Plataforma JEE

Plataforma JEE

O que é?

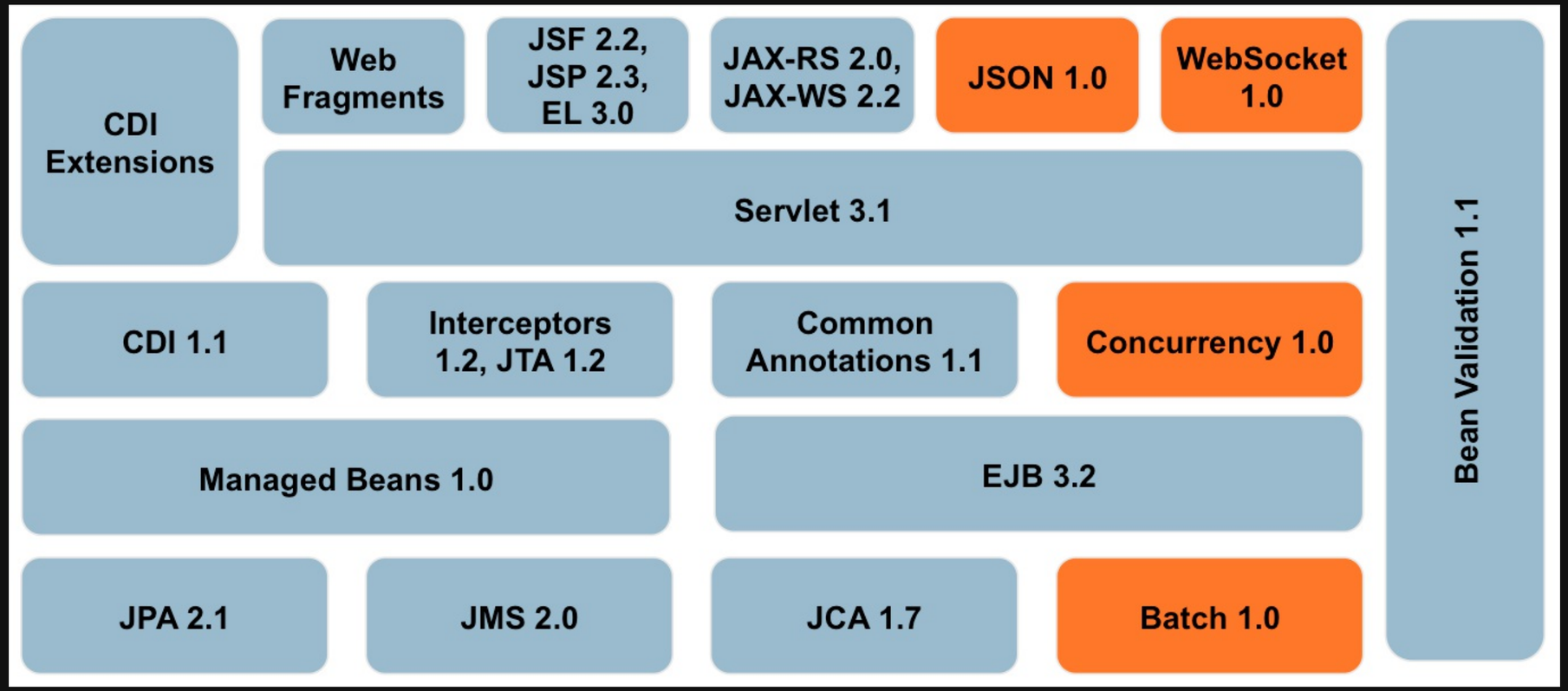
Plataforma JEE

Conjunto de especificações concebidas pela comunidade (JCP) e industria para fornecer um ambiente onde componentes distribuídos colaboram entre si.

