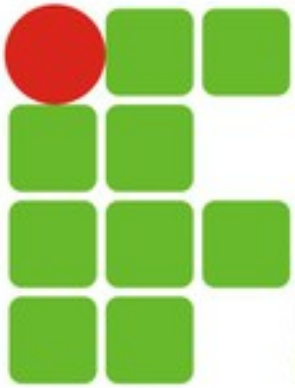


INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
RIO GRANDE DO NORTE

Fundamentos da Plataforma Java EE

Prof. Fellipe Aleixo (*fellipe.aleixo@ifrn.edu.br*)

Como a plataforma Java EE trata o

SERVIÇO DE NOMES

Serviço de Nomes

- Num sistema distribuído os componentes necessitam acessar outros componentes, bem como os recursos
- Na JEE o serviço de nomes é o *Java Naming and Directory Interface* (JNDI)
- Cada componente e/ou recurso é associado a um nome único
  - Exemplo: **java:comp/DefaultDataSource**

Recursos

- São objetos que proporcionam conexão aos serviços oferecidos pela plataforma
  - Exemplos:
    - Serviço de armazenamento persistente
    - Serviço de mensagens assíncronas

- Um administrador é responsável por criar os recursos e associa-los a um *namespace*

Injeção de Dependência

- Os recursos podem ser “injetados” em um componente da aplicação através de anotação
  - **@Resource**

- Outra opção é especificar o que deve ser injetado através dos descritores de implantação

Armazenamento Persistente

- Para o armazenamento persistente das informações do sistema geralmente utilizamos um banco de dados relacional
  - Acesso através da API JDBC – através de um objeto **DataSource**
  - Um objeto DataSource pode ser imaginado como uma fábrica de conexões para um banco de dados
    - Método **getConnection()**

Recurso JDBC

- Um objeto DataSource pode ser registrado com um nome JNDI
  - Tal objeto implementa um “pool de conexões”

- Aplicações que usam a API de persistência Java – JPA especificam um DataSource em um arquivo denominado de **persistence.xml**
  - **<jta-data-source>jdbc/MyOrderDB</jta-data-source>**

Criação de Recursos

- Antes de implantar e executar aplicações corporativas, os recursos necessários precisam ser criados
  - Através de utilitários disponibilizados pelos servidores de aplicação específicos

Como a plataforma Java EE trata a

INJEÇÃO DE REFERÊNCIAS

Injeção

- A plataforma Java EE provê dois mecanismos
  - Injeção de recurso
  - Injeção de dependência

- Tais mecanismos permitem que componentes obtenham a referência a um recurso ou outro componente sem ter que instanciar

Injeção

- As referências necessárias são declaradas (geralmente interfaces) e anotadas
  - A anotação define um “ponto de injeção”

- Em tempo de execução, o contêiner providencia a instanciação e atribuição dos elementos desejados

Injeção de Recurso

- Permite a injeção de qualquer recurso disponível no *namespace* JNDI
- Exemplo:

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

Injeção de Recurso

- De forma semelhante, pode ser anotado um método modificador da referência desejada

```
public class MyServlet extends HttpServlet {
   private javax.sql.DataSource dsc;
   ...
   @Resource(name="java:comp/DefaultDataSource")
   public void setDsc(java.sql.DataSource ds) {
      dsc = ds;
   }
}
```

Injeção de Dependência

- Permite que um componente da aplicação tenha acesso a outro componente distribuído
  - No componente dependente ocorre a declaração do elemento do qual depende
  - O contêiner providencia a instanciação e a injeção das referências às mesmas nos pontos de injeção
  - A injeção ocorrem em tempo de execução

Injeção de Dependência

- A injeção de dependência do Java EE leva em consideração o escopo desejado

```
@javax.enterprise.context.RequestScoped
public class CurrencyConverter { ... }
```

- O ponto de injeção é marcado por **@Inject**

```
public class MyServlet extends HttpServlet {
   @Inject CurrencyConverter cc;
   ...
}
```

Diferença entre os Tipos de Injeção

Mecanismo de Injeção	Pode injetar recursos JNDI diretamente	Pode injetar classes regulares diretamente	Resolvido por	*Typesafe*
Injeção de Recurso	Sim	Não	Nome do recurso	Não
Injeção de dependência	Não	Sim	Tipo	Sim

