Part 4: Web Services and EIS Technologies

In this part we introduce the concepts of web services, and introduce some Java-based and other technologies that can be used to implement EIS applications.

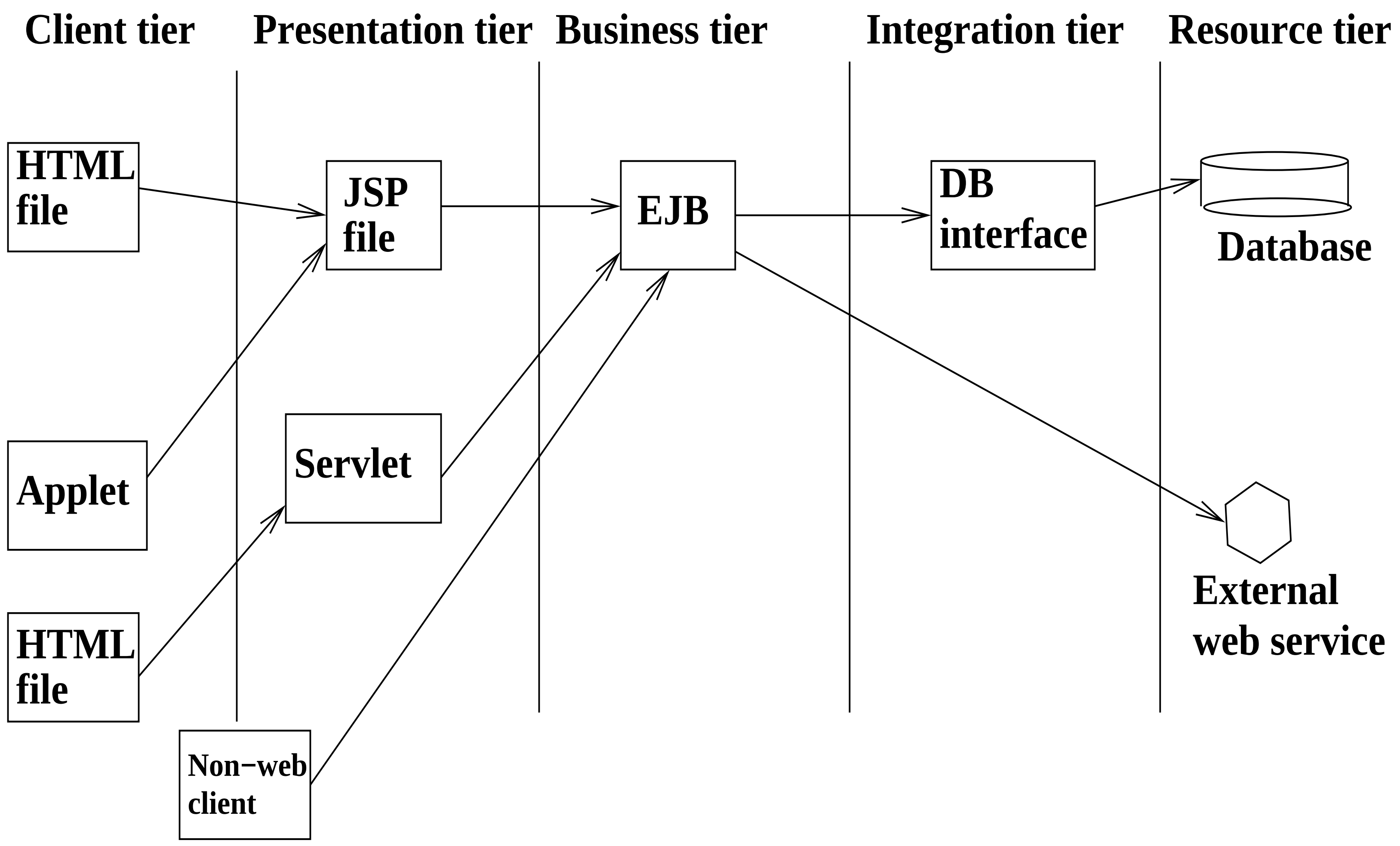
- 4.1: J2EE
- 4.2: Java EE
- 4.3: Web services and web service design patterns
- 4.4: RESTful web services

