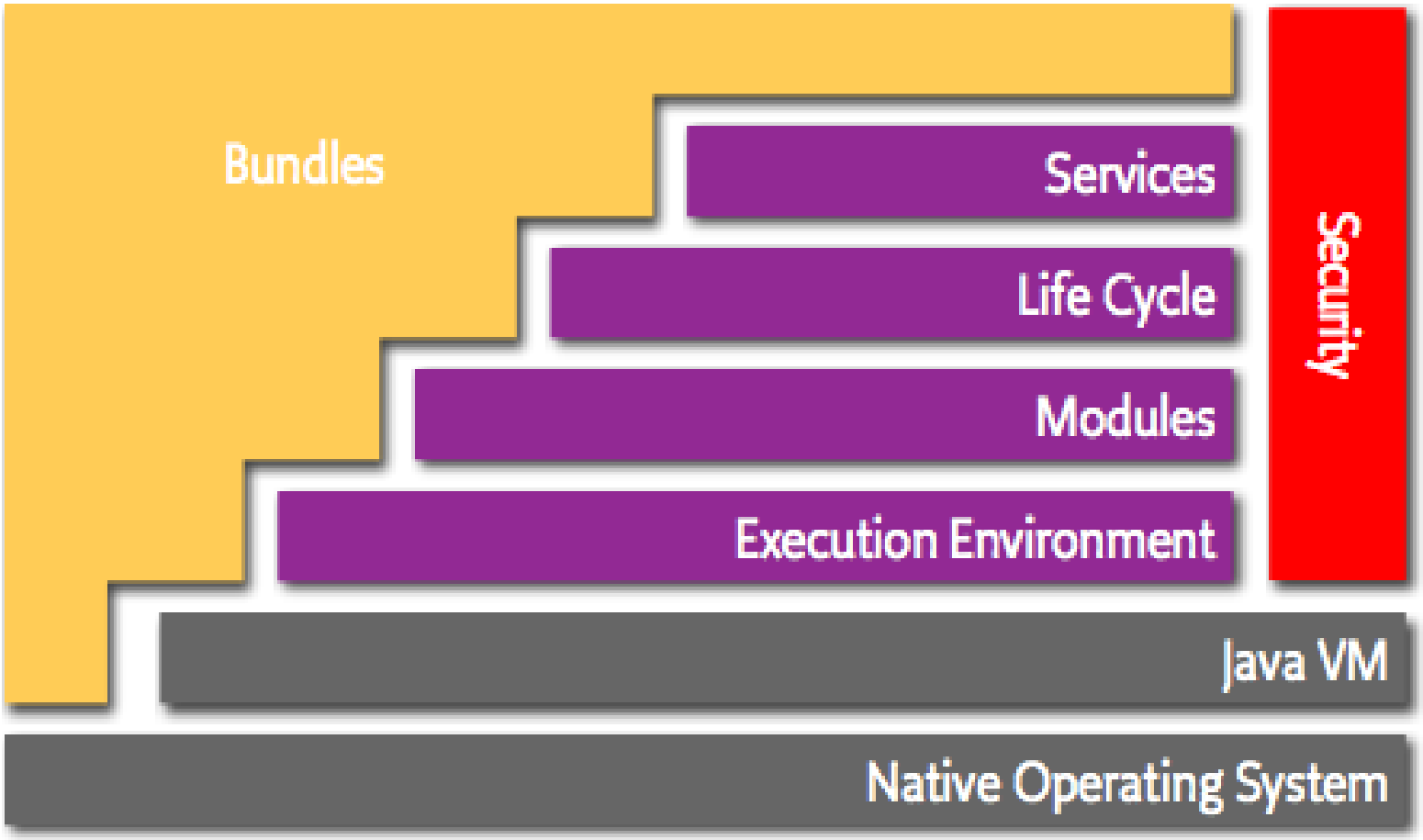


Principais Benefícios

THE DEVELOPER'S CONFERENCE

- Encapsulamento de fato
- Deploy Dinâmico
- Versionamento
- Gerenciamento de Dependências
- Modelo de Execução padronizado
- Outros: http://www.osgi.org/Technology/WhyOSGi

