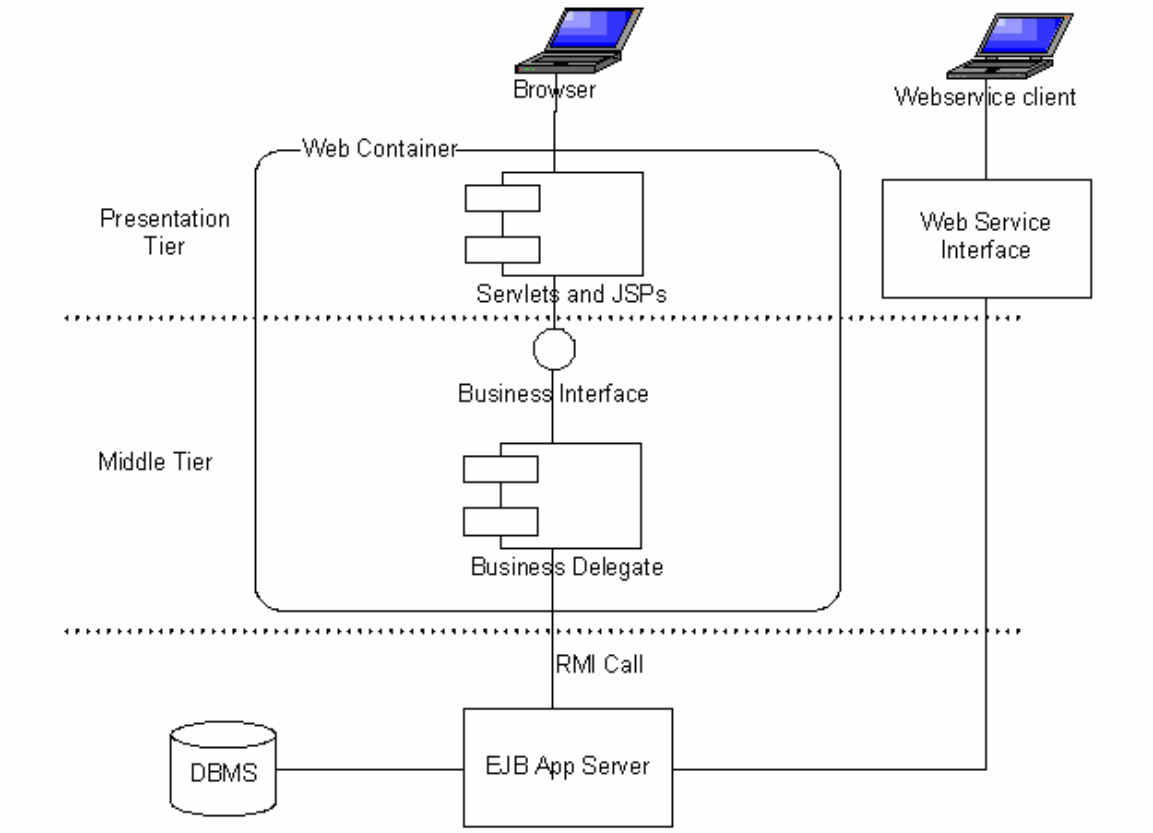
Distributed vs Non-distributed Architecture

Is distributed architecture more scalable than non-distributed architecture?

Many J2EE architects and developers tend to assume distributed architecture offers unmatched scalability. However this assumption is questionable.

Distributed Architecture

Distributed architecture splits the presentation tier and middle tier physically and logically to run in different servers.



This is a complex architecture with significant overhead. The architecture uses RMI between presentation tier and middle tier. This makes remote invocation a major determinant of performance and design consideration. It forces to minimize number of remote calls.

Strengths of this architecture

- It can support different type of clients by providing a shared middle tier.
- It permits the distribution of application components across different physical servers.

