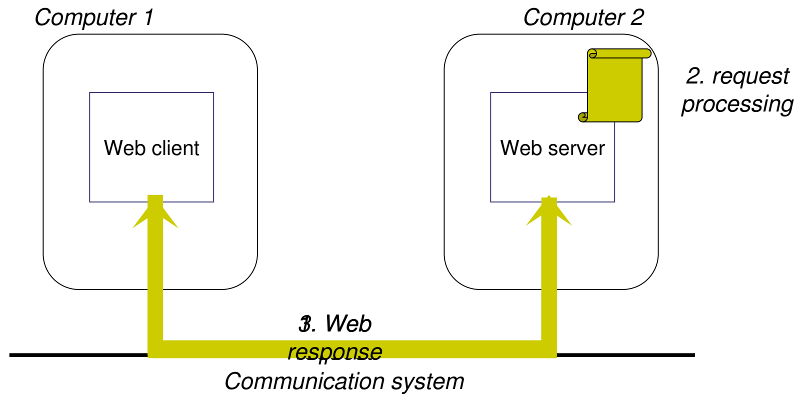
Distributed Multi-Tier Web Applications

Sara Bouchenak

Sara.Bouchenak@imag.fr
http://sardes.inrialpes.fr/~bouchena/teaching/

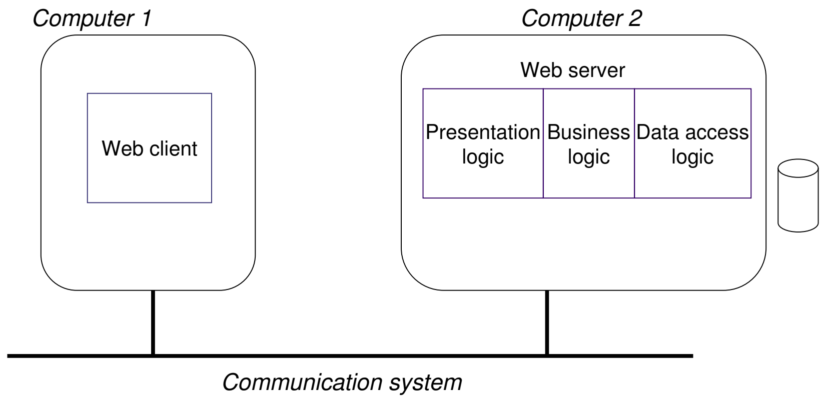
Introduction – Web applications



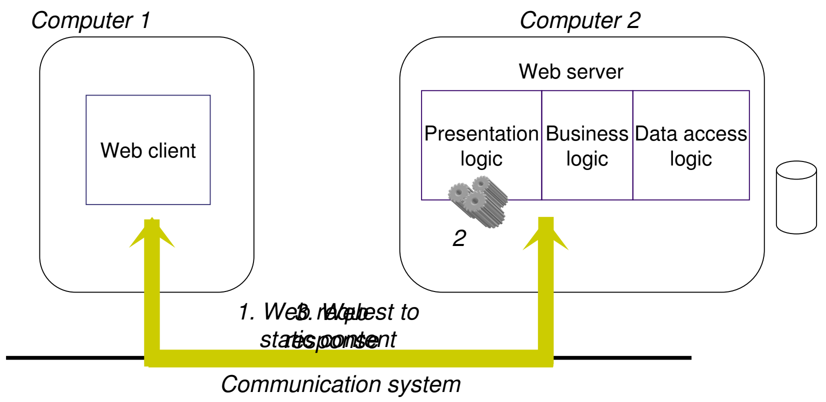
Motivations

- Processing a request on the server may sucessiveley 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

