Overview of lectures and practical work

- Lectures
  - Introduction to distributed systems and middleware
  - RMI-based distributed systems
  - Servlet-based distributed systems
  - Introduction to multi-tier distributed web applications

- Practical work
  - Programming distributed systems with RMI
  - Project on multi-tier distributed web applications

Motivations

- Processing a request on the server may successively involve several types of logic:
  - Data access logic
    - Example: read data from a persistent storage (e.g., a database)
  - Business logic
    - Example: use the read data to perform any application-specific processing
  - Presentation logic
    - Example: use the obtained result to build a user-friendly response to the client
Motivations

- These types of logic may be more or less heavy in terms of processing time.
- A unique server that hosts multiple types of logic may suffer from scalability issues in case of heavy workload (#concurrent web clients).
- Solution:
  - Separate the different types of logic in different servers
  - Multi-tier architecture

Overview of the multi-tier architecture

Multi-tier architecture

- Application servers
  - Goal: Simplify/Speed up business application development
  - Multi-tiers architecture
  - Host applications and provide them with services (persistence, security, ...)
- Java Enterprise Edition (formerly J2EE)
  - Developed by SUN since 1997
  - Based on Java
  - Many commercial/free implementations which may follow JEE specifications
    - Bea WebLogic,
    - IBM Websphere,
    - JBoss,
    - Jonas, ...

Multi-tier architecture (2)

- Web tier
  - Receives requests from web clients
  - Runs web components
  - May forward requests to the business tier
  - Returns web documents as responses (e.g. static HTML pages or dynamically generated web pages)
- Business tier
  - Receives requests from the web tier (may also be called directly)
  - Runs business components
  - May forward requests to the data access tier (through JDBC)
- Data access tier
  - Runs a database server
  - Receives requests from the business tier
Multi-tier architecture (3)

- Web components
  - JEE web components are either servlets or JSP pages.

- Some notes about JSP
  - Goal: Allows to build web responses in such a way that the static part is separated from the dynamic part
    - For the static parts of the web response, write regular HTML
    - For the dynamic parts of the web response, enclose code for the dynamic parts using special tags
  - How it works
    - A JSP page automatically gets converted to a normal Servlet
      - The static HTML is printed to the output stream associated with the servlet's `service` method while the dynamic part correspond to Java code
      - Build is performed automatically

JEE multi-tier systems

- Business components
  - Meets the needs of a particular business domain
    - Ex: banking, retail, finance, ...
  - There are three kinds of enterprise beans: session beans, entity beans, and message-driven beans
  - Managed by an EJB container
    - Provides non-functional services
      - Lifecycle management
      - Persistence
      - Security
      - Transactions
      - ...
  - EJB may be distributed
  - EJB are invoked through different protocols (ex: RMI)

A JSP example

```
<%! private int counter = 0; %>
<HTML>
<HEAD><TITLE>Hello</TITLE></HEAD>
<BODY>
<H1>Hello</H1>
<% counter++;
String pname;
pname = request.getParameter("name");
if (pname==null) {
  out.println("World");
} else {
  %>
Mister <%=pname%>
<% } // fin du else %>
</H1>
</BODY>
</HTML>
```
### JEE multi-tier systems

- **Business components**
  - **Session bean**
    - Represents a transient conversation with a client (stateful or stateless)
    - When the client finishes executing, the session bean and its data are gone
    - Front-end to entity beans
  - **Entity bean**
    - Represents persistent data stored in the database.
    - Persistence may be managed by the bean or by the container
    - Concurrency is managed by the container
  - **Message-driven bean**
    - Combines features of a session bean and a Java Message Service (JMS) message listener.
    - Allowing a business component to receive JMS messages asynchronously.

### Entity Bean example

```java
@Entity
public class Facture {
    @Id
    private String numfact;
    private Client client;
    public Facture() {
    }
    public Facture(String numfact) {
        this.numfact = numfact;
    }
    public void setMontant(double montant) {
        this.montant = montant;
    }
    public double getMontant() {
        return montant;
    }
    @ManyToOne
    public Client getClient() {
        return client;
    }
    public void setClient(Client client) {
        this.client = client;
    }
}
```

### Session Bean example

```java
@Stateless
@Remote
public class FacturationBean implements Facturation {

    @PersistenceContext
    private EntityManager entityManager = null;

    public void creerFacture(String numfact, double montant) {
        Facture fact = new Facture(numfact);
        fact.setMontant(montant);
        entityManager.persist(fact);
    }

    public Facture getFacture(String numfact) {
        return entityManager.find(Facture.class, numfact);
    }
}
```

### A setup example

```
JEE application server (JOnAS)

Servlets
JSP
EJB
DBMS
Oracle
JDBC
JOBC
Servlet container (Tomcat)
RMI
RMI / JMS/ …

WebServer (Apache)

client
```

---

F. Gaud  Distributed systems & Middleware

17

JEE multi-tier systems

- **Business components**
  - **Session bean**
    - Represents a transient conversation with a client (stateful or stateless)
    - When the client finishes executing, the session bean and its data are gone
    - Front-end to entity beans
  - **Entity bean**
    - Represents persistent data stored in the database.
    - Persistence may be managed by the bean or by the container
    - Concurrency is managed by the container
  - **Message-driven bean**
    - Combines features of a session bean and a Java Message Service (JMS) message listener.
    - Allowing a business component to receive JMS messages asynchronously.

---

F. Gaud  Distributed systems & Middleware

18

Entity Bean example

```java
@Entity
public class Facture {
    @Id
    private String numfact;
    private Client client;

    public Facture() {
    }
    public Facture(String numfact) {
        this.numfact = numfact;
    }
    public void setMontant(double montant) {
        this.montant = montant;
    }
    public double getMontant() {
        return montant;
    }
    @ManyToOne
    public Client getClient() {
        return client;
    }
    public void setClient(Client client) {
        this.client = client;
    }
}
```

---

F. Gaud  Distributed systems & Middleware

19

Session Bean example

```java
@Stateless
@Remote
public class FacturationBean implements Facturation {

    @PersistenceContext
    private EntityManager entityManager = null;

    public void creerFacture(String numfact, double montant) {
        Facture fact = new Facture(numfact);
        fact.setMontant(montant);
        entityManager.persist(fact);
    }

    public Facture getFacture(String numfact) {
        return entityManager.find(Facture.class, numfact);
    }
}
```

---

F. Gaud  Distributed systems & Middleware

20

A setup example

```
JEE application server (JOnAS)

Servlets
JSP
EJB
DBMS
Oracle
JDBC
JOBC
Servlet container (Tomcat)
RMI
RMI / JMS/ …

WebServer (Apache)

client
```

---

F. Gaud  Distributed systems & Middleware
Load balancing

Incoming lectures and practical work on middleware

- Lectures
  - Introduction to distributed systems and middleware
  - RMI-based distributed systems
  - Servlet-based distributed systems
  - Introduction to multi-tier distributed Internet services

- Practical work
  - Programming distributed systems with RMI
  - Project on multi-tier distributed web applications

References

- This lecture is extensively based on:
  - Sun Microsystems. The J2EE Tutorial
    http://java.sun.com/j2ee/1.4/docs/tutorial/
  - Jonas documentation
    http://wiki.jonas.objectweb.org/xwiki/bin/view/Main/WebHome
  - Courses given by D. Donsez
    http://membres-iglab.imag.fr/donsez/cours/
  - Courses given by S. Bouchenak
    http://sardes.inrialpes.fr/~bouchena/
  - Courses given by R. Lachaize
    http://sardes.inrialpes.fr/~rlachaiz
  - Courses given by P.Y. Gibello