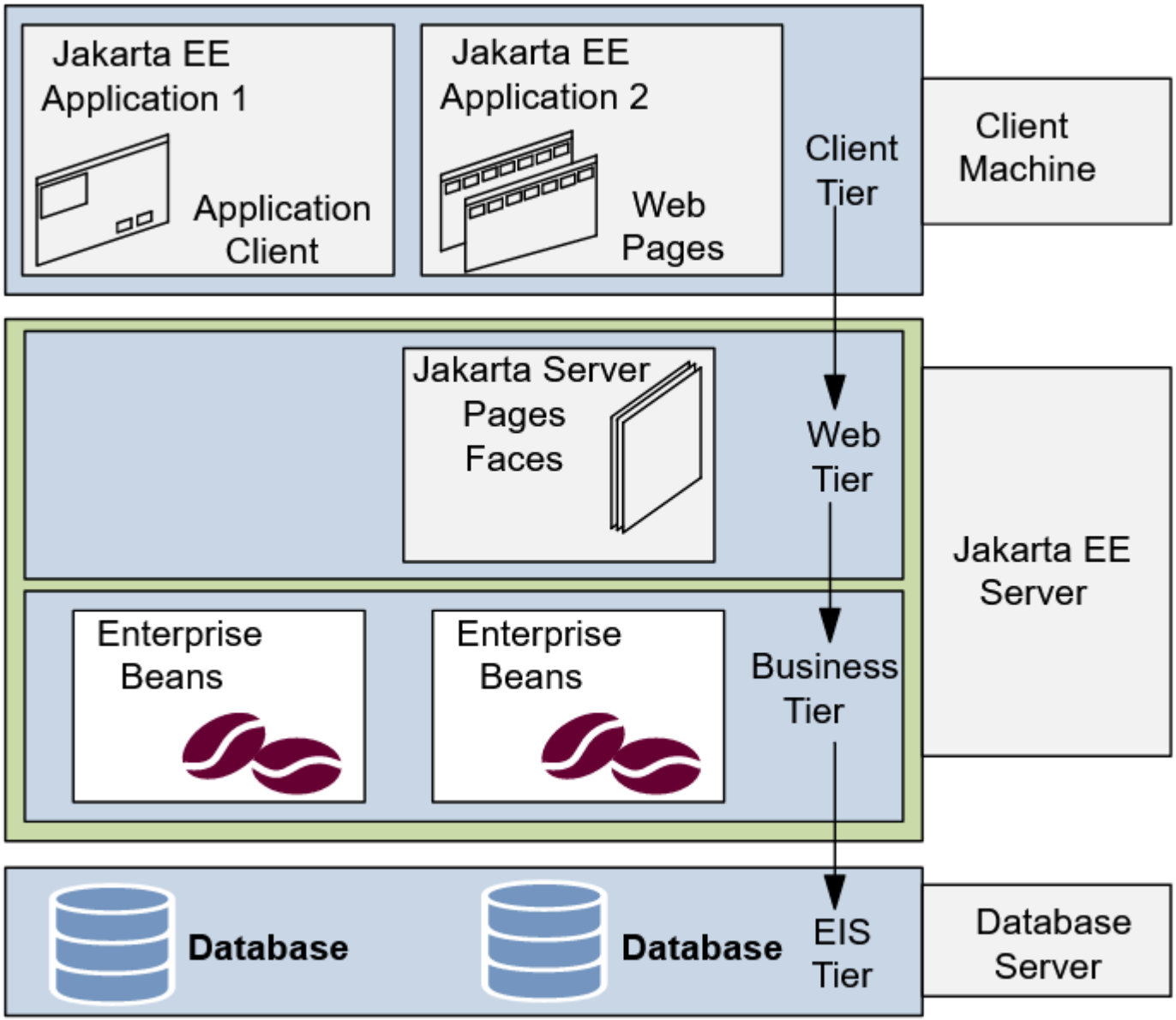
Especificação do JAVA EE (jakarta):
https://jakarta.ee/specifications/