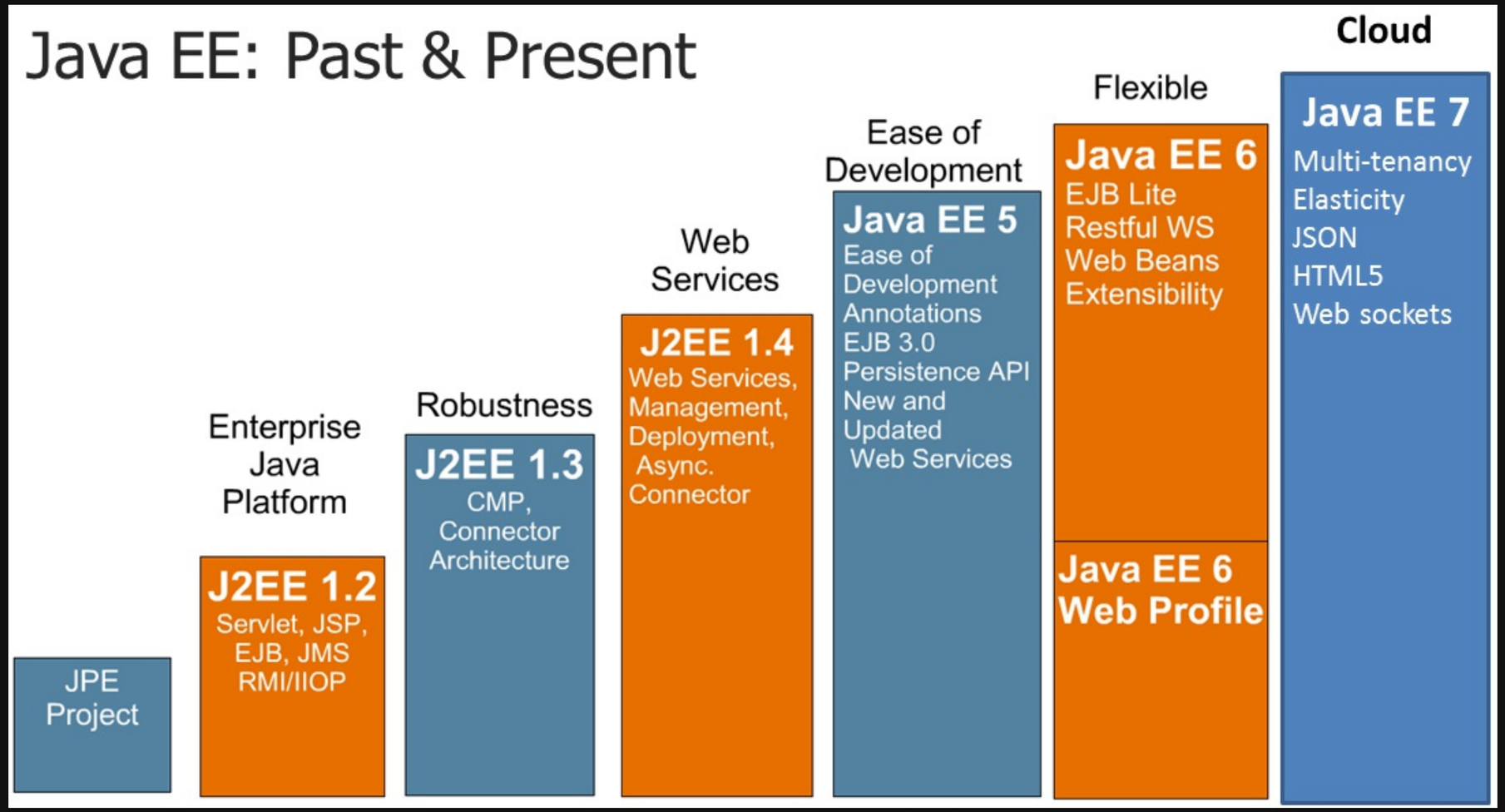
Plataforma JEE

O objetivo da plataforma é reduzir o tempo de desenvolvimento e complexidade das aplicações enquanto maximiza o seu desempenho.

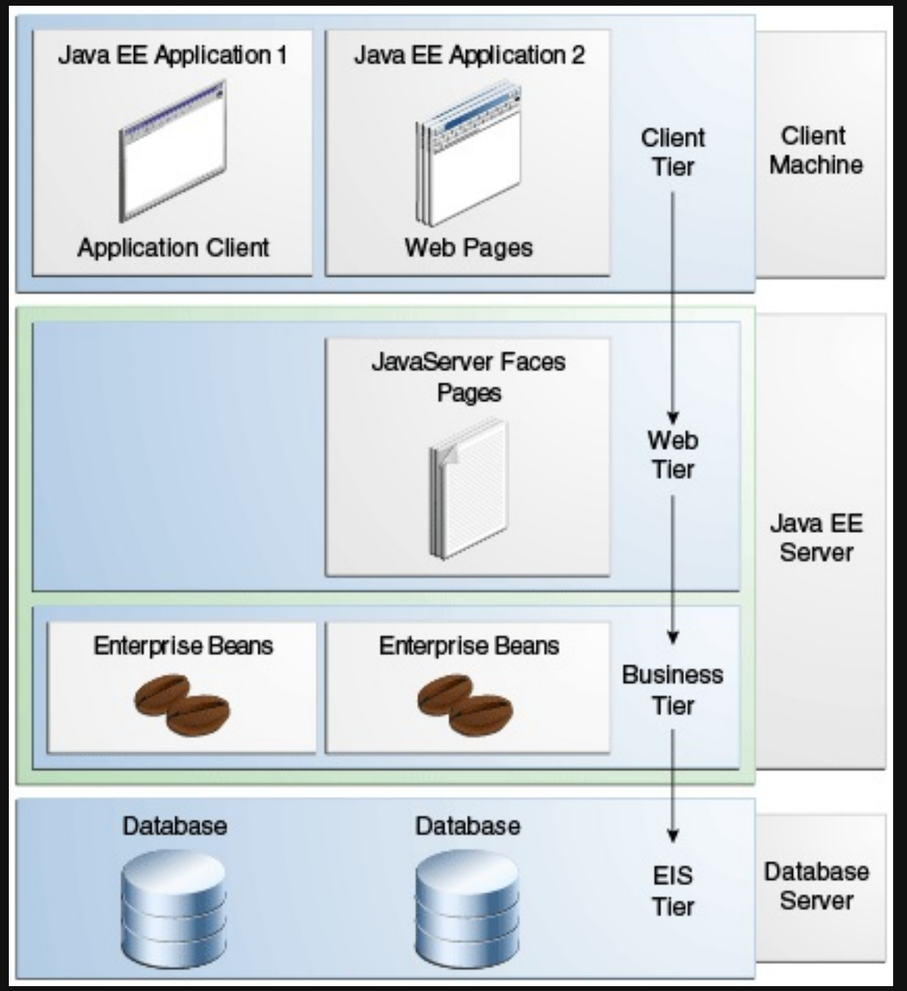
Especificações



Histórico


Java EE: Past & Present
JPE Project
Enterprise Java Platform
J2EE 1.2
Servlet, JSP, EJB, JMS RMI/IIOP
Robustness
J2EE 1.3
CMP, Connector Architecture
Web Services
J2EE 1.4
Web Services, Management, Deployment, Async. Connector
Ease of Development
Java EE 5
Ease of Development Annotations EJB 3.0 Persistence API New and Updated Web Services
Flexible
Java EE 6
EJB Lite Restful WS Web Beans Extensibility
Java EE 6 Web Profile
Cloud
Java EE 7
Multi-tenancy Elasticity JSON HTML5 Web sockets

