**UNIVERSIDADE SENAI JOINVILLE**

**Felipe Rafael Rothbarth**

**OT09 - Introdução a back-end**

**JOINVILLE**

**2024**

### **Aplicação Cliente-Servidor**

**Definição:**A arquitetura cliente-servidor é um modelo de computação distribuída onde o processamento de dados é dividido entre clientes (dispositivos ou programas que solicitam serviços) e servidores (dispositivos ou programas que fornecem serviços). O cliente envia uma requisição ao servidor, que processa a requisição e retorna uma resposta ao cliente.

**Características:**

* **Centralização:** Os dados e a lógica de negócios geralmente residem no servidor, proporcionando maior controle e segurança.
* **Escalabilidade:** Pode ser dimensionado horizontalmente adicionando mais servidores ou clientes.
* **Manutenção:** Facilita a manutenção e atualização do software, pois as mudanças podem ser feitas no servidor sem a necessidade de distribuir o software novamente aos clientes.

### **Apache Tomcat**

**Definição:**Apache Tomcat é um servidor de aplicações Java de código aberto, desenvolvido pela Apache Software Foundation, utilizado para executar aplicações Java baseadas na tecnologia Java Servlet e JavaServer Pages (JSP).

**Características:**

* **Suporte a Servlets e JSP:** Tomcat implementa as especificações de Servlets e JSP da Oracle.
* **Leveza:** Comparado com servidores de aplicação completos, como JBoss ou WebSphere, Tomcat é mais leve e fácil de configurar.
* **Extensibilidade:** Pode ser configurado para suportar outras tecnologias e frameworks Java, como Spring.

### **HTML (HyperText Markup Language)**

**Definição:**HTML é a linguagem padrão para a criação de páginas web. É usada para estruturar o conteúdo de uma página, definindo elementos como parágrafos, títulos, links, imagens, listas, entre outros.

**Características:**

* **Estrutura de Documentos:** Define a estrutura básica de um documento web com tags como <html>, <head>, <body>, <div>, <p>, etc.
* **Hipertexto:** Permite a criação de links entre páginas web, facilitando a navegação.
* **Compatibilidade:** Funciona em todos os navegadores web, tornando-se uma tecnologia essencial para o desenvolvimento web.

### **CSS (Cascading Style Sheets)**

**Definição:**CSS é uma linguagem de estilo usada para definir a aparência e o layout de páginas web. Separa o conteúdo da apresentação, permitindo que o mesmo HTML seja estilizado de diversas maneiras.

**Características:**

* **Estilização:** Define cores, fontes, espaçamento, bordas, alinhamento, e muito mais.
* **Reutilização:** Permite a aplicação de estilos consistentes em múltiplas páginas ou elementos através de classes e IDs.
* **Responsividade:** Facilita a criação de layouts que se adaptam a diferentes tamanhos de tela, essencial para o design responsivo.

### **JavaScript**

**Definição:**JavaScript é uma linguagem de programação de alto nível, interpretada, e frequentemente utilizada para criar interatividade em páginas web. Diferente de HTML e CSS, que são usados para estruturar e estilizar o conteúdo, o JavaScript permite manipular esse conteúdo em tempo real.

**Características:**

* **Dinamicidade:** Adiciona funcionalidades interativas a páginas web, como validação de formulários, animações e manipulação de DOM (Document Object Model).
* **Execução no Cliente:** O código JavaScript é executado no navegador do cliente, proporcionando uma experiência de usuário mais rápida e dinâmica.
* **Linguagem Universal:** JavaScript é suportado por todos os navegadores modernos, sendo essencial no desenvolvimento web.

### **jQuery**

**Definição:**jQuery é uma biblioteca JavaScript rápida e concisa, criada para simplificar a manipulação de HTML, eventos, animações e Ajax. Lançada em 2006, tornou-se extremamente popular devido à sua facilidade de uso e compatibilidade cross-browser.

**Características:**

* **Sintaxe Simplificada:** Reduz a quantidade de código JavaScript necessário para realizar tarefas comuns, como selecionar elementos do DOM ou manipular eventos.
* **Compatibilidade:** Resolve problemas de compatibilidade entre navegadores, permitindo que os desenvolvedores escrevam código que funcione uniformemente em diferentes ambientes.
* **Extensibilidade:** jQuery tem um vasto ecossistema de plugins, que estendem suas funcionalidades para atender a diversas necessidades.

### **O que é uma URL?**

**URL** (Uniform Resource Locator) é o endereço utilizado para acessar recursos na internet. Ela especifica a localização de um recurso, como uma página web, uma imagem, um vídeo ou um documento, e o método para recuperá-lo. URLs são essenciais para a navegação na web, permitindo que os usuários acessem conteúdos em servidores ao redor do mundo.