Globalcode


Bundles
Services
Life Cycle
Modules
Execution Environment
Security
Java VM
Native Operating System

Bundles = Módulo

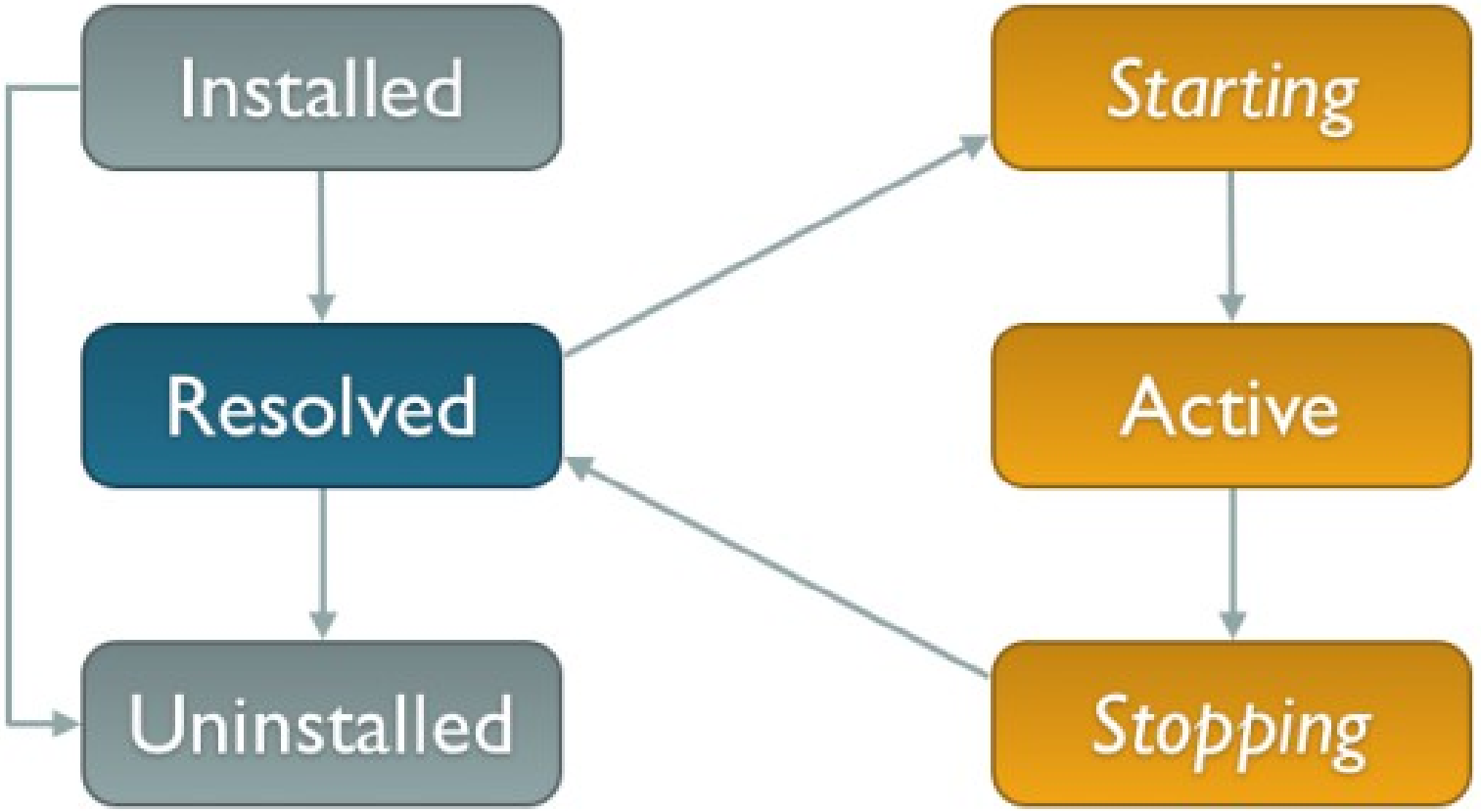
THE DEVELOPER'S CONFERENCE

- Bundle é um módulo ou parte dele
- Equivalente a um Jar, porém com diversas informações adicionais que são interpretadas pela plataforma.
  - Identificador
  - Versão
  - Dependências (bundles e versões)
  - Exposições/publicações