Camadas

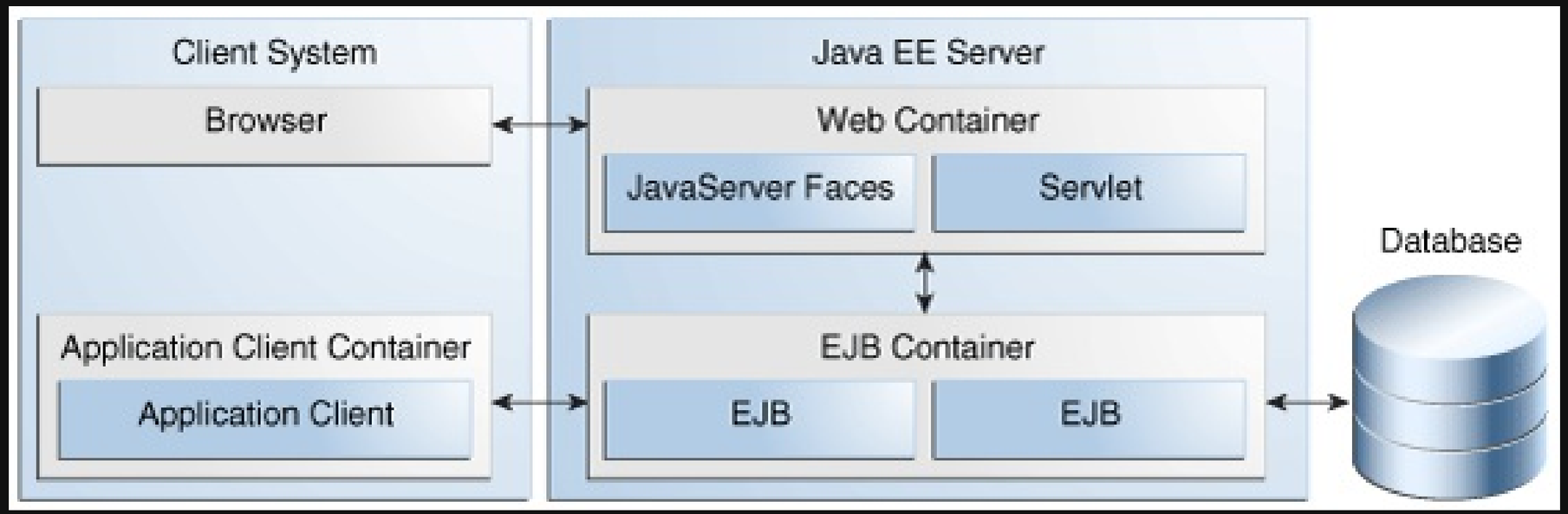

Java EE Application 1
Application Client
Java EE Application 2
Web Pages
Client Tier
Client Machine
JavaServer Faces Pages
Web Tier
Java EE Server
Enterprise Beans
Enterprise Beans
Business Tier
Database
Database
EIS Tier
Database Server

Na prática, quem junta essa "salada" de especificações?

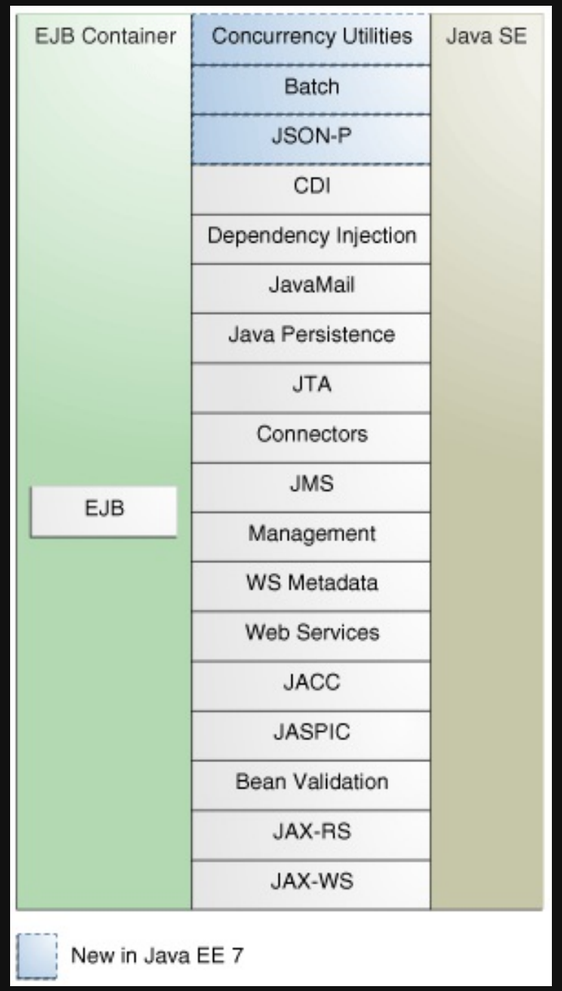
Container

- Ambiente propício para a execução de aplicações JEE.
- Conjunto de serviços básicos (operacionais) para as aplicações
- Componentes focam na implementação da lógica de negócio