### **Como é composta uma URL?**

Uma URL é composta por várias partes, cada uma desempenhando uma função específica:

1. **Esquema (Protocolo):**
   * **Exemplo:** https://
   * **Descrição:** Indica o protocolo utilizado para acessar o recurso. Comuns são http (HyperText Transfer Protocol) e https (HTTP Secure, uma versão segura do HTTP), mas outros esquemas como ftp (File Transfer Protocol) também existem.
2. **Hostname (Domínio):**
   * **Exemplo:** www.example.com
   * **Descrição:** Especifica o nome do servidor onde o recurso está hospedado. Pode incluir um subdomínio (como www) e o domínio de nível superior (como .com, .org, .gov).
3. **Porta (Opcional):**
   * **Exemplo:** :80 ou :443
   * **Descrição:** Especifica a porta de rede no servidor usada para a conexão. A porta 80 é padrão para HTTP e a 443 para HTTPS. Quando padrão, pode ser omitida na URL.
4. **Caminho:**
   * **Exemplo:** /caminho/para/recurso
   * **Descrição:** Especifica o caminho até o recurso específico no servidor. Pode apontar para arquivos ou diretórios.
5. **Query String (Consulta):**
   * **Exemplo:** ?chave=valor&outro=valor2
   * **Descrição:** Consiste em uma sequência de parâmetros em pares chave-valor, usados para enviar dados ao servidor. A string de consulta é precedida por um ponto de interrogação (?).
6. **Fragmento (Âncora):**
   * **Exemplo:** #seção1
   * **Descrição:** Usado para direcionar a um ponto específico dentro do recurso, como uma seção de uma página. O fragmento é precedido por um caractere cerquilha (#).

**Dividindo a URL:**

* **Esquema:** https://
* **Hostname:** www.example.com
* **Porta:** :443
* **Caminho:** /caminho/para/recurso
* **Query String:** ?chave=valor
* **Fragmento:** #seção1

Cada componente da URL tem sua função específica e, juntos, eles permitem que um navegador ou outro cliente acesse e interaja com recursos na internet de maneira eficiente e estruturada.

### **Servlets**

#### **Definição**

Servlets são classes Java utilizadas para criar aplicações web dinâmicas. Eles atuam como intermediários entre o cliente (navegador) e o servidor, processando requisições HTTP e gerando respostas. Servlets fazem parte da especificação Java EE (Enterprise Edition) e são executados em servidores de aplicação como Apache Tomcat.

#### **Papel na Programação Java para Web**

Os Servlets desempenham um papel crucial na programação Java para web, sendo responsáveis por:

* **Processamento de Requisições HTTP:** Capturam dados enviados pelo cliente (via GET ou POST) e executam a lógica de negócios.
* **Geração de Respostas Dinâmicas:** Produzem conteúdo dinâmico, como HTML, JSON, XML, entre outros, que é enviado de volta ao cliente.
* **Interação com Componentes do Servidor:** Podem acessar recursos do servidor, como bases de dados e APIs, para processar as informações e responder ao cliente.
* **Gerenciamento de Sessão:** Ajudam a manter o estado entre diferentes requisições de um mesmo usuário, usando sessões.

#### **Vantagens e Desvantagens**

**Vantagens:**

* **Portabilidade:** Servlets são baseados em Java, uma linguagem multiplataforma, o que garante que o código seja executado em qualquer servidor compatível com Java.
* **Desempenho:** Como são executados no servidor, os Servlets podem processar as requisições rapidamente e interagir diretamente com outros componentes do servidor.
* **Integração:** São bem integrados ao ecossistema Java, permitindo fácil uso de APIs e bibliotecas Java.
* **Segurança:** O ambiente Java fornece segurança robusta, incluindo controle de acesso e gerenciamento de threads.

**Desvantagens:**

* **Complexidade:** O desenvolvimento de aplicações web complexas com Servlets puros pode se tornar complicado e trabalhoso, especialmente para a criação de interfaces dinâmicas ricas.
* **Manutenção:** O código pode se tornar difícil de manter e escalar à medida que a aplicação cresce, devido à mistura de lógica de negócios e apresentação.
* **Menor Produtividade:** Comparado a frameworks modernos como Spring e Hibernate, que oferecem abstrações mais altas, o desenvolvimento com Servlets pode ser mais lento e menos produtivo.

### **Servidores de Aplicação Web**

#### **Definição**

Servidores de aplicação web são plataformas de software que fornecem o ambiente necessário para executar aplicações web. Eles gerenciam a lógica de negócios, o processamento de requisições HTTP, a execução de Servlets, JavaServer Pages (JSP), e outros componentes da aplicação. Esses servidores facilitam a comunicação entre o cliente (navegador) e o servidor de backend, processando e respondendo às solicitações dos usuários.