4.1: J2EE: Java 2 Enterprise Edition

J2EE is a Java framework for distributed enterprise systems, which includes:

- Servlets, JDBC and JSP.
- Enterprise Java Beans (EJB) – representing distributed business components, possibly with persistent data.
- Java Naming and Directory Interface (JNDI) – an interface to support naming and directory services, such as the Java RMI registry for locating remote methods.
- JavaMail – an API for platform-independent mailing and messaging in Java.

J2EE uses the five-tier architecture defined in Part 3.



Typical J2EE system structure

Enterprise Java Beans

EJBs are core mechanism for carrying out business logic on server side of a J2EE-based system. Two forms of EJB are:

Session bean A business component: dedicated to a single client; that lives only for the duration of the client's session; that is not persistent; that can be used to model statefull or stateless interactions between the client and business tier components.

Entity bean A coarse-grained business component which: provides an object view of persistent data; is multiuser and long-lived.

EJB interfaces

Each EJB has a number of different interfaces which provide alternative ways that bean can be used by different clients:

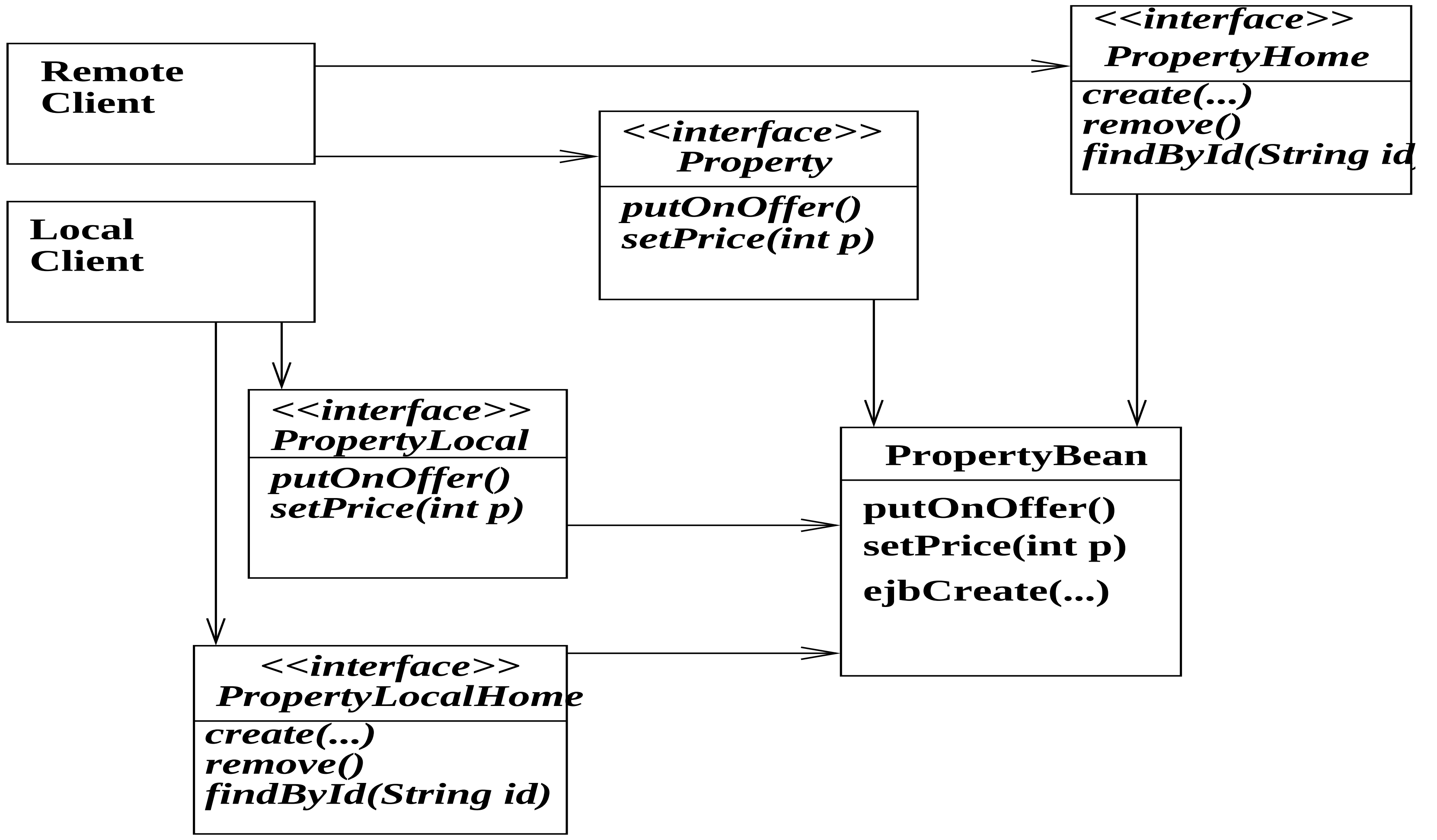
- *Remote interface* lists business operations specific to bean.