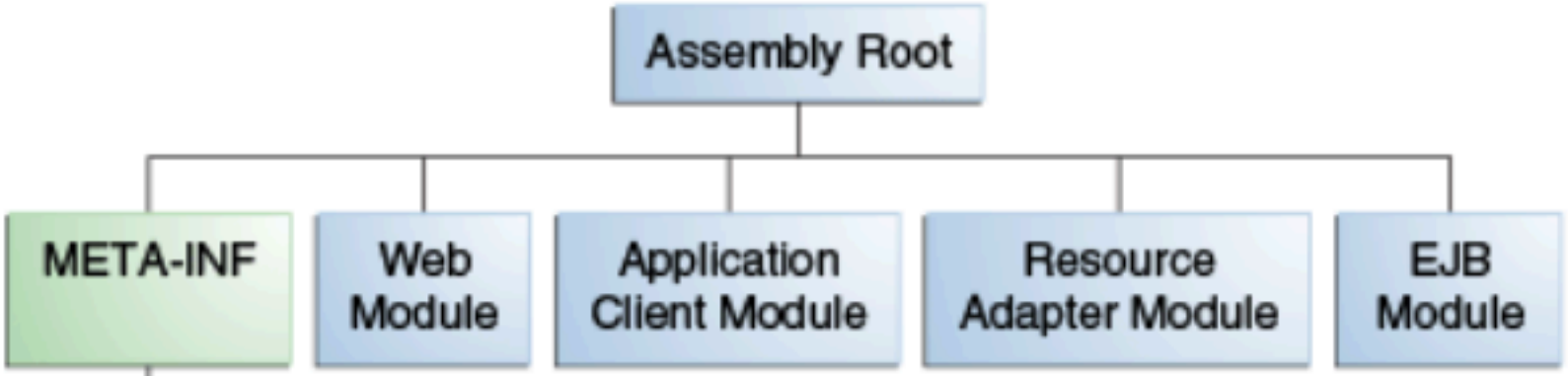
Como a plataforma Java EE propõe o

EMPACOTAMENTO DE APLICAÇÕES

Empacotamento de Aplicações

- Uma aplicação corporativa pode ser empacotada em
  - Um arquivo Java – JAR
  - Um arquivo Web – WAR
  - Um arquivo *Enterprise* – EAR
- Vários módulos Java EE podem ser empacotados em tais arquivos

Estrutura de um Pacote EAR



application.xml
glassfish-application.xml
(optional)

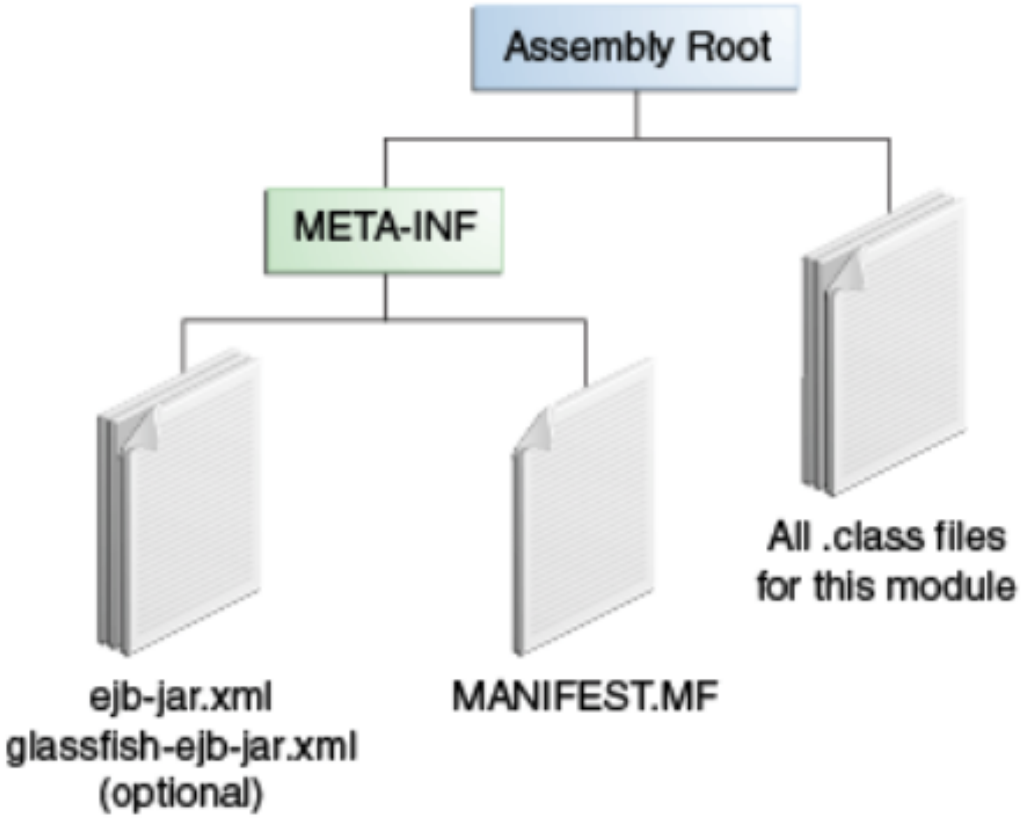
- Um pacote EAR contém:
  - módulos Java EE
  - descritores de implantação
  - instruções através de anotações

Módulos Java EE

- Tipos de módulos Java EE
  - Módulo EJB
    - Contém classes Java ou EJBs (JAR)
  - Módulo Web
    - Contém componentes Web (JSP, Servlets, etc.) (WAR)
  - Módulo de aplicação cliente
    - Contém classes Java (JAR)
  - Módulo de adaptador de recurso
    - Contém interfaces, classes e bibliotecas nativas (RAR)

