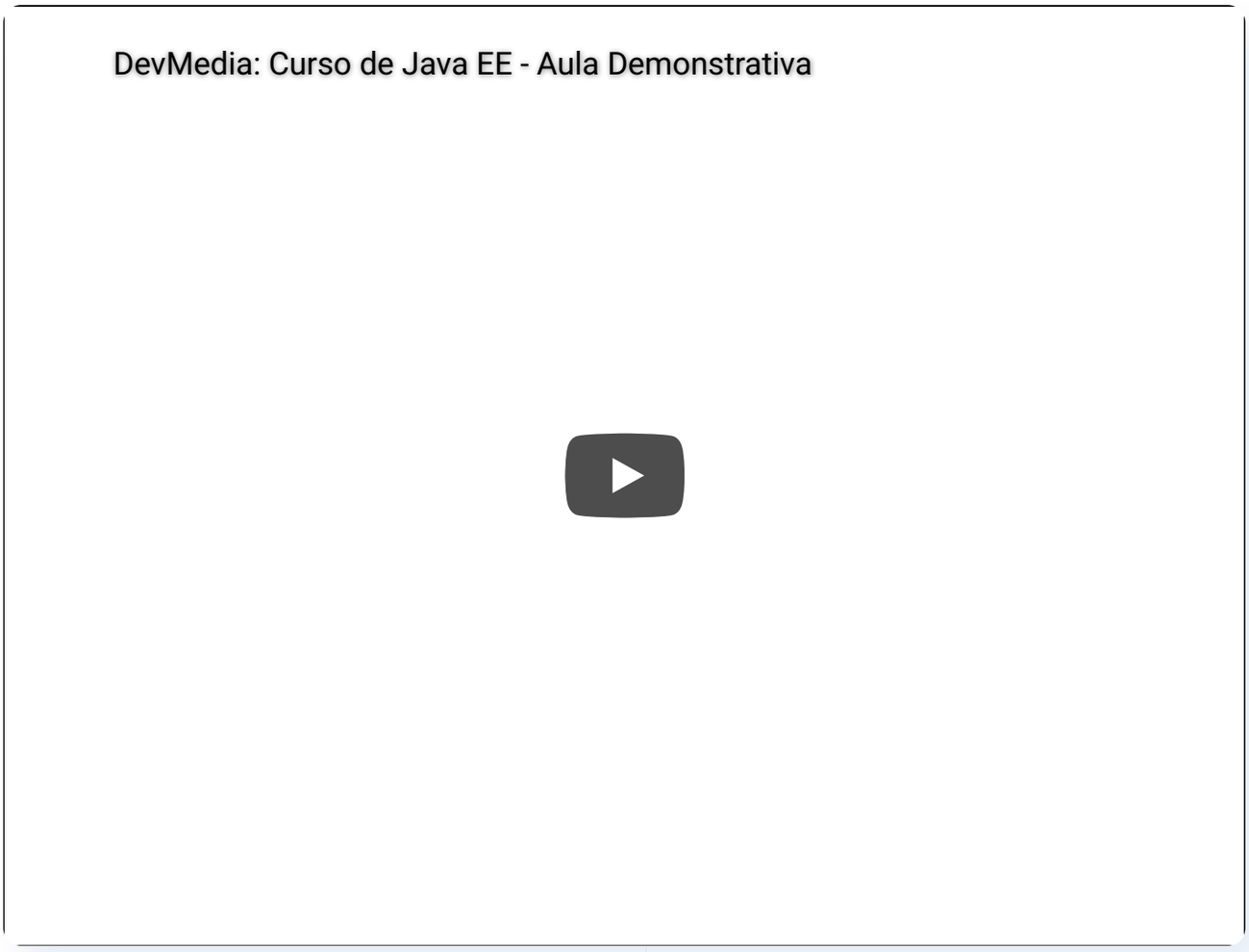
Por exemplo, existem sistemas antigos que geram problemas no momento que obtém um número elevado de acessos simultâneos ou até mesmo no armazenamento de grande quantidade de informações. Para acabar com esse tipo de problema, a tecnologia Java Enterprise Edition, mais conhecida como Java EE vêm para ajudar a facilitar o ciclo de vida desses sistemas.

O que é Java EE?