Organização



APIs


EJB Container
Concurrency Utilities
Java SE
Batch
JSON-P
CDI
Dependency Injection
JavaMail
Java Persistence
JTA
Connectors
JMS
EJB
Management
WS Metadata
Web Services
JACC
JASPIC
Bean Validation
JAX-RS
JAX-WS
New in Java EE 7

Alguns Fornecedores

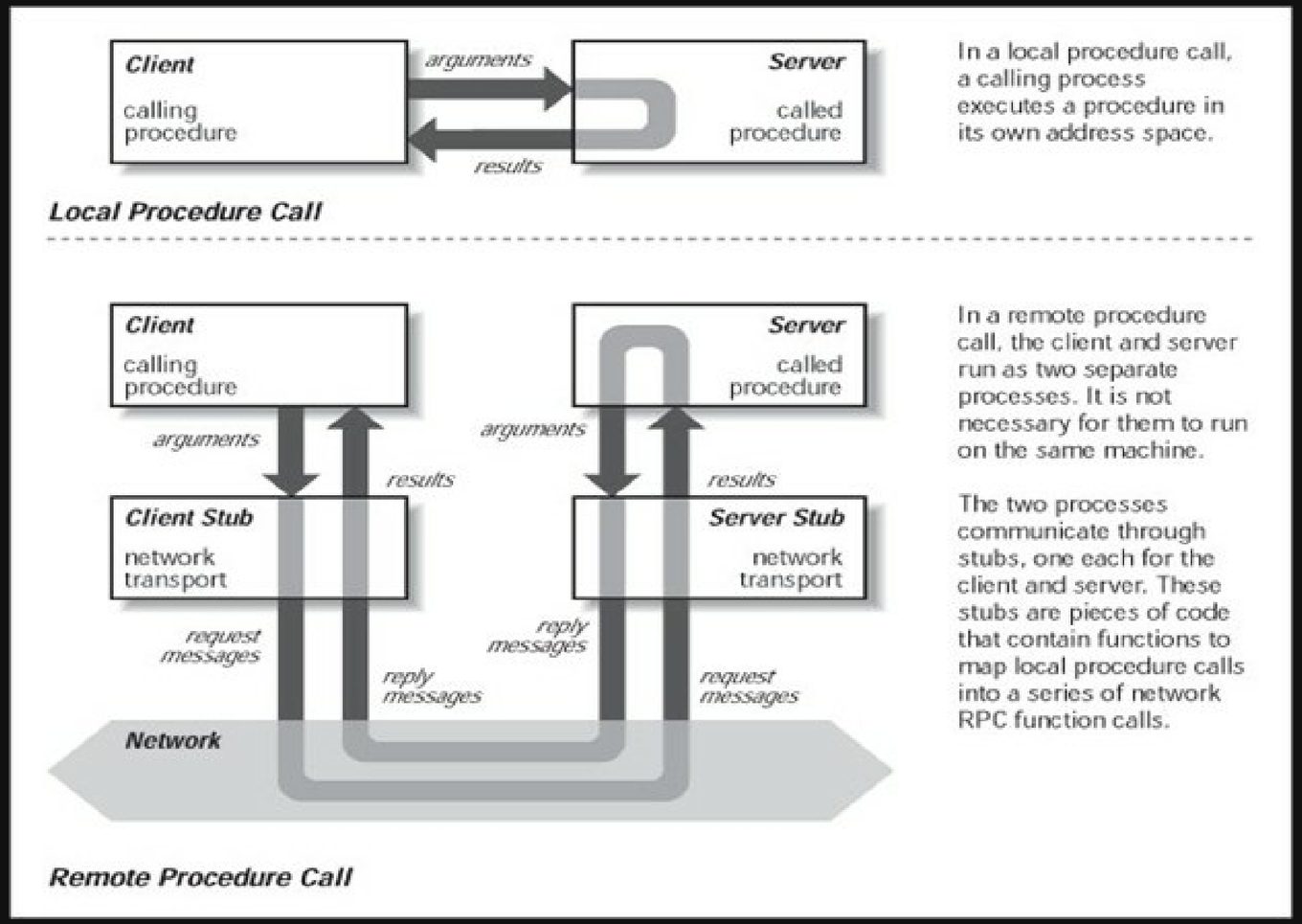
Serviços Básicos

Invocação Remota

Uma aplicação distribuída necessita disparar execuções em componentes que não estão disponíveis localmente.

Invocação Remota

RPC



Invocação Remota

RMI/IIOP

- Protocolo que permite comunicação remota entre objetos
- Objeto remoto acessado via proxy
- Permite interoperar Java e Corba

