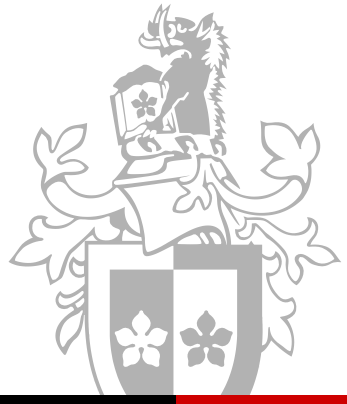
COS30041 Creating Secure and Scalable Software

Lecture 04b Intro. to Enterprise JavaBean (EJB)



SWINBURNE UNIVERSITY OF TECHNOLOGY

Learning Objectives

- After studying the lecture material, you will be able to
  - ☐ Understand and explain what a Enterprise JavaBean is
  - ☐ Understand and explain the benefits of EJB
  - ☐ Understand and explain the composition of EJB

Roadmap

- **Distributed Objects and Middleware**
- Enterprise JavaBean, EJB

Distributed Object

- An object that is callable from a remote system
- These objects are the foundation of Enterprise JavaBean (EJB)
- A typical client interacting a distributed object
  - See next slide

Example (local object & method)

```
public class HelloWorldAppClient {
  public static void main(String[] args) {
            HelloWorld hw= new HelloWorld ();
            String result = hw. getGreetings("Peter");
            System.out.println(result);
  }
}
```

```
public class HelloWorld {
  public String getGreetings(String name) {
            return "Hello," + name + "!";
  }
}
```

Remote object - hw: HellowWorld?