#### **Papel na Programação Java para Web**

Na programação Java para web, os servidores de aplicação desempenham um papel fundamental ao fornecer um ambiente de execução para Servlets, JSPs, e outros componentes baseados em Java. Eles são responsáveis por:

* **Gerenciamento de Ciclo de Vida:** Iniciar, parar, e gerenciar o ciclo de vida dos Servlets e outras aplicações.
* **Segurança:** Implementar e gerenciar a autenticação, autorização e outras políticas de segurança.
* **Manejo de Sessões:** Gerenciar sessões de usuário, permitindo a persistência de dados entre diferentes requisições.
* **Escalabilidade:** Permitir que a aplicação se expanda para lidar com mais requisições simultâneas, distribuindo a carga de trabalho.
* **Conectividade com Banco de Dados:** Facilitar a comunicação com bancos de dados e outros serviços backend.

#### **Exemplo de Como Funcionam as Aplicações Web**

Vamos entender o fluxo de funcionamento de uma aplicação web baseada em Java:

1. **Requisição do Cliente:**
   * Um usuário interage com uma interface web (página HTML) e realiza uma ação, como clicar em um botão "Enviar".
   * O navegador envia uma requisição HTTP ao servidor de aplicação, normalmente utilizando métodos como GET ou POST.
2. **Processamento pelo Servidor:**
   * O servidor de aplicação, como Apache Tomcat, recebe a requisição e encaminha para o Servlet apropriado, com base na URL.
   * O Servlet processa a requisição, possivelmente interagindo com o banco de dados ou outros recursos, para executar a lógica de negócios.
3. **Geração da Resposta:**
   * O Servlet gera uma resposta, que pode ser uma página HTML, dados JSON, ou outro tipo de conteúdo dinâmico.
   * Essa resposta é enviada de volta ao navegador do cliente.
4. **Resposta ao Cliente:**
   * O navegador do cliente recebe a resposta do servidor e a exibe ao usuário, concluindo o ciclo de comunicação.

**Fluxo de Funcionamento:**

Cliente (Navegador) -> Requisição HTTP -> Servidor de Aplicação (Tomcat) -> Processamento (Servlet) -> Resposta HTTP -> Cliente (Navegador)

Esse processo permite que as aplicações web sejam dinâmicas, interativas e responsivas às ações dos usuários.

#### **Qual servidor web utilizaremos nesse projeto?**

Para este projeto, utilizaremos o **Apache Tomcat** como o servidor de aplicação web. O Tomcat é um servidor robusto e amplamente utilizado, especialmente em projetos baseados em Java. Ele oferece suporte completo para Servlets, JSP, e outros componentes essenciais para a criação de aplicações web em Java. Além disso, o Tomcat é conhecido por sua facilidade de uso e configuração, tornando-o uma excelente escolha para projetos de pequeno a médio porte.

**REFERÊNCIAS BIBLIOGRÁFICAS**

KROENKE, David M.; AUER, David J. **Database Concepts**. 9. ed. Upper Saddle River: Pearson, 2018.

APACHE SOFTWARE FOUNDATION. **Apache Tomcat**. Disponível em:<https://tomcat.apache.org/>. Acesso em: 12 ago. 2024.

W3C. **HTML & CSS**. Disponível em:<https://www.w3.org/standards/webdesign/htmlcss>. Acesso em: 12 ago. 2024.

MOZILLA DEVELOPER NETWORK (MDN). **JavaScript**. Disponível em:<https://developer.mozilla.org/pt-BR/docs/Web/JavaScript>. Acesso em: 12 ago. 2024.

JQUERY FOUNDATION. **jQuery**. Disponível em:<https://jquery.com/>. Acesso em: 12 ago. 2024.

BERNERS-LEE, Tim; MASINTER, L.; MCCAHILL, M. **Uniform Resource Locators (URL)**. RFC 1738, dez. 1994. Disponível em: https://www.rfc-editor.org/rfc/rfc1738. Acesso em: 12 ago. 2024.

MOZILLA DEVELOPER NETWORK (MDN). **URL - Uniform Resource Locator**. Disponível em:<https://developer.mozilla.org/en-US/docs/Web/API/URL>. Acesso em: 12 ago. 2024.

W3C. **URIs, URLs, and URNs: Clarifications and Recommendations 1.0**. Disponível em:<https://www.w3.org/TR/uri-clarification/>. Acesso em: 12 ago. 2024.

HALL, Marty; BROWN, Larry. **Core Servlets and JavaServer Pages (JSP): Volume 1: Core Technologies**. 2. ed. Upper Saddle River: Prentice Hall, 2003.

ORACLE. **Servlet Technology**. Disponível em:<https://docs.oracle.com/javaee/7/tutorial/servlets.htm>. Acesso em: 12 ago. 2024.

RAO, Subrahmanyam Allamaraju; BUDAK, Cedric. **Java Servlet Programming**. 2. ed. Beijing; Sebastopol: O'Reilly Media, 2001.

APACHE TOMCAT. **Apache Tomcat Documentation**. Disponível em:<https://tomcat.apache.org/tomcat-9.0-doc/servletapi/index.html>. Acesso em: 12 ago. 2024.