O Java EE é uma plataforma ou ambiente para desenvolvimento de aplicações de grande porte e aplicações web que possui bibliotecas e funcionalidades que implementam softwares baseados na linguagem Java.

Confira o vídeo abaixo:



Confira: Curso Completo de Java EE

sistema com segurança, escalabilidade, integridade, confiabilidade entre outros.

Essa plataforma tem uma série de tecnologias que têm objetivos distintos, sendo que as mais conhecidas são:

- **Servlets-** São componentes Java executados no servidor que tem o objetivo de gerar conteúdo (HTML e XML) dinâmico para web.

  > Confira nosso artigo sobre **Servlets em Java**

- **Java Server Pages (JSP) -** Especialização de servlets que permite aplicações em Java serem mais robustas e tenham facilidades no desenvolvimento.
- **Java Server Faces (JSF) -** É framework web com o padrão MVC (Model, View e Controller) baseado em Java que ajuda a simplificar o desenvolvimento de interfaces (telas do sistema) através de um modelo de UI.

  > Confira nosso artigo sobre **Desenvolvimento de aplicações web com Java Server Faces**

- **Java Persistence API (JPA) -** É uma API padrão do Java que usa o conceito de mapeamento objeto relacional, sendo utilizada para a persistência dos dados. Essa ferramenta traz muita produtividade, pois consegue desenvolver aplicações que trabalham com banco de dados sem escrever nenhuma linha SQL.

Curso: O que é Java Persistence API?

- Enterprise Java Beans (EJB) - São componentes que executam em um container de aplicação e que oferecem a facilidade e produtividade no desenvolvimento de componentes distribuídos, transacionados, seguros e portáteis.

Containeres e Componentes

Para fornecer a comunicação entre componentes, APIs, persistência, serviços entre outros, **o Java EE oferece os containeres para abrigá-los**. Os containeres são implementados pelos fornecedores de servidor de **aplicativos Java EE**, que disponibilizam um container para cada tipo de componente como: Applet, cliente de aplicativo, web e EJBs.

Relacionado: Cursos de Java DevMedia

Os componentes Web e EJB são distribuídos, gerenciados e executados em um servidor Java EE. Veja como funciona na **Figura 1**.

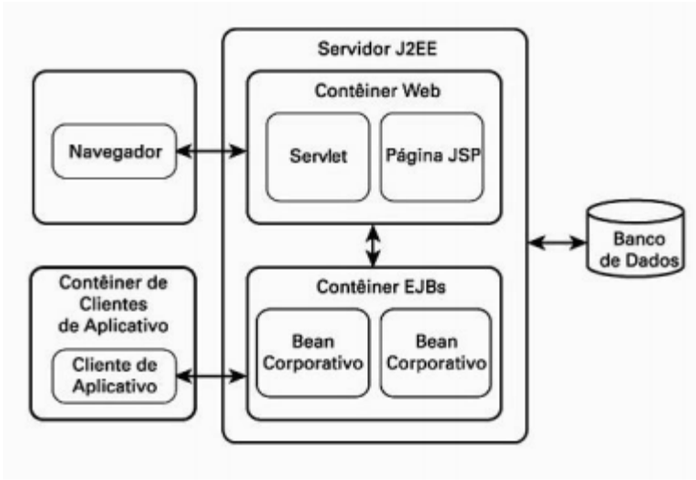


Figura 1. Arquitetura Lógica do Java EE.

Contêineres

Os contêineres armazenam, oferecem serviços (gerenciamento ciclo de vida, segurança, implementação, e comunicação com outros componentes) e recursos como um ambiente em tempo de execução compatível para os **componentes Java EE conseguirem trabalhar**.

Serviços dos Contêineres

Os contêineres fornecem a cada tipo de componente um conjunto de serviços:

- **Conectividade:** o tipo de conectividade atribuída é realizado pelos objetos distribuídos através de RMI (Remote Method Invocation) e COBRA. Porém, todo o ciclo da conectividade deve ser fornecido através dos protocolos HTTP e HTTPS.
- **Serviços de Diretório:** Conhecido como JNDI (Java Naming and Directory Interfaces), é uma API que acessa serviço de diretórios, sendo que cada serviço deve fornecer um provedor de serviços (Service Provider). Os **servidores Java EE** são obrigados a possuir esse tipo de serviço.
- **JDBC:** O acesso aos dados acontece através da API JDBC (Java Database Connection) que permite conectar com um banco de dados através de um driver específico para o tipo de banco (MySQL, Oracle, Postgresql e outros). Com essa API também consegue-se executar instruções SQL que auxiliam na manipulação das