Distributed Object – Client Interaction

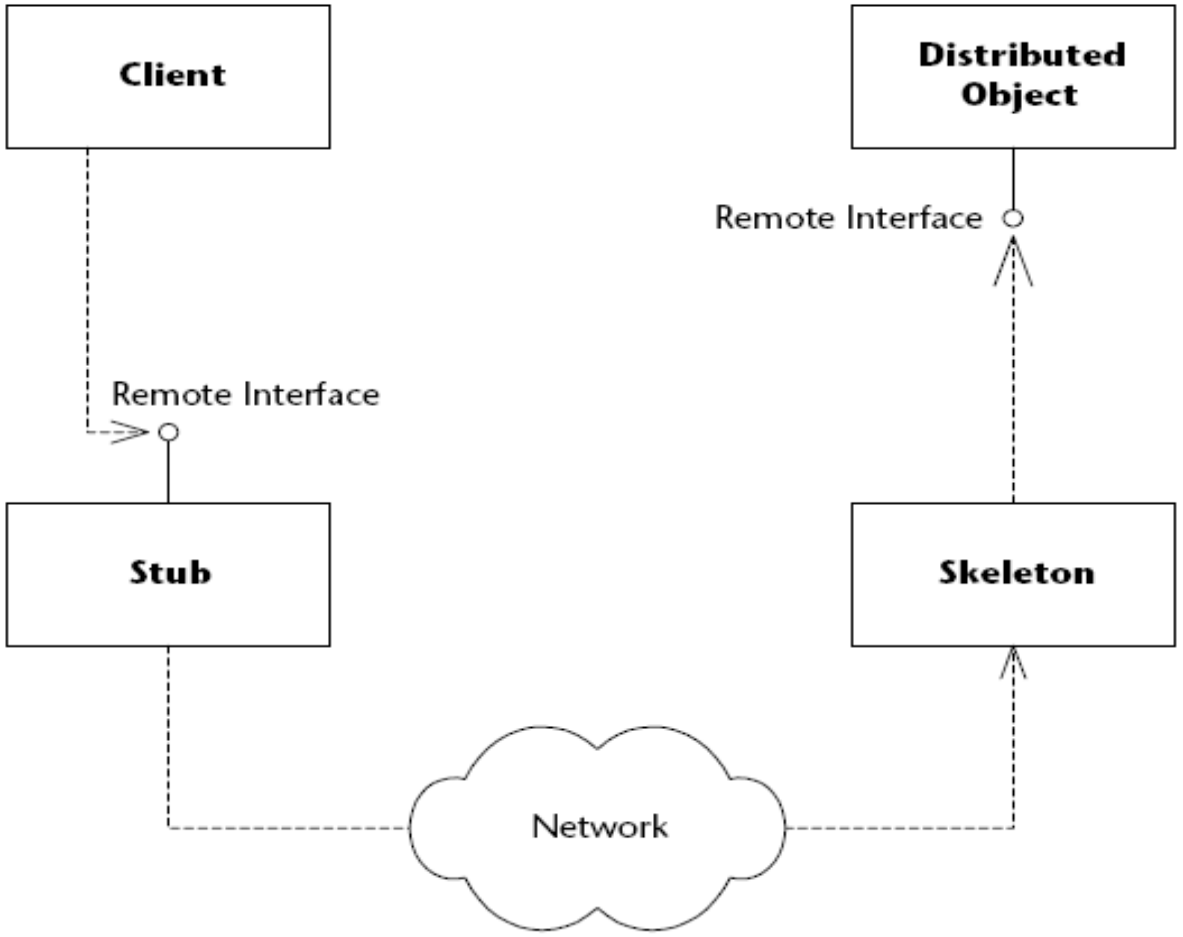


Figure 2.2 Distributed objects.

Standard distributed objects

See Fig. 2.2 of [MEJB-old,p.31]

Example (EJB – remote object)

```
import javax.ejb.EJB;
public class HelloWorldAppClient {
@EJB
private static HelloWorldRemote helloWorld;
private static HelloWorldBeanRemote
  public static void main(String[] args) {
            HelloWorld hw= new HelloWorld ();
            String result = helloWorld . getGreetings("Peter");
            System.out.println(result);
  }
}
```

```
import javax.ejb.Stateless;
@Stateless
public class HelloWorld implements HelloWorldRemote {
  public String getGreetings(String name) {
            return "Hello," + name + "!";
  }
}
```

Distributed Object and Middleware

- Two Types of Middleware
- Explicit Middleware
  - Developers explicitly specify the services required and needs to do the programming properly
- Implicit Middleware
  - Services are implicitly provided by the server
  - The server will intercept the request and manage the services properly