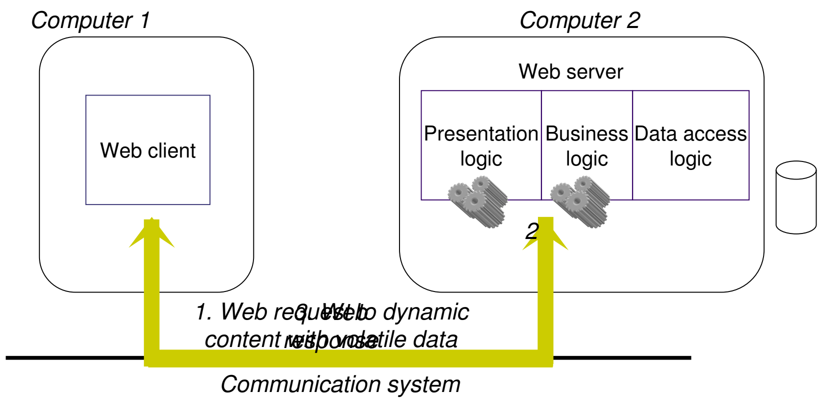
Example 1



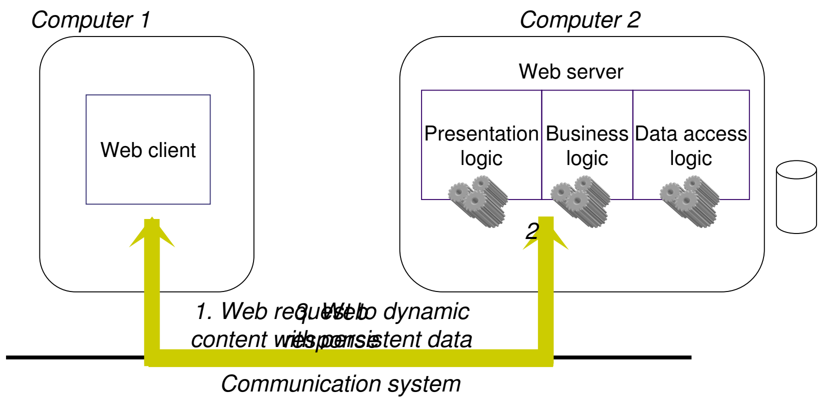
Example 1



Example 2



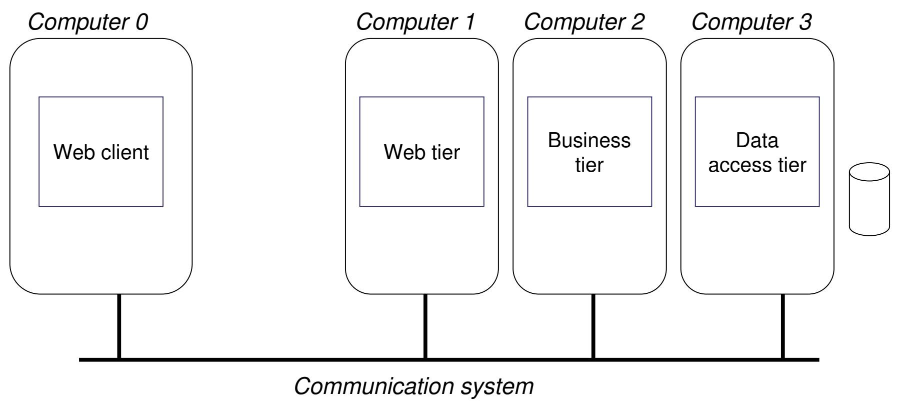
Example 3



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

- Java 2 Enterprise Edition
- Web tier
  - Run a web server
  - Receive requests from web clients
  - Run web components
  - May forward requests to the business tier
  - Return web documents as responses (e.g. static HTML pages or dynamically generated web pages)
- Business tier
  - Run an application server
  - Receive requests from the web tier
  - Run business components
  - May forward requests to the data access tier (via JDBC)
- Data access tier
  - Run a database server
  - Receive requests from the business tier

J2EE multi-tier systems

- Web components
  - J2EE web components are either servlets or pages created using JSP technology (JSP pages)
  - *Servlets* are Java programming language classes that dynamically process requests and construct responses
  - *JSP pages* are text-based documents that execute as servlets but allow a more natural approach to creating static content
  - Static HTML pages and applets are bundled with web components during application assembly

J2EE multi-tier systems (2)

- Business components
  - Business code, i.e. the logic that solves or meets the needs of a particular business domain such as banking, retail, or finance, is handled by enterprise beans running in the business tier
  - There are three kinds of enterprise beans: session beans, entity beans, and message-driven beans
  - A *session bean* represents a transient conversation with a client. When the client finishes executing, the session bean and its data are gone
  - An *entity bean* represents persistent data stored in one row of a database table. If the client terminates or if the server shuts down, the underlying services ensure that the entity bean data is saved
  - A *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