- **JCA (Java Connector Architecture):** Conhecido também como conexão legada, ajuda a fornecer suporte de integrações de servidores de informações corporativas e sistemas legados. Esse componente ajuda muito no processamento de transação de ERPs e computadores de grande porte.
- **Segurança:** Usa APIs como JAAS (Java Authentication and Authorization Service) que é um serviço que ajuda na autenticação e autorização dos usuários no **sistema de plataformas Java EE**.
- **XML:** Permite suportar XML (eXtensible Markup Language) usando DOM (Document Object Model), documentos SAX (Simple API for XML) e XSLT (extensible Stylesheet Language Transformations).
- **Transações: Em um servidor Java EE é fornecido serviços de transações para todos seus componentes**. Geralmente, o container assume essa responsabilidade, porém a JTA (Java Transaction API) permite que o componente controle suas próprias transações.
- **JTA (Java Transaction API):** É uma API que permite trabalhar com transações independentes do gerenciador.
- **E-mail:** Esse componente permite a troca de mensagens através do serviço de mensagens do Java, conhecido como JMS (Java Message Service). Nesse serviço existe a API JavaMail que oferece manipular os envios e recebimentos de e-mail e troca de mensagens entre clientes. Especificamente, o JavaMail permite enviar e receber e-mail utilizando vários protocolos como: POP, SMTP e IMAP.

Veja também: Enviando email com JavaMail utilizando Gmail

- JMS (Java Message Service)**:** É uma API que oferece o mecanismo de criar, ler e armazenar mensagens. A JMS suporta dois tipos de trocas de mensagem:
  o **Ponto a ponto**: Esse tipo acontece quando um aplicativo envia uma mensagem para uma fila (destino preparado para receber as mensagens) e após isso, um aplicativo coleta essa mensagem. Por exemplo, quando a pessoa deixa uma mensagem na caixa postal.
  o **Publicação assinatura**: Esse modelo exige que os aplicativos clientes inscrevam-se para receber as informações sobre o assunto desejado. O processo de envio das informações acontece quando um aplicativo envia uma mensagem para o agente

de mensagens, sendo encaminhada a mensagem de forma imediata para todos os assinantes inscritos. Por exemplo, no canal do Youtube ou cadastro nos newsletters dos sites, ou seja, no momento em que é cadastrada uma nova informação, todos os usuários inscritos recebem em forma de mensagem de que foi realizada uma atualização de um novo conteúdo inserido no local.

Na **Figura 2** vemos os principais serviços do Java EE.

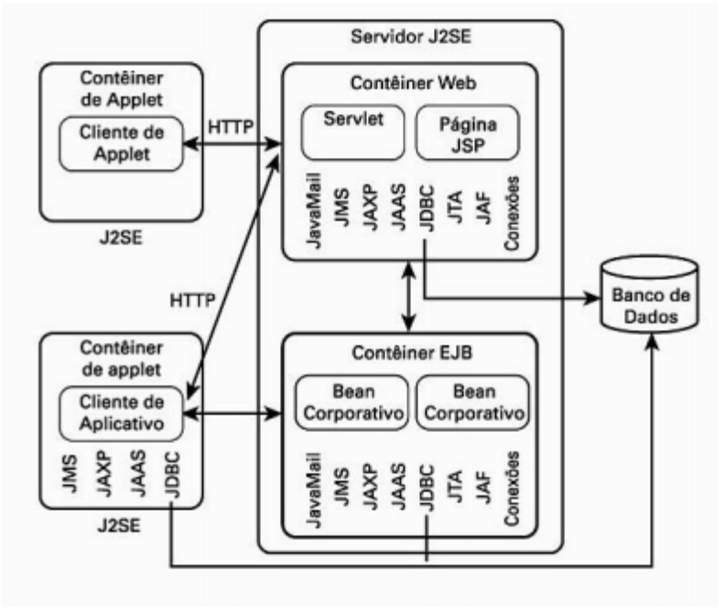


Figura 2. Plataforma Java EE com serviços.