Localização

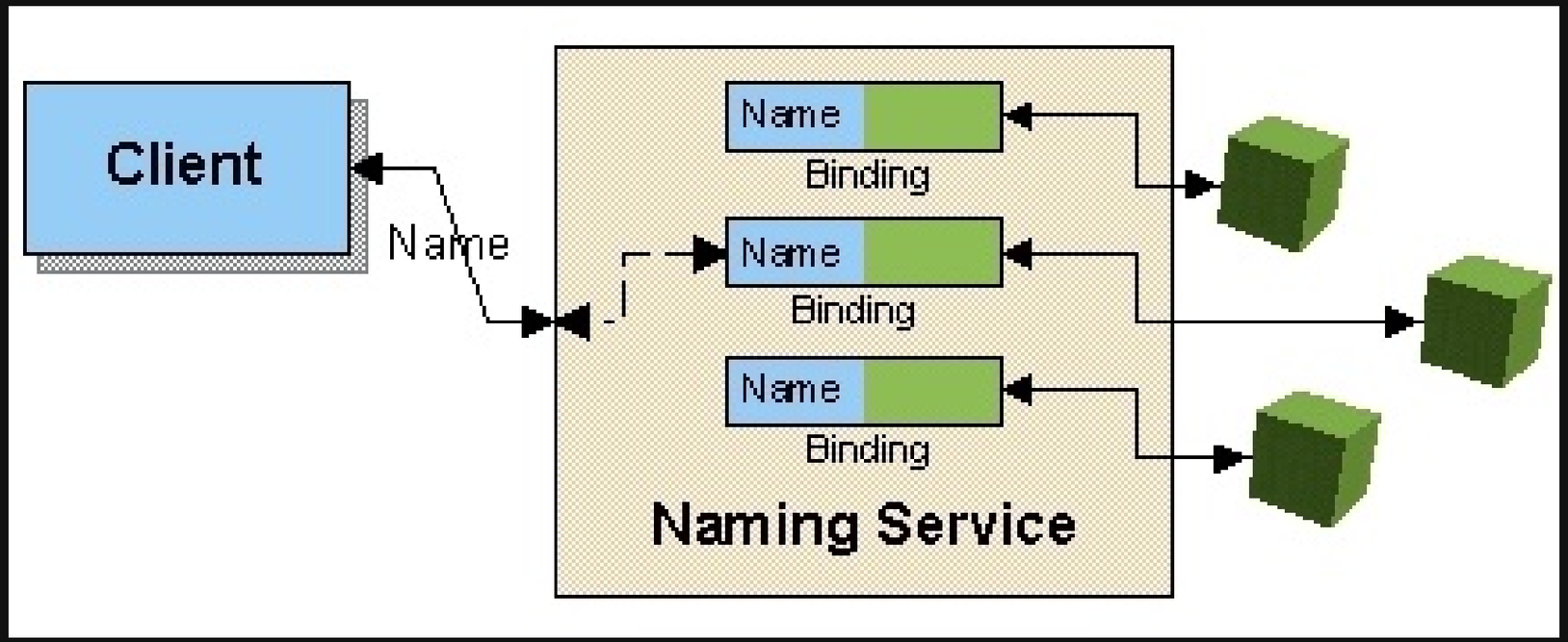
Aplicações distribuídas necessitam acessar recursos e componentes para executar suas ações.

Localização

A aplicação pode estar diretamente "ligada" ao recurso ou acessá-lo de outra forma.