Empacotando EJBs em um EJB JAR

- Um EJB JAR é portável e pode ser utilizado em várias aplicações
- Uma aplicação Java EE pode conter **um ou mais** módulos EJB (dentro do EAR correspondente)



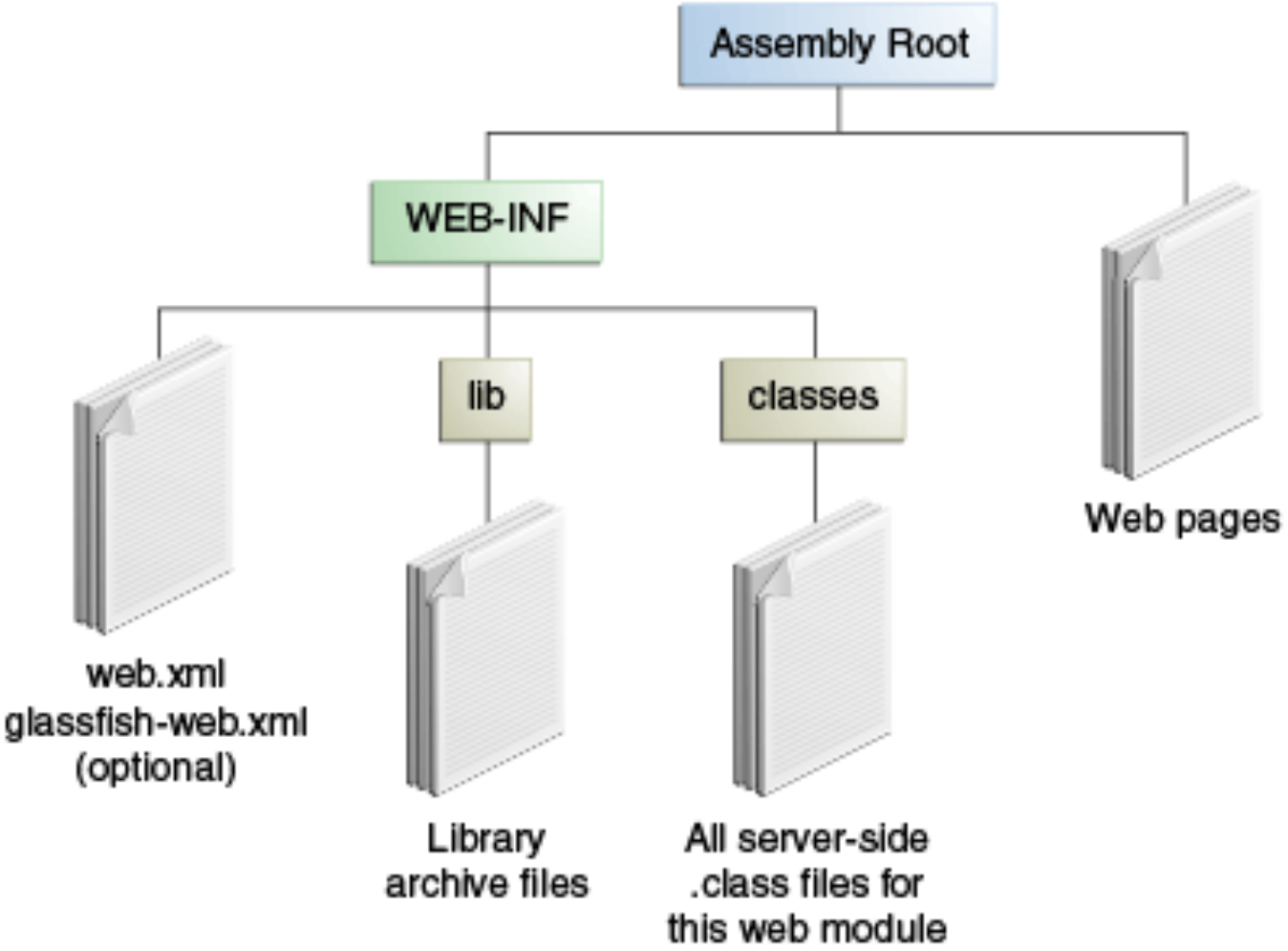
Empacotando EJBs em um WAR

- Essa opção simplifica o empacotamento da aplicação correspondente
  - Quando a aplicação é exclusivamente Web
- Deve estar no diretório **WEB-INF/classes**

Empacotando um Módulo Web

- Contem os componentes Web e os arquivos de conteúdo Web estático (imagens, etc.)
- Como subdiretórios de WEB-INF (padrão)
  - **classes** – classes Java em geral, Servlets, EJBs, etc.
  - **lib** – arquivos JAR necessários
  - Além de descritores de implantação como o **web.xml** e o **ejb-jar.xml**

Estrutura de um WAR



Empacotando Adaptadores de Recursos

- Armazenado em um arquivo RAR
  - Contendo: arquivos XML, classes Java, e outros elementos relativos a aplicação JCA (*Java EE Connector Architecture*)

- Pode fazer parte de um EAR