Linguagem de Programação II

Prof.ª Ma. Jessica Oliveira

FBr Faculdade Brasília

Aula 16 – 18/11/2024

Revisão para a AV2.

Faculdade Brasília

MySQL

Sistema de gerenciamento de banco de dados **relacional** (organiza os dados em tabelas que podem ser relacionadas entre si através de chaves primárias e estrangeiras) de **código aberto** (os desenvolvedores podem inspecionar, modificar e distribuir o *software* conforme necessário), que utiliza a **linguagem SQL** (*Structured Query Language*) como padrão para realizar consultas e manipulação de dados e é **multiusuário e *multithread*** (permite que múltiplos usuários realizem consultas simultâneas e suporta o uso de várias *threads*).

MySQL

No quesito **desempenho e escalabilidade**, é conhecido por ser leve e rápido, o que o torna uma escolha popular para aplicações *web* e sistemas de médio e grande porte, além de oferecer suporte a tabelas de grande escala e transações complexas.

Possui vários níveis de **segurança**, como controle de acesso baseado em privilégios, autenticação segura, criptografia SSL, entre outros, o que o torna uma escolha confiável para proteger os dados em aplicações sensíveis.

Oferece o **PHPMyAdmin**, uma interface gráfica baseada na *web* que facilita a administração de bancos de dados, permitindo a criação, modificação e manipulação sem a necessidade de escrever código.

Sessões

São uma **maneira de armazenar informações do usuário** enquanto ele navega em diferentes páginas de um site, e são úteis para aplicações que exigem que o usuário esteja autenticado, pois permitem manter os dados ativos enquanto o usuário está no site, sem que os mesmos sejam armazenados no navegador.

Diferente dos *cookies*, que armazenam dados no navegador do cliente, **as sessões utilizam um identificador único** (`session_id`) para **localizar e manipular dados diretamente no servidor**.

Upload de arquivos

Permite que os usuários interajam com o sistema enviando documentos, imagens, vídeos, ou outros tipos de arquivos, tornando o sistema dinâmico e personalizado. Em sistemas corporativos, por exemplo, colaboradores podem enviar relatórios, contratos ou comprovantes. Além disso, facilita a coleta de informações de maneira automatizada, eliminando a necessidade de envio manual por e-mail ou outros meios, agilizando fluxos de trabalho. Inclusive, garante que os arquivos enviados pelos usuários fiquem armazenados em um local seguro e organizado, seja no servidor ou integrado com um serviço de armazenamento em nuvem.

Upload de arquivos

Importante atentar-se a alguns pontos nesse tema:

- Na **configuração**, é preciso ajustar parâmetros no `php.ini`, como `upload_max_filesize` e `post_max_size`, para definir limites de tamanho.
- Na **segurança**, é essencial validar o tipo e o tamanho do arquivo, renomeá-lo com funções como