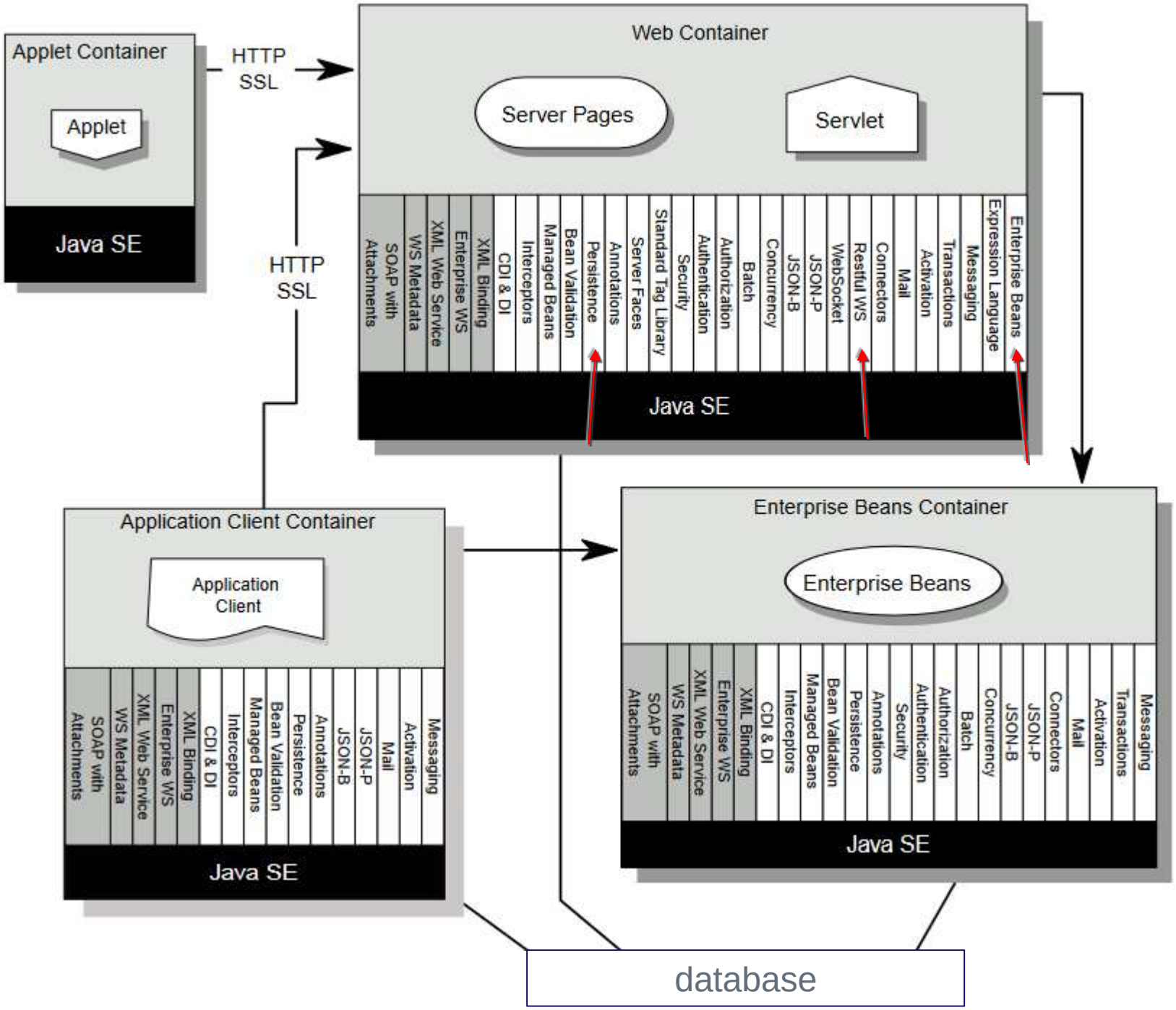
Eclipse Starter for Jakarta EE:
https://start.jakarta.ee/

Java EE versus Spring boot:

https://blog.payara.fish/jakarta-ee-vs.-spring-boot-choosing-the-right-framework-for-your-project



Arquitetura do Jakarta Enterprise Edition


Applet Container
Applet
Java SE
HTTP
SSL
HTTP
SSL
Web Container
Server Pages
Servlet
Enterprise Beans
Expression Language
Messaging
Transactions
Activation
Mail
Connectors
Restful WS
WebSocket
JSON-P
JSON-B
Concurrency
Batch
Authorization
Authentication
Security
Standard Tag Library
Server Faces
Annotations
Persistence
Bean Validation
Managed Beans
Interceptors
CDI & DI
XML Binding
Enterprise WS
XML Web Service
WS Metadata
SOAP with Attachments
Java SE
Application Client Container
Application Client
Java SE
Enterprise Beans Container
Enterprise Beans
Java SE
database

Java Bean: Standard para definição de classes reutilizáveis

- Classe Java que:
  - tem todas as propriedades (atributos) privadas
  - métodos públicos (getters e setters) acedem às propriedades
  - tem um construtor sem parâmetros
  - implementa a interface Serializable
- POJO (Plain Old Java Object) serializável.

Enterprise Beans (EB)

- Session Beans
  - Stateless
  - Stateful
  - Singleton
- Tipos de acesso a um Bean
- Message Driven Beans

Enterprise Beans

Conceito

- Componente executado do lado do servidor que implementa a lógica da aplicação / negócio

- O Bean *container* oferece os serviços do sistema para:
  - Gestão de transações
  - Segurança
  - Concorrência
  - Escalabilidade

Enterprise Beans (EB)

- O código da aplicação deve poder executar em qualquer plataforma que cumpra as especificações Jakarta EE.

(e.g. GlassFish, Apache TomEE), etc. *)

* https://jakarta.ee/compatibility/

Tipos de Enterprise Beans

- Session Beans

- Implementam uma tarefa para a aplicação cliente.
- Opcionalmente podem implementar um *Web service*.