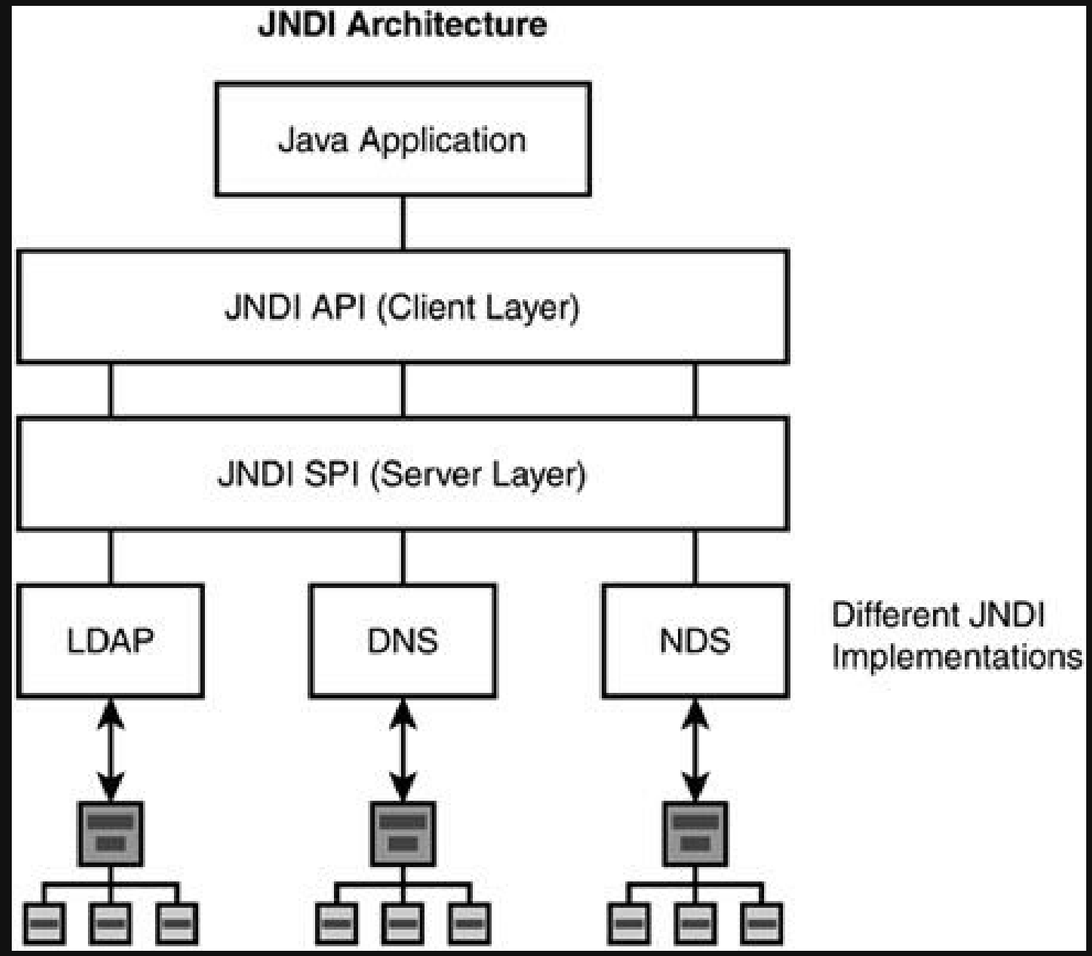
Localização

Serviço de Nomes



Localização

JNDI



Localização

Localização

```
// Lookup the remote object via JNDI
Context ctx = new InitialContext(System.getProperties());
Object remoteObject = ctx.lookup(“MyObj”);
// Cast the remote object, RMI-IIOP style
MyRemoteObj generator = (MyRemoteObj)
    PortableRemoteObject.narrow(remoteObject, MyRemoteObj.class);
generator.doSomething();
```

Recursos e Dependência

Uma aplicação pode localizar ou construir diretamente o recurso que necessita

```
protected void doGet(...) {
    String mercadoria = req.getParameter("mercadoria");
    ControladorEstoque control = new ControladorEstoque();
    String status = control.reservaMercadoria(mercadoria);
    resp.getWriter().append("Status da Reserva: ").append(status);
}
```

Recursos e Dependência

Problemas da abordagem

- Código de infraestrutura misturado ao código útil (negócio).
- Aplicação conhece como construir o recurso. Logo, está acoplado a ele.

Recursos e Dependência

Uma alternativa a essa abordagem, é inverter a ordem das coisas...

Recursos e Dependência

Injeção de Recurso

- Permite injetar recursos em um componente
- Ambos devem ser gerenciados pelo container
- O recurso é identificado por um nome
  - @Resource