Distributed Object and Explicit Middleware

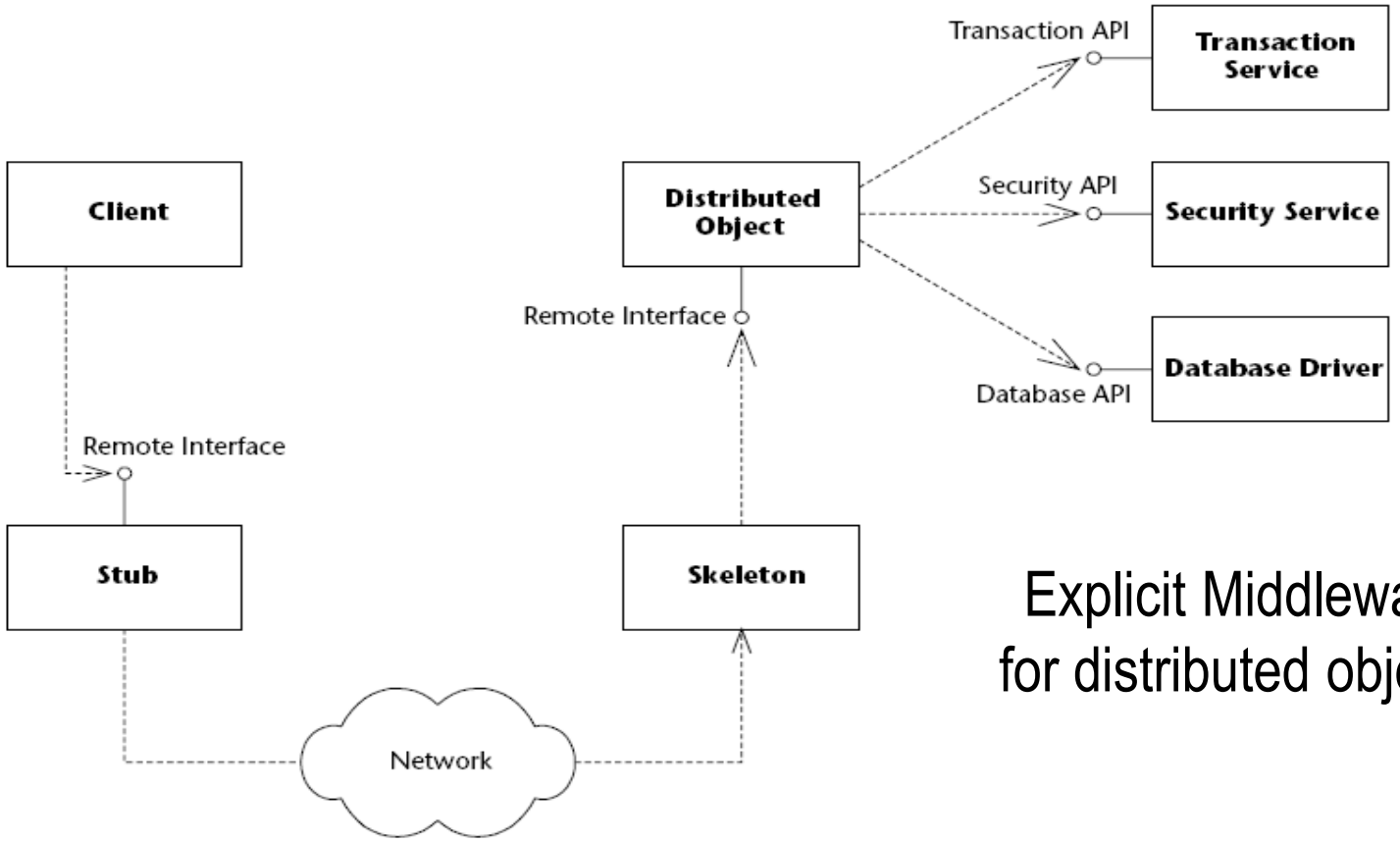


Figure 2.3 Explicit middleware (gained through APIs).

Explicit Middleware for distributed objects

See Fig. 2.3 of [MEJB-old,p.33]