Conhecendo os componentes no Java EE

Os componentes são divididos em EJBs e Componente Web. Os EJBs possuem outras subdivisões: Beans de sessão, Beans de Mensagem e Beans de Entidade.

Enterprise JavaBeans – EJB

Os EJBs são responsáveis por facilitar o encapsulamento e compartilhamento da lógica do negócio. Para expor a lógica de negócio para outros componentes conseguirem utilizar e facilitar a interação entre o EJB e o cliente é envolvido dois mecanismos: RMI (Invocação de métodos remotos) e a JNDI (Interface de atribuição de nomes e diretórios do Java). Observe a **Figura 3.**

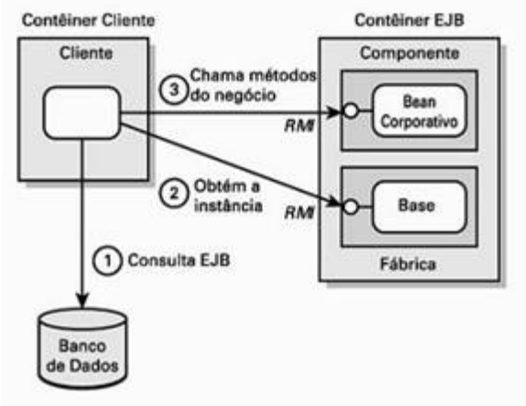


Figura 3. Uso de um cliente com a JNDI e RMI para acessar um EJB.

Conforme mostrado na **Figura 3**, um cliente usa a JNDI para procurar a localização do EJB, interagindo com a Fábrica (Base do EJB) para obter uma instância do EJB.

- **Beans de Sessão:** Esse tipo permite que a regra de negócio seja desenvolvida e depois implantada, independentemente da camada lógica de apresentação, ou seja, é uma extensão da funcionalidade do negócio de um cliente na camada intermediária. Um exemplo prático é quando há necessidade de desenvolver certas regras no login do sistema para tipos de usuários.
- **Beans de Entidade:** Esses beans representam um dado corporativo (entidade) durante os processos, sendo possível com que o cliente consiga acessar os dados e funcionalidade. Podem também conseguirem acessar banco dados ou sistemas de ERP, com o objetivo para reunir todas as informações corporativas que representam. Observe a **Figura 4.**

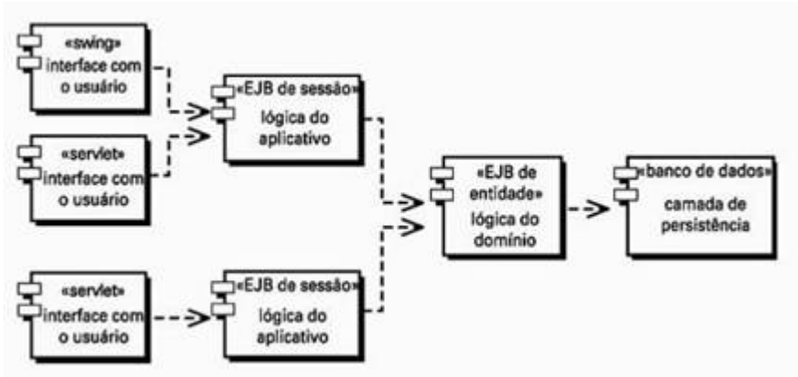


Figura 4. EJBs separando a lógica do negócio em lógico do aplicativo e do domínio

- **Beans de Mensagem:** Esse tipo é associado a uma fila de mensagens em particular que armazena todas as mensagens que chegam nessa fila, sendo distribuídas para

negócio, em vez de dados, acessando os dados exigidos através do JDBC ou beans de entidade.

Algumas Vantagens dos Componentes Corporativos

- **Eficiência:** Existem componentes que oferecem a agilidade no desenvolvimento nas telas de apresentação do sistema conhecidas como GUIs (interfaces do usuário) e com isso ajuda a identificarem a lógica de negócio e acesso aos dados.
- **Extensibilidade:** permite adicionar ou remover componentes em um aplicativo para oferecer mais funcionalidade.
- **Independência:** Permite ter a independência de linguagens.
- **Atualização do sistema:** O uso de componente permite a alteração sem afetar outros componentes do sistema.

Camadas

Na plataforma Java EE existem três tipos de camadas: Camada de Negócio, Camada de Apresentação e Camada Cliente.