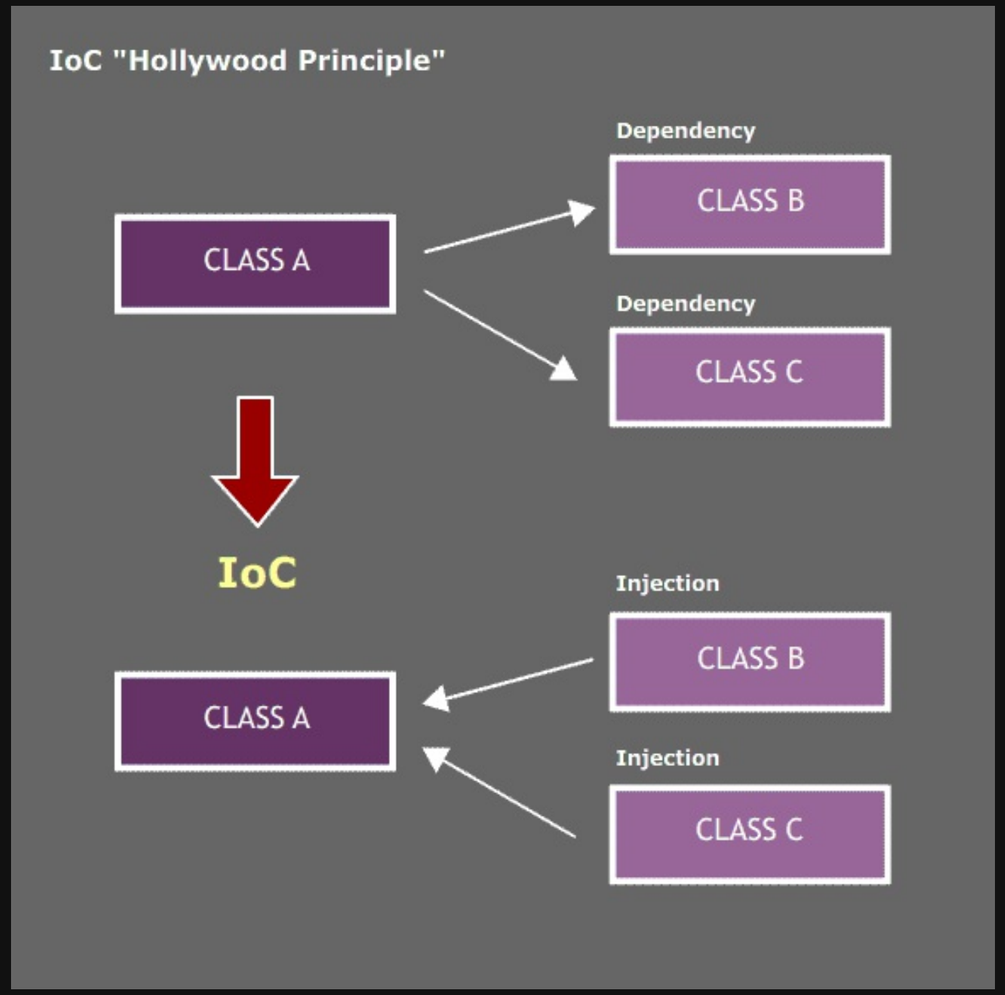
Recursos e Dependência

Exemplo

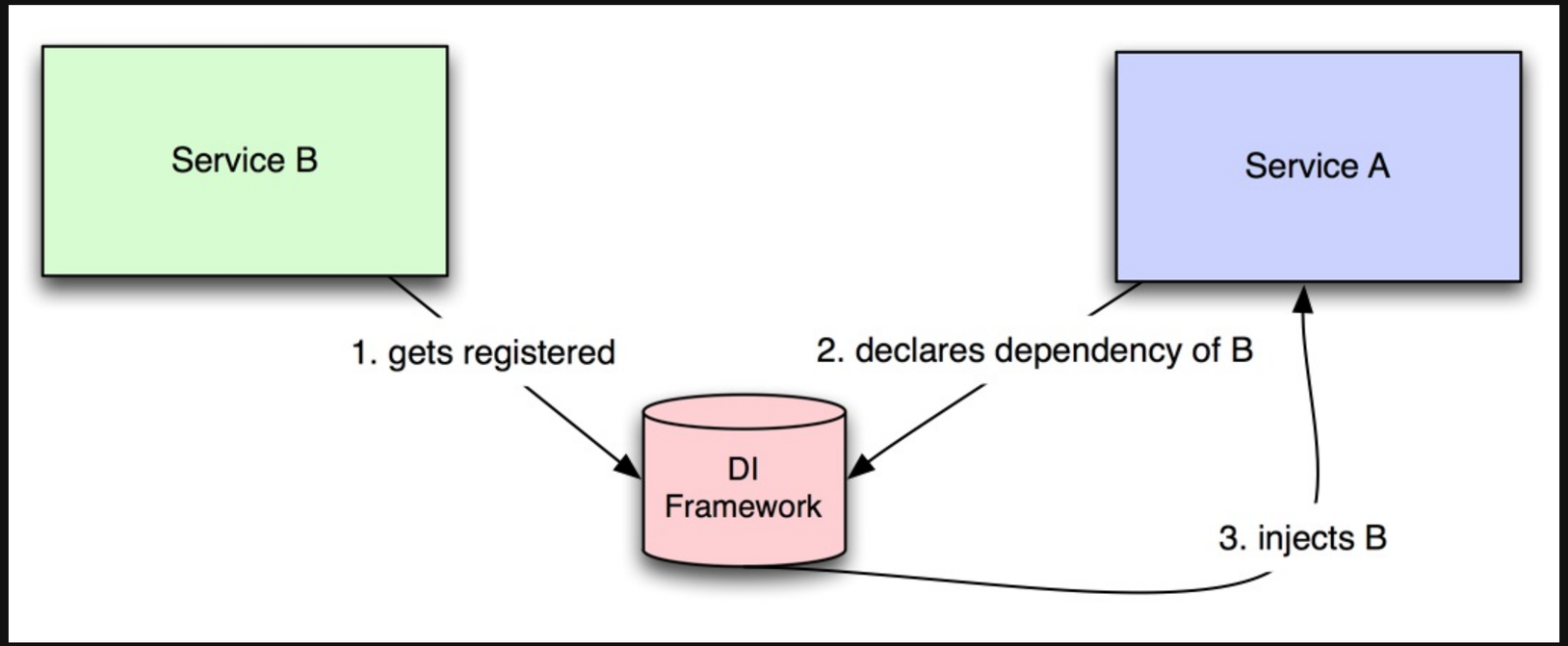
```
public class LibraryServlet extends HttpServlet {
    @Resource(lookup="java:comp/SQLiteDataSource")
    private javax.sql.DataSource dsc;
    ...
}
```

Recursos e Dependência

Inversão de Controle



Recursos e Dependência



Recursos e Dependência

Considerações

- **O componente e o objeto devem ser criados pelo container**
- **O componente é um objeto gerenciado pelo container**
  - Servlets, Managed Beans, EJBs, etc
- **As dependências são declaradas por anotações**
  - @Inject, @EJB, etc
- **O objeto inserido possui um escopo**
  - @RequestScoped, @SessionScoped, @ApplicationScoped, etc

Recursos e Dependência

Objeto

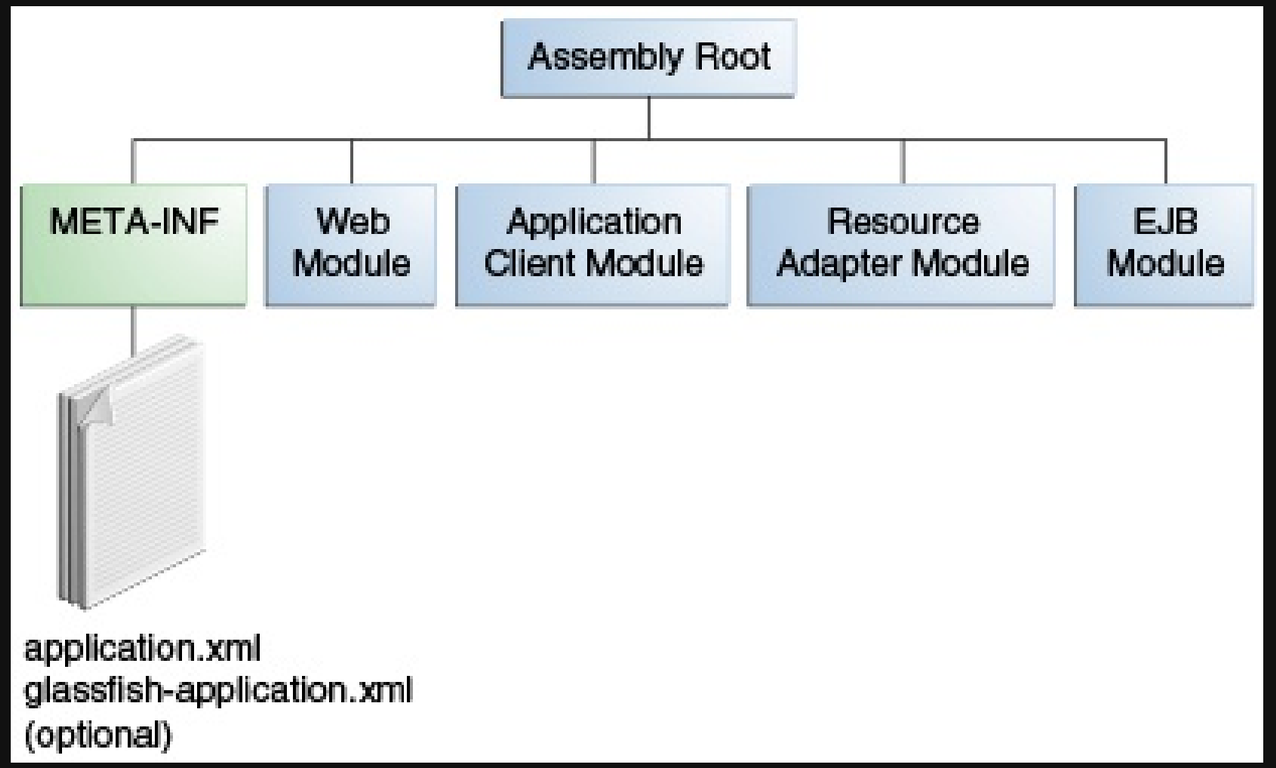
```
@RequestScoped
public class EstoqueSimples implements Estoque {
...
}
```