Distributed Object and Implicit Middleware

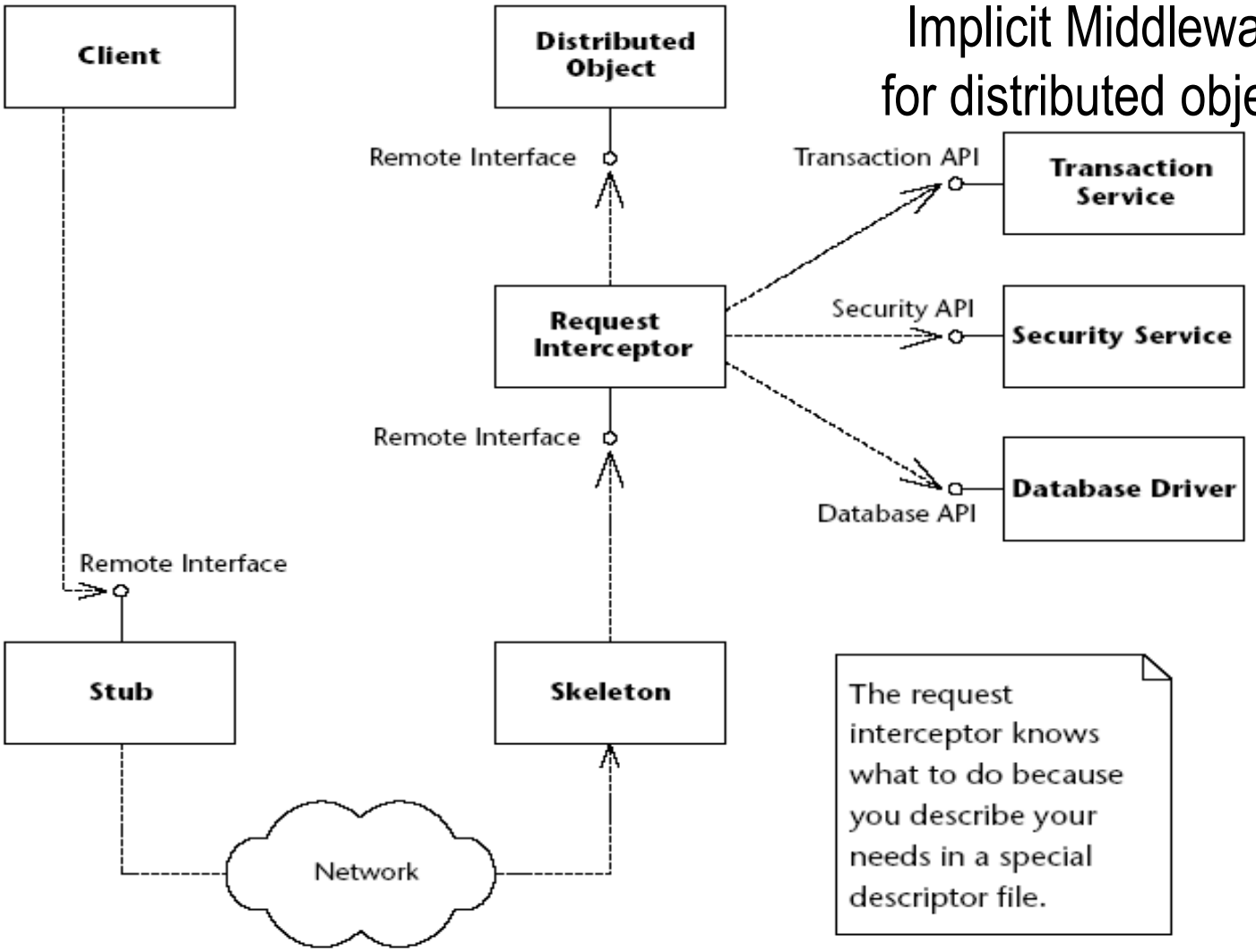


Figure 2.4 Implicit middleware (gained through declarations).

See Fig. 2.4 of [MEJB-old,p.34]

Advantages of Implicit Middleware

- Easy to write
- Easy to maintain
- Easy to support
- Comments:
  - EJB makes heavy use of the implicit middleware concepts
  - Most of the development framework uses implicit middleware for ease of development

Roadmap

- Distributed Objects and Middleware
- **Enterprise JavaBeans, EJB**

Enterprise JavaBeans (Enterprise Beans)

- A server-side software component that can be deployed in a distributed multi-tier environment
- Can compose one or more Java objects
- Different clients of the bean deal with a single exposed component interface
- The enterprise bean and the interface must conform to the EJB specification

Why EJB?

- Can quickly and easily construct server-side components in Java by leveraging a prewritten distributed infrastructure (Java EE)
- Design to support application portability and reusability across different enterprise middleware services

Different Types of EJB

- Session Bean: Bean that models the business processes
  - Examples:
    accessing bank account
    verifying credit card details
    preparing an invoice
- Message Driven Bean: Bean that handles messages
  - Examples:
    email messages
    JMS messages

Typical EJB Architecture (3 Tier vs 4 Tier)

3 Tier Architecture

Client Tier

←→ EJB Tier (Business Tier)

←→ EIS Tier (Database Tier, or Data Tier)