Globalcode

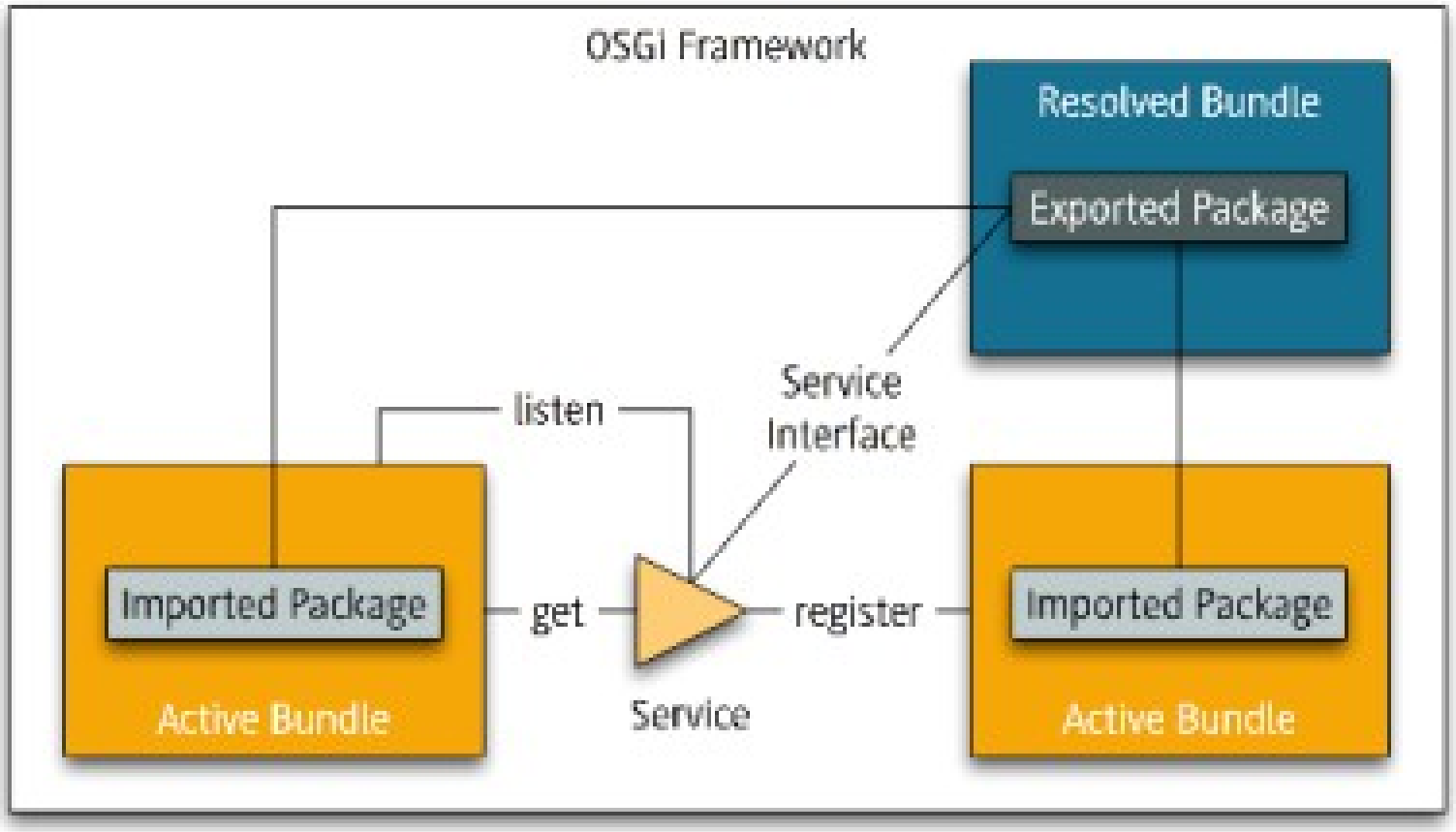
LifeCycle de um Bundle

THE DEVELOPER'S CONFERENCE



Globalcode

Services


OSGi Framework
Resolved Bundle
Exported Package
Service
Interface
listen
Imported Package
get
register
Imported Package
Active Bundle
Service
Active Bundle

OSGI Services

THE DEVELOPER'S CONFERENCE

- "This is similar to the **service-oriented architecture** made popular with **web services**. The key difference between web services and **OSGi services** is that web services always require some **transport layer**, which makes it **thousands times slower than OSGi** services that use direct method invocations."