Componente

```
public class ReservaServlet extends HttpServlet {
    @Inject private ControladorEstoque controle;
    ...
    protected void doGet(...) {
    String merc = request.getParameter("mercadoria");
    controle.avaliarPedido(merc));
}
```

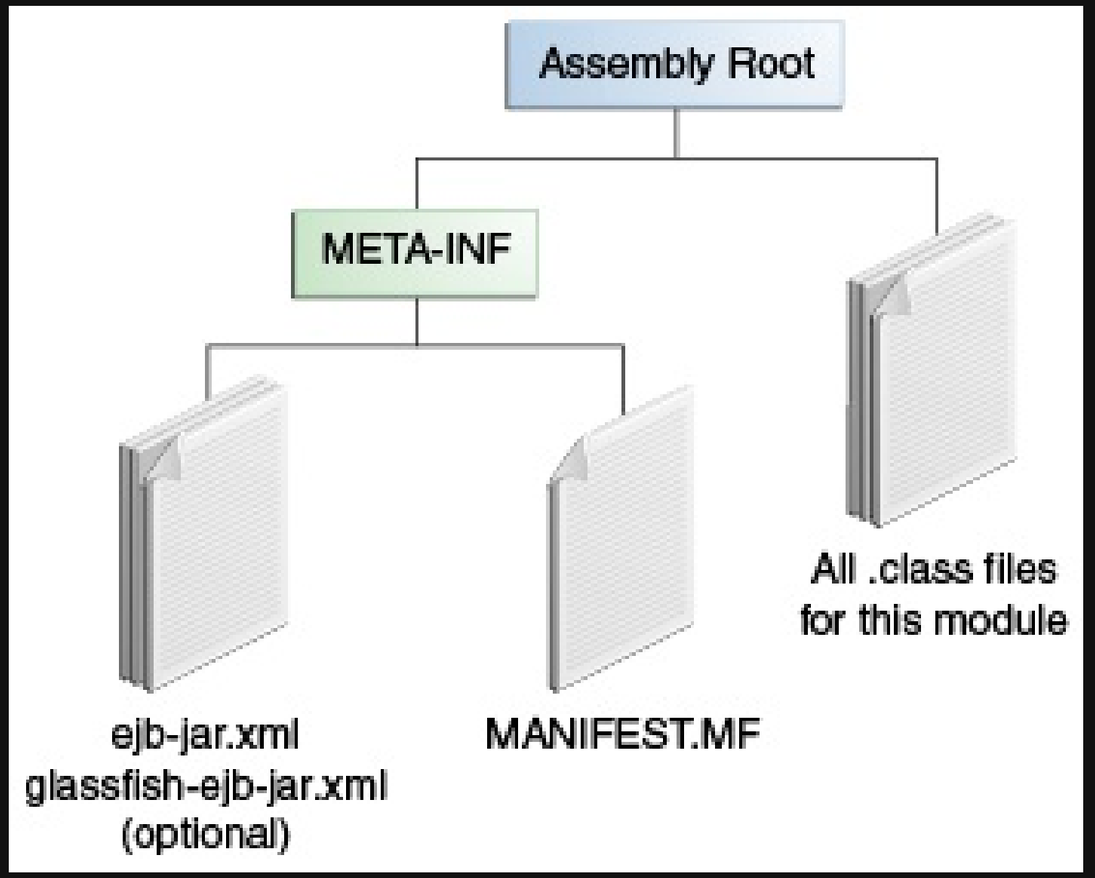
Empacotamento

Estrutura de uma aplicação



Empacotamento

Estrutura de um componente



Exercício 1

Uma aplicação corporativa reserva de mercadorias por meio de um servlet ReservaMercadoriaServlet e um objeto do tipo ControleEstoque.

Adaptar o servlet e o objeto para que a referência ao objeto seja obtida por meio de injeção simples com o escopo de Requisição.

Dicas

1. Crie o servidor de aplicação
2. Importe o projeto distribuidora que se encontra na pasta de exercícios
3. User as anotações @Inject e RequestScoped