4-Tier Architecture

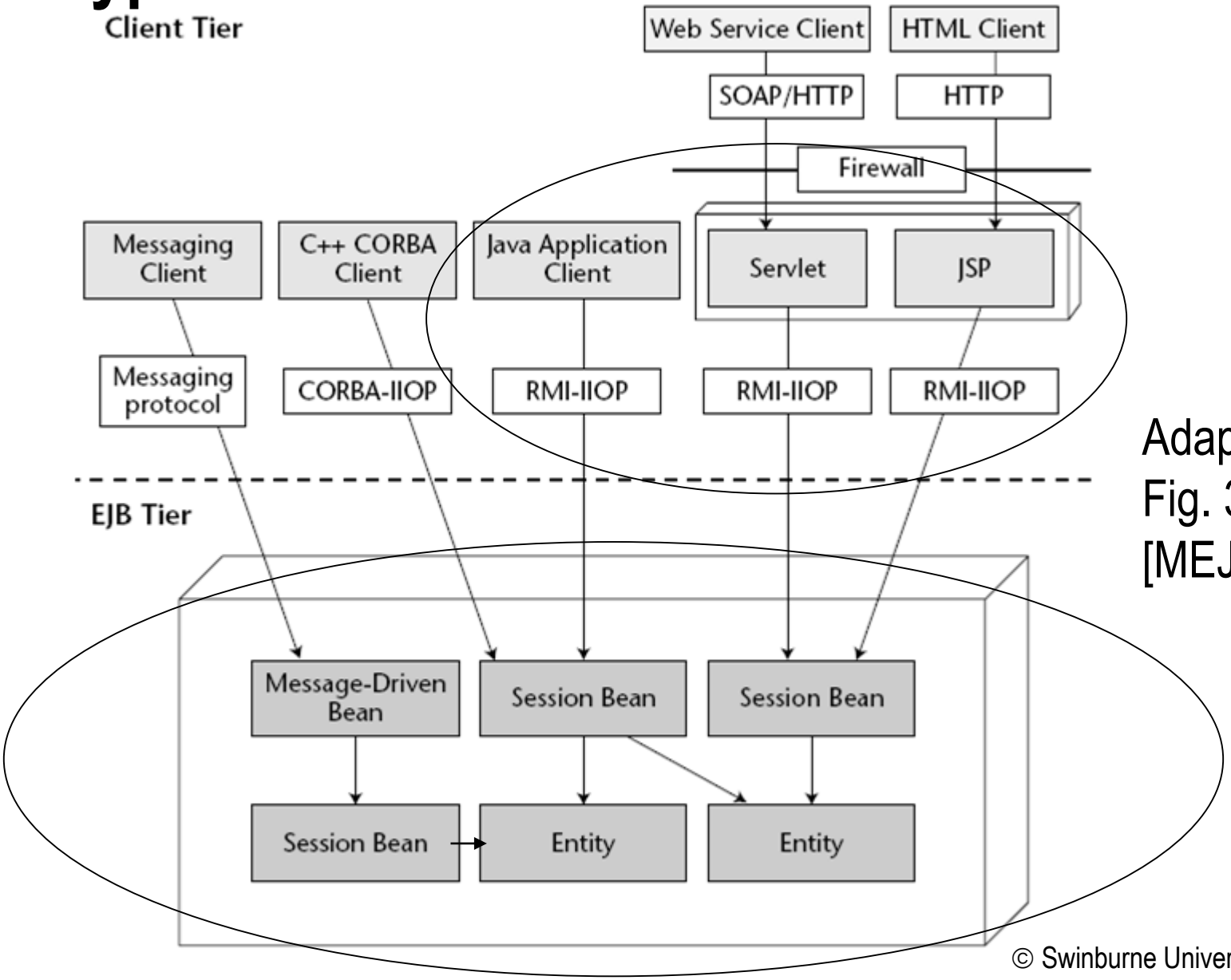
Web Browser

←→ Web Server

←→ EJB Tier

←→ EIS Tier

Typical Client Interactions



Adapted from Fig. 3.2 of [MEJB,p.63]

Enterprise JavaBeans Composition

- **The Enterprise Bean Class***
  - The actual implementation class
- **The Remote Interface*, if any**
  - Expose the business methods of the EJB
- **The Local Interface*, if any**
  - Local counterparts for the EJB Remote Interface