Camada de Negócio

Essa camada também pode ser conhecida como camada física e tem o objetivo de encapsular a lógica do negócio ou EJBs que são usados pelos componentes da camada da apresentação que fornecem essa funcionalidade para usuários do aplicativo. Observe um exemplo na **Figura 5.**

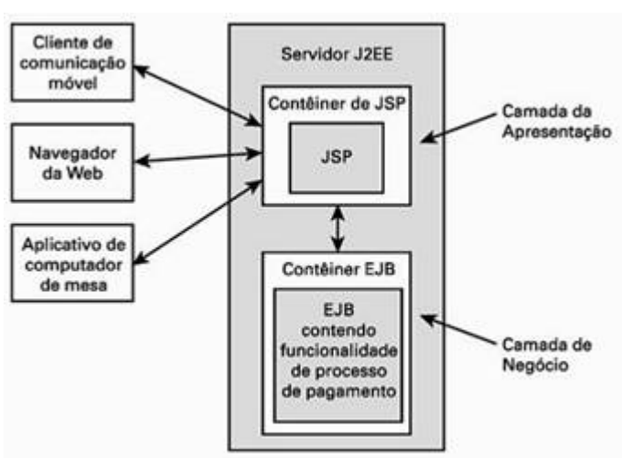


Figura 5. Simulação de acessos simultâneos.

Essa camada é considerada como lógica de apresentação, pois governa as telas que são exibidas para o usuário e de como a lógica interagem entre a regra de negócio na sua camada respectiva para conseguirem trabalhar no processo corporativo. Também se leva em conta a maneira que o usuário visualiza a página, pois tudo depende do tipo de cliente (browser) que está tentando acessar.

- **Componente Web:** Esses componentes oferecem a comunicação entre a camada de apresentação com o usuário que acontece somente por dois componentes oferecidos pelo Java EE: **JSP**, **servlets** e **Web Services**

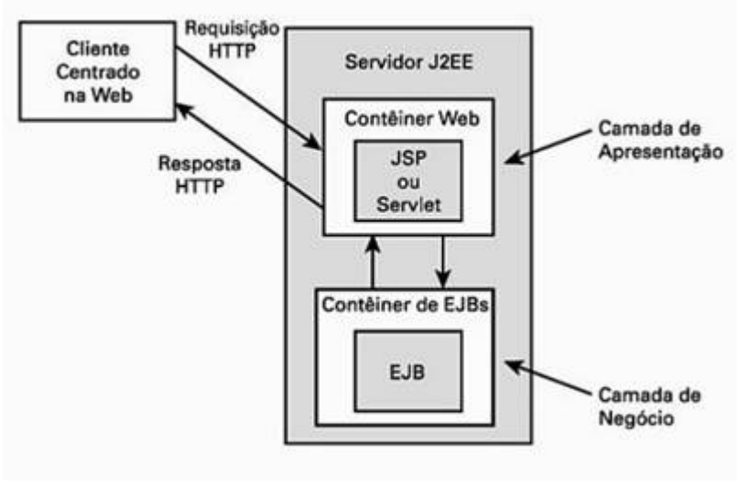


Figura 6. Exemplo de componentes centrados na Web.

- Na **Figura 6** é apresentado um exemplo de requisição do cliente para um servlets ou JSP com dados fornecidos por um EJB. No lado do servidor é analisada a requisição e logo após chama-se o EJB que é responsável por conter toda a lógica de negócio da aplicação. Nesse fluxo, quando a servlet ou JSP recebem uma resposta EJB, passam a ser responsáveis por apresentar os dados que recebem. Logo depois que o servlet ou JSP possuir uma resposta, é devolvida para o cliente que acaba completando o ciclo requisição-resposta.
- **JSP:** Considerada uma tecnologia baseada em Java que permite criar páginas dinamicamente HTML, XML que consegue resposta às requisições dos clientes. Uma das fortes características é o poder de combinar tags JSP e scriptlets que possuem códigos executáveis e marcações estáticas. Uma das vantagens é que esses códigos que ficam armazenados nas páginas JSPs são executados pelo servidor e a página resultante é enviada para o cliente, sendo assim o cliente não visualiza nada de processamento apenas o resultado que retorna na página.

para uma página JSP. Nesse caso, no momento da requisição, o contêiner de JSPs é manipulado, fazendo a conversão em um servlet. Após isso, o servlet encaminha a requisição para um componente de lógica do negócio, como outra servlet, para um JavaBean ou para um EJB que pode fazer algum processamento de acesso ao banco de dados ou tratamento de alguma regra.