  In dating agency system, an operation to determine matching members for a user would be listed in remote interface of *Member* EJB.

- *home interface*, which lists lifecycle operations (creation, deletion) and methods (such as *findByPrimaryKey*) to return particular bean objects.

- *local interface*, listing business operations that can be accessed by local clients, ie, those executing in the same JVM as the EJB.

- *local home interface*, listing life-cycle and finder methods for local clients.



EJB interfaces for a *Property* entity

Using J2EE

J2EE provides a sophisticated environment for distributed and internet system construction, and for definition of web services.

However its complexity can lead to poor design practices, and a substantial amount of experience and familiarity with J2EE seems necessary to take full advantage of its features. Solutions to this are the definition of J2EE design patterns to express good design structures for J2EE in a reusable way, or to encode expert knowledge of J2EE into a code generation tool for J2EE applications.

4.2: Java Enterprise Edition Platform

Java EE is successor to J2EE. It incorporates all J2EE technologies, but simplifies their use, via defaults and annotations in source code to indicate what role a class plays in system and what interfaces (remote or local) are required for it.

Entity beans are now incorporated as Entity components in Java Persistence API:

```
@Entity
public class Student implements Serializable
{ @Id
  private String id;
  private String name;
  private Course course;

  public Student() { }
  public Student(String nme, Course crse)
  { name = nme; course = crse; }

  @ManyToOne
  public Course getCourse()  { return course; }
  public void setCourse(Course cx)  { course = cx; }
}
```

ManyToOne annotation indicates that relationship between *Student* and *Course* is of *-multiplicity at *Student* end and of 1-multiplicity at *Course* end.

This entity bean could be used by a session bean:

```
@Stateless
public class RegisterBean implements RegisterOps
{
  @PersistenceContext
  private EntityManager em;
  public Student registerStudent(String sname, String cname)
  { Course crse = em.find(Course.class,cname);
    Student stud = new Student(sname,crse);
    crse.getStudents().add(stud);
    em.persist(stud);
    return stud;
  }
}

@Remote
public interface RegisterOps
{ public Student registerStudent(String sname, String cname);
}
```

EntityManager class performs CMP tasks of synchronising entity and database data. Developer needs to include code to maintain mutually inverse relationships, such as ends *students* and *course* of *Student_Course* association in above example.

Java EE also provides simpler specification of web services, using annotations to declare that a component offers a web service.

Many other web and EIS platforms are based on J2EE/Java EE, eg., Struts, Spring, Hibernate.