*Retirado do site www.osgi.org

Globalcode

Demonstração

THE DEVELOPER'S CONFERENCE

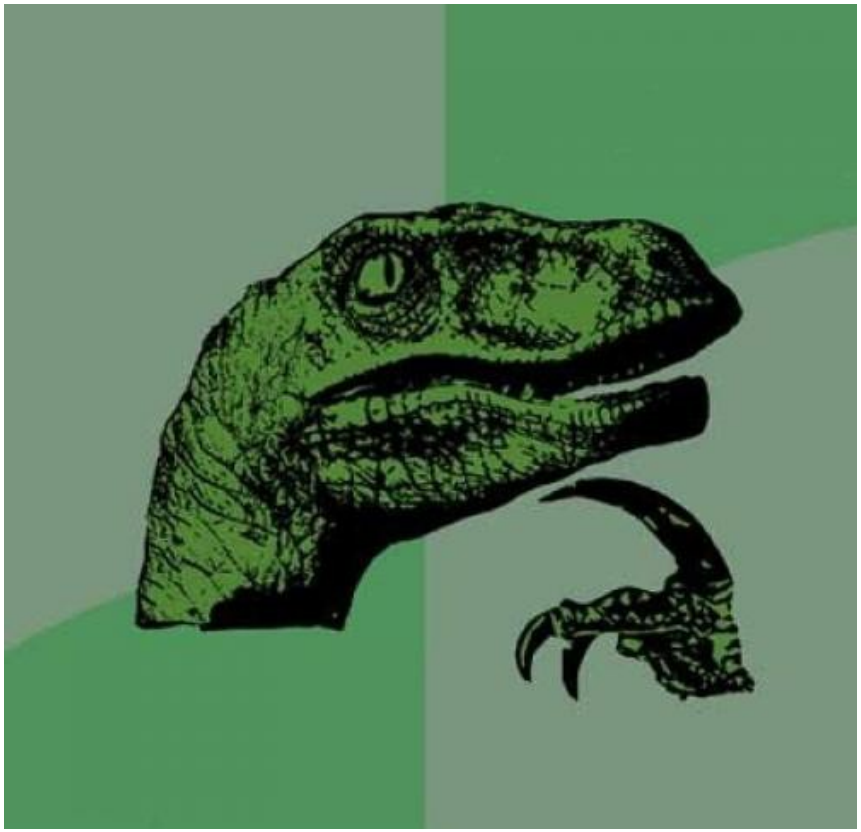
- Projeto web modular OSGi + JavaEE
- Empacotamento e geração de arquivos Manifest.mf via maven
- EJBs publicados como osgi services
- Integração a JPA, JTA
- Integração a CDI provida pelo Glassfish Server

- https://github.com/filipeportes/goevent

Globalcode

Dúvidas?

THE DEVELOPER'S CONFERENCE



@filipeportes
omeuefilipe@gmail.com
github.com/filipeportes

Globalcode