Saiba mais sobre Java ;)

- **Carreira Programador Java**:

  Aprender Java não é uma tarefa simples, mas seguindo a ordem proposta nesse Guia, você evitará muitas confusões e perdas de tempo no seu aprendizado. Vem aprender java de verdade, vem!

- **Guia Completo de Java**:

  Neste Guia de Referência você encontrará todo o conteúdo que precisa para começar a programar com a linguagem Java, a sua caixa de ferramentas base para criar aplicações com Java.

- **Guia Completo de Java Enterprise Edition - Java EE**:

  Neste Guia de Referência você encontrará todo o conteúdo que precisa para conhecer o Java EE, Java Enterprise Edition, a plataforma Java voltada para o desenvolvimento de aplicações web/corporativas.

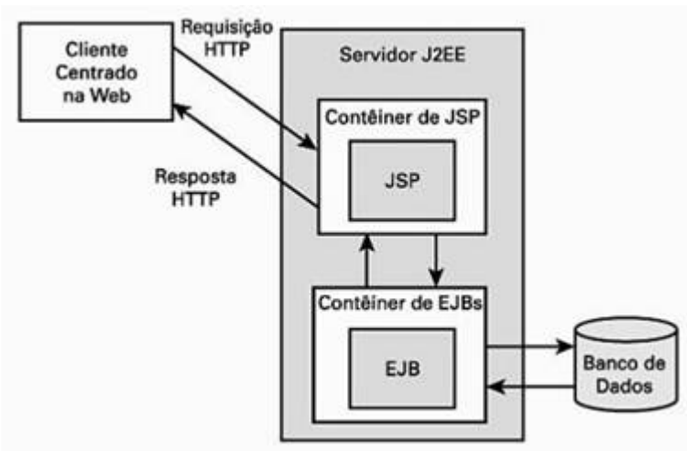


Figura 7. Fluxo de requisição nas páginas JSPs.

- **Servlets:** Os servlets são códigos Java que processam requisições (**request**) e resposta (**response**) em servidores que suportam essa tecnologia. Uma das habilidades dos servlets é a facilidade na comunicação com outros componentes Java EE. Na **Figura 8** vemos como funciona o processo de requisição.

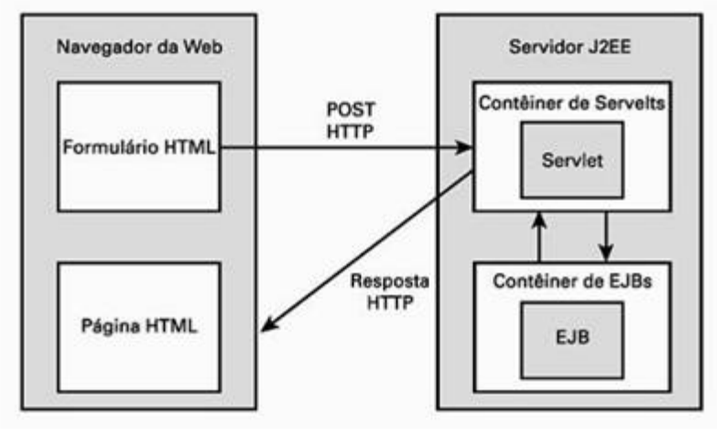


Figura 8. Processo da requisição de um cliente para uma servlet e um EJB.

- **Web Services:** São componentes de middleware baseados em XML (eXtensible Markup Language) que permitem os aplicativos acessarem através de HTTP e SOAP, por isso conseguem consumir serviços web.
  O middleware é um programa que faz a interface entre o software e outras aplicações. O motivo da existência é porque pode ser utilizado para mover ou transportar informações e dados entre programas de diferentes protocolos de comunicação, plataformas e dependências do sistema operacional em questão.

Camada Cliente

Suporta clientes web e portáteis (dispositivos móveis) que se comunicam através do protocolo HTTP, porém outros conseguem usar SOAP e outros tipos. Essa camada está formada por vários clientes como: HTML estático, HTML dinâmicos, Applet Java, independentes, empresa para empresas, clientes não java e outros clientes HTTP.