A simple example

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class MyServlet extends HttpServlet {
    public void doGet(HttpServletRequest req, HttpServletResponse res)
           throws ServletException, IOException {
           // Use "request" to read incoming HTTP headers  and HTML form data
           // (e.g. data the user entered and submitted)
           ...
           // Perform any internal processing for generating dynamic results
           ...
           // Use "response" to specify the HTTP response line and headers
           // (e.g. specifying the content type).
           PrintWriter out = response.getWriter();
           // Use "out" to send content to browser
           ...
    }
...
```

A simple example (2)

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class MyServlet extends HttpServlet {
    public void doGet(HttpServletRequest req, HttpServletResponse res)
           throws ServletException, IOException {
           // Use "request" to read incoming HTTP headers  and HTML form data
           // (e.g. data the user entered and submitted)

           String accountIdStr = req.getParameter("accountId");
           int accountId = Integer.parseInt(accountIdStr);

           if (accountId != null) {

                      ...
           }

           ...

    }
```

A simple example (3)

```
import java.sql.*;

public class MyServlet extends HttpServlet {
    public void doGet(HttpServletRequest req, HttpServletResponse res)
           throws ServletException, IOException {
           ...
           // Perform any internal processing for generating dynamic results
           float balance = 0;
           Connection conn = DriverManager.getConnection(url, user, password);
           Statement stmt = conn.createStatement();
           ResultSet rs =stmt.executeQuery("SELECT balance FROM accounts WHERE id="
                   + accountId);
           try {
              if (rs.next())
                 balance = rs.getFloat("balance");
              rs.close(); stmt.close();
            } catch (Exception e) {
              e.printStackTrace();
            }
           ...
    }
```

A simple example (4)

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class MyServlet extends HttpServlet {

    public void doGet(HttpServletRequest req, HttpServletResponse res)
           throws ServletException, IOException {
           ...

           res.setContentType("text/html");
           PrintWriter out = res.getWriter();

           out.println("<HTML>");
           out.println("<HEAD> <TITLE> Account " + accountId + "</TITLE></HEAD>");
           out.println("<BODY>");
           out.println("Current balance is " + balance);
           out.println("</BODY>");
           out.println("</HTML>");
           out.close();
    }

}
```

J2EE features

- Java Servlet technology
- JavaServer Pages technology
- Enterprise JavaBeans technology
- Java Message Service
- Java Transaction
- JavaMail
- Java API for XML processing
- Java API for XML-based RPC
- Java DataBase Connectivity (JDBC)
- Java Naming and Discovery Interface (JNDI)
- Java authentication and authorization service

Other features of distributed Web applications

- Caching
- Prefetching
- Partitioning
- Replication
- Load balancing
- Cloud computing: toward on-demand remote and elastic applications

References

- Sun Microsystems. The J2EE Tutorial http://java.sun.com/j2ee/1.4/docs/tutorial/

Agenda

Week	Wednesday, 13:30 – 18:30
S5	Introduction to distributed systems and middleware (CM), S. Bouchenak, 13:30 – 15:00 Introduction to JDBC (CM), C. Labbé, 15:15 – 16:45
S6	RMI-based distributed systems (CM), S. Bouchenak , 13:30 – 15:00 RMI-based distributed systems (TD), S. Bouchenak, S. Gueye, 15:15 – 18:30
S7	Interruption week
S8	Servlet-based distributed systems (CM), S. Bouchenak , 13:30 – 15:00 RMI-based distributed systems (TD), S. Bouchenak , S. Gueye, 15:15 – 18:30
S9	Multi-tier distributed systems (CM), S. Bouchenak , 13:30 – 15:00 **Servlet-based distributed systems (TD), S. Bouchenak , S. Gueye, 15:15 – 18:30**
S10	Introduction to transactions (CM), C. Labbé, 13:30 – 15:00 Introduction to transactions (TD), C. Labbé, S. Gueye, 15:15 – 18:30
S11	–
S12	Presentation of the project (CM), S. Bouchenak , 13:30 – 15:00 Multi-tier distributed systems (TD), S. Bouchenak , S. Gueye, 15:15 – 18:30
S13	–
S14	Project, S. Bouchenak & C. Labbé & S. Gueye, 13:30 – 16:45