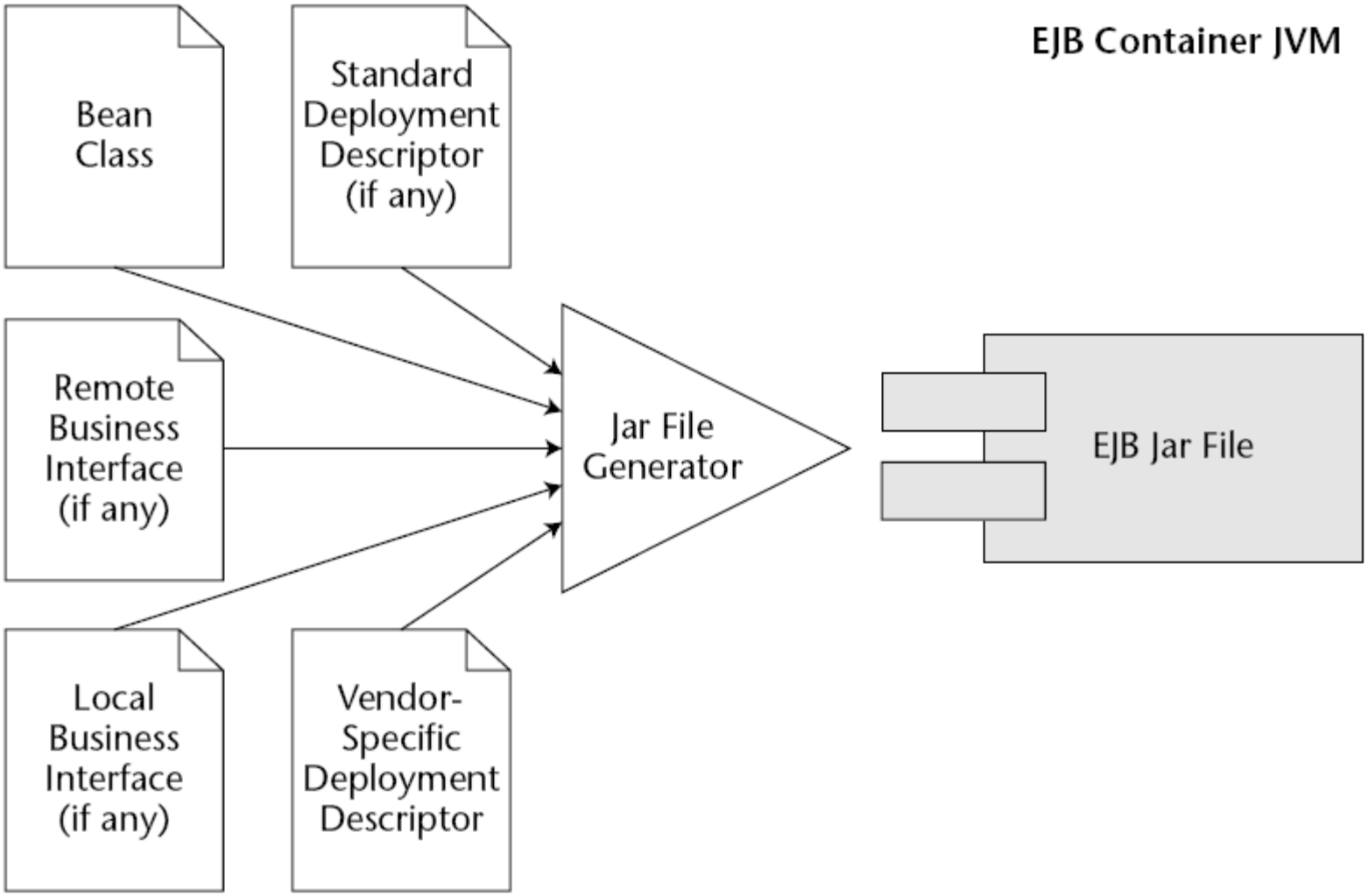
*Note: Developers program these classes

Enterprise JavaBeans Composition (cont'd)

- The Deployment Descriptor
  - Describes the middleware service requirements of the EJB
  - Example
    - Bean management and lifecycle requirements
    - Persistence requirements
    - Security requirements
- Vendor-specific files
- The EJB-JAR file
  - The jar file for the entire EJB

Note: Application Server or Some IDE do these automatically

Enterprise JavaBeans Composition (cont'd)



See Fig. 3.5 of [MEJB,p.82]

References

- [MEJB] R.P. Sriganesh, G. Brose, M. Silverman (2008) *Mastering Enterprise JavaBeans 3.0*, 4th ed., John Wiley & Sons
  - Chapter 3
- [MEJB-old] E. Roman, R.P. Sriganesh, G. Brose (2005) *Mastering Enterprise JavaBeans*, 3rd ed., John Wiley & Sons
  - Chapter 2
  - Some diagrams in this slide set are from [MEJB-old], which can also be downloaded from www.theServerSide.com