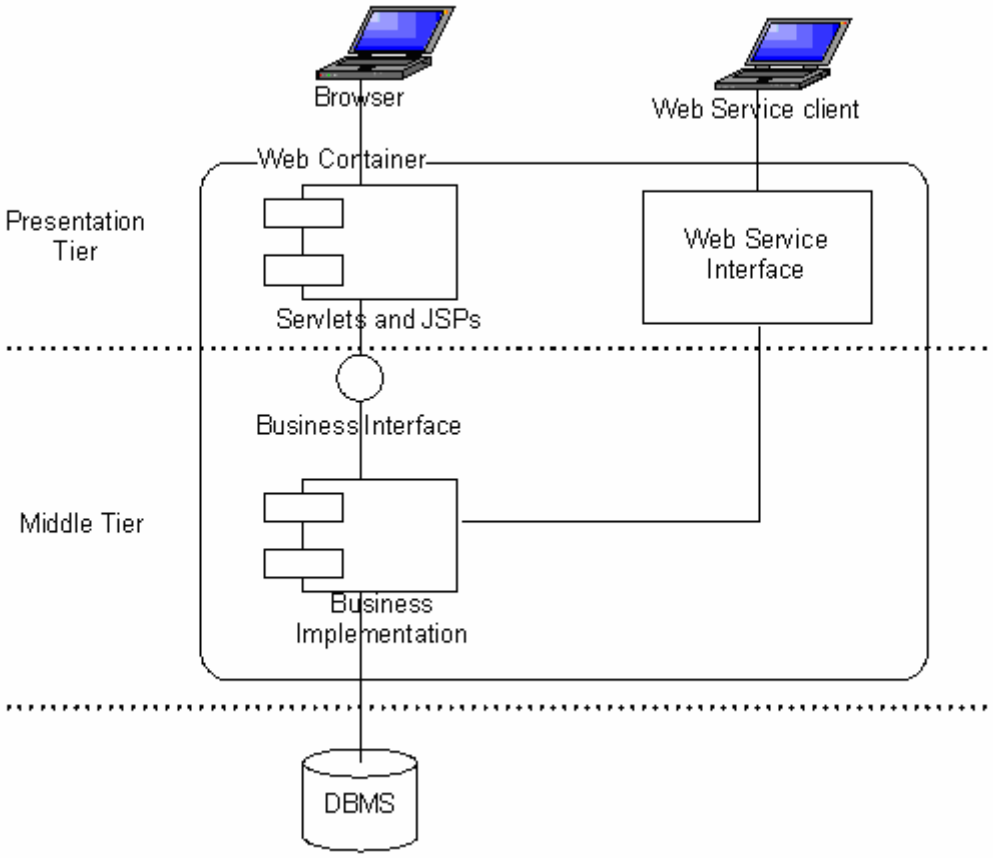
Weakness of this architecture

- It is a complex approach should be chosen only if the requirement dictates.
- It affects performance. Remote method calls can be hundred of times slower than local calls. This will determine the overall performance of the application.

- Distributed applications are hard to test and debug, because it is heavily dependent on container service.
- OO design is severely hampered by the central use of RMI.
- Exception handlings are more complex. Must handle both application failure and transport failure.
- Should rely on third party application server.
- Complex to deploy.

Non-Distributed Architecture

In this architecture, presentation tier and middle tier of the application run in the same server. So the business components can be accessed locally without remote calls; this will improve significant performance of the overall application. However it is vital that presentation tier and middle tier are kept logically distinct. The main risk in the web application is that blurred responsibilities between presentation and business logic components.



Strengths of this architecture

- Simplest architecture for web applications
- Application can be developed faster
- Can implement good OO design
- Scales well. Can cluster to handle larger number of request.
- Less cost, many open-source stable web containers are available.

Weakness of this architecture

- This architecture supports only web interface clients.
- The whole application run in a single server while it boost performance, components cannot freely allocated to different physical layers.
- Need to write code to manage transactions.
- Need to write code for concurrent threading issues.

In most cases J2EE is used to build web applications. Thus, a J2EE web-container can provide the entire infrastructure required by many applications. Except entity bean all other services can be used in web-container such as JMX, JMS, JDBC, JavaMail, Database Transaction and Connection Pooling.