  - Stateful
  - Stateless
  - Singleton

Tipos de EJB

- Message-Driven Beans

- Atuam como *Listeners* para um determinado tipo de mensagens

(e.g. mensagens da Java Message Service API)

Acesso a *session* Beans (1)

- O cliente de um *session* Bean obtém a referência para uma instância do Bean por:

  - *Dependency injection (usando anotações da linguagem)*

  ou

  - *JNDI* (Java Naming and Directory Interface) *lookup*

- O cliente tem acesso a um *session* Bean através dos métodos de uma interface (acesso remoto) ou através dos métodos públicos do Bean (acesso local).

Acesso a *session* Beans (1)

- 3 tipos de acesso:

  - *Local*
  - *Remoto*
  - *Como web service*

Acesso a *session* Beans (2)

- **Local** – contexto local (JVM) do servidor

```
@Local
public interface ExampleLocal { …  }
```

- **Remote** – interface remota (dentro ou fora da JVM)

```
@Remote
public interface ExampleRemote { …  }
```

Acesso a *session* Beans (2)

- **Web Service** – serviço sobre HTTP

```
@WebService
public interface InterfaceName { …  }
```

Acesso a *session* Beans (3)

- Abordagem clássica através do serviço JNDI:

```
ExampleLocal example = (ExampleLocal)
        InitialContext.lookup
("java:module/ExampleLocal");
```

```
ExampleRemote example = (ExampleRemote)
        InitialContext.lookup
("java:global/ExampleRemote");
```

Acesso a *session* Beans (3)

- Por *dependency injection*

```
@EJB
Example example
```

O habitual *lookup* pode ser substituído pela anotação @EJB em que o servidor JEE implicitamente “injecta” o código para obter o EJB referenciado.

Stateful Session Beans

Características

Cada instância de um Stateful Bean está associada a um único cliente.

- O Bean mantém o estado do objeto entre as chamadas de métodos por um dado cliente (estado = valores das variáveis de instância)
- Permitem guardar informação do cliente entre múltiplas invocações
- Possuem um tempo de vida (configurável) até serem removidos

Stateful Session Beans

Exemplo:

Interface remota

```
@Remote
public interface CarrinhoDeCompras {

   public void adicionarItem(String item);
   public  void removerItem(String item);
   public List<String> getItens();
   public void limparCarrinho();
}
```

Stateful Session Beans

Exemplo: Bean

```
@Stateful
public class CarrinhoDeComprasBean implements CarrinhoDeCompras {

    private List<String> itens = new ArrayList<>();

    public void adicionarItem(String item) {
        itens.add(item);    }
    public void removerItem(String item) {
        itens.remove(item);    }
    public List<String> getItens() {
        return itens;    }
    public void limparCarrinho() {
        itens.clear();
    } }
```

Stateful Session Beans

Exemplo: Aceder através de um outro Bean

```
public class PedidoServiceBean {

    @EJB
    private CarrinhoDeCompras carrinho;

    public void processarPedido() {
        carrinho.adicionarItem("Monitor");
        carrinho.adicionarItem("Teclado");

        List<String> itens = carrinho.getItens();
        System.out.println("Itens do pedido:");
        for (String item : itens) {
            System.out.println("- " + item);
        }
```

Stateless Session Beans

Características

- Não mantêm informação especifica de um cliente;
- O estado existe apenas durante a invocação de um método;
- O servidor gere uma *pool* de instâncias que servem pedidos de vários clientes;
- Interface pode ser exposta como *web service*;

Stateless Session Beans

Exemplo: interface

```
@Remote
public interface Calculadora {
   public  int somar(int a, int b);
   public  int subtrair(int a, int b);
   public  int multiplicar(int a, int b);
   public double dividir(int a, int b);
}
```

Stateless Session Beans

Exemplo: bean

```
@Stateless
public class CalculadoraBean implements Calculadora {

    @Override
    public int somar(int a, int b) {
        return a + b;
    }
    @Override
    public int subtrair(int a, int b) {
        return a - b;
    } …
```

Stateless Session Beans

Exemplo: Restful webservice

```
@Path("/calc")
@Produces(MediaType.TEXT_PLAIN)
public class CalculadoraResource {

    @EJB
    private Calculadora calculadora;

    @GET
    @Path("/soma")
    public String soma(@QueryParam("a") int a, @QueryParam("b") int b) {
        return "Resultado: " + calculadora.somar(a, b);
    }
```

Singleton Session Beans

Características

- Instanciado apenas uma vez, por aplicação
- Estado partilhado entre todos os clientes
- Estado preservado entre invocações
- Anotação *@Startup* indica que deve ser instanciado no arranque da aplicação

Exemplo:

```
@Singleton
public class CounterBean {
    private int hits = 1;
    // Increment and return the number of hits
    public int getHits() {
    return hits++;
}
```

Message Driven Beans

Características

- Consiste num receptor **assíncrono** de mensagens Java
- Não mantém estado
- Um MDB pode processar mensagens de múltiplos clientes

Message Driven Beans

Características

- O cliente não acede diretamente à interface do MDB, usa um serviço de mensagens, e.g.,

  JMS – Java Messaging Service;