- **Cliente - HTML Estático:** Quando a internet surgiu, as primeiras páginas web eram estáticas, ou seja, páginas sem interação, pois os sites possuíam pequenos recursos e poucas funcionalidades como: navegação limitada, processamento básico de formulários de resposta para o usuário e comercialização de e-mails.
  Na **Figura 9** é mostrado um exemplo de uma aplicação web no momento em que o usuário preenche os dados de um formulário HTML e efetua uma requisição via POST através do servlet correspondente. Após os dados chegarem ao servlet, é analisado e enviado as informações para o componente JavaMail.

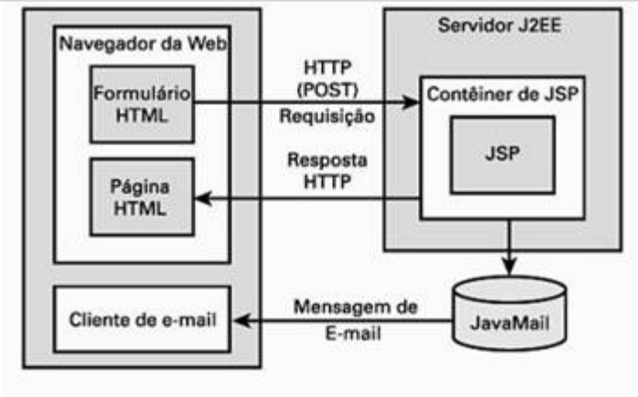


Figura 9. Processamento de um formulário com interação de um servlet.

- **Cliente - HTML Dinâmico:** Geralmente, são aplicações web que usam o acesso ao banco de dados que efetuam algum tratamento ou recuperam uma informação que precisa ser armazenada ou exposta para o usuário.
  Na **Figura 10** é mostrado um exemplo da requisição HTTP para a página inicial (índex.jsp). Pelo fato de que o mecanismo JSP interprete as tags dentro da página, isso acaba chamando o EJB que retorna uma resposta e em seguida é retornado uma página de código HTML para o cliente. Nesse exemplo, os tratamentos de regras de negócios são encapsulados no EJB.

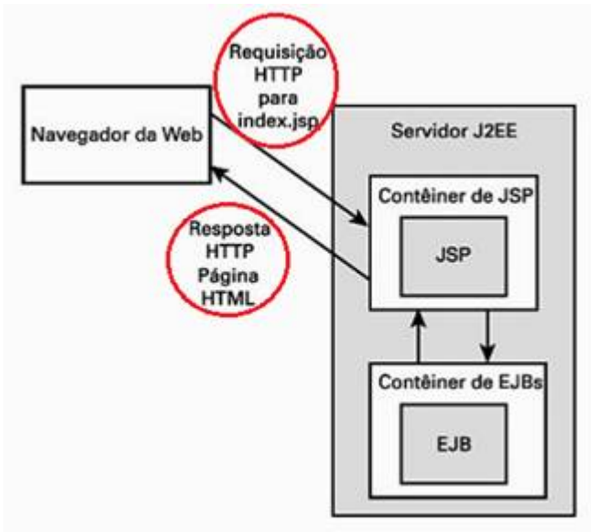


Figura 10. Cliente Web interagindo com uma JSP.

- **Cliente - Applet Java:** Os `applets` eram muito utilizados em sites de banco no início da internet, mas com o tempo caiu em desuso, pelo fato de que o cliente precisava ter uma cópia da **Java RunTime Enviroment** (JRE) instalado da mesma versão do site e além disso, possuir o plugin do applet Java no navegador para conseguir o acesso do banco.

dentro do navegador web, por isso que sempre é necessário que o navegador esteja com o plugin ativo para o funcionamento.

A invocação de um `applet` acontece no momento em que o cliente requisita uma página HTML que faz referência a um arquivo de classe (.class) no lado do servidor. Então, o servidor responde ao cliente retornando esse arquivo de classe e depois o `applet` é executado dentro da JVM fornecida pelo navegador.

A utilização hoje dos `applet` está bastante escassa, mais ainda existe em corporações que utilizam em ambientes de rede fechada.

- **Cliente Independente:** Esse tipo de cliente consegue acessar componentes através de seu contêiner. Por exemplo, na **Figura 11** o cliente EJB consegue mandar os dados para um servlet que será gerada a resposta de acordo com a requisição feita pelo cliente.

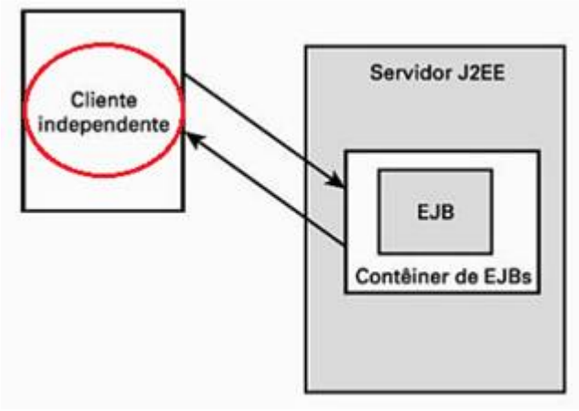
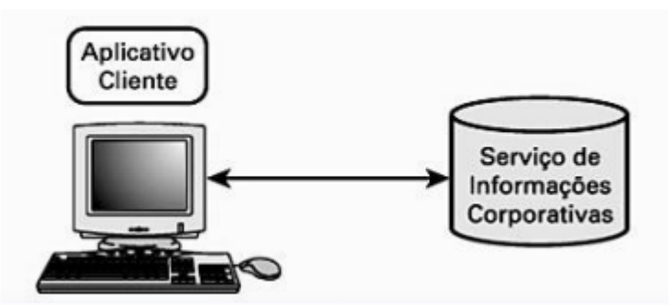


Figura 11. Cliente independente interagindo diretamente com um EJB.

Em outros casos, esse tipo de cliente ignora a camada de apresentação e de negócio justamente para acessar diretamente os recursos do sistema de informações corporativas através do JDBC.

No exemplo da **Figura 12** o cliente assume a responsabilidade pela camada da apresentação e de negócio.



corporativas.

Exemplo Empresa para Empresa

Nesse caso, muitos componentes conseguem conexão com componentes dentro da camada física do mesmo nível.

Na **Figura 13** é apresentada a arquitetura utilizada em um exemplo de aplicação web de uma loja de informática. Considerem que nessa loja, os atendentes solicitam os equipamentos, peças e acessórios que faltam através de um catálogo, através de um celular (dispositivo portátil).

Portanto, cada pedido efetuado em uma página JSP que manipula os pedidos e acessa um catálogo através do recurso JDBC. Portanto, uma página JSP baixa a quantidade em estoque dessa peça e quando não possuir mais peças no estoque, essa página se conecta com outra página JSP no atacadista de peças para fazer o pedido de mais peças.

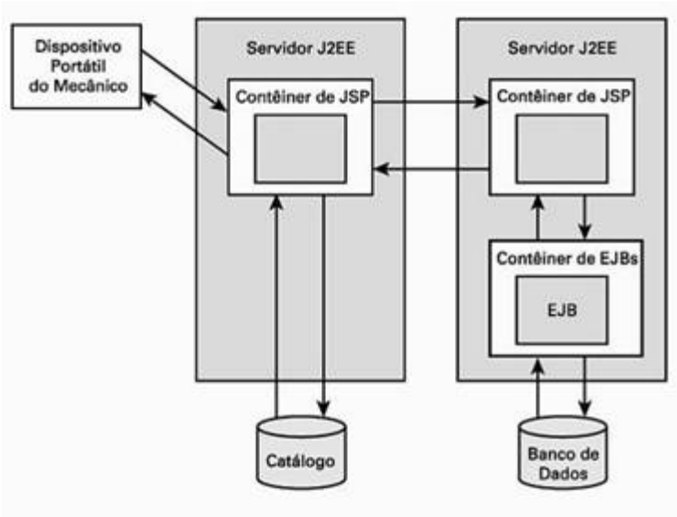


Figura13. Comunicação entre contêineres diferentes de JSP.

Essa prática seria melhor através dos recursos de troca de mensagens (JMS).

Clientes – Não-Java: Esses clientes conseguem utilizar HTTP para acessar os serviços oferecidos por uma servlet ou JSP. Isso é aplicado para casos em que um servlet ou JSP precisa consumir e produzir informações orientadas a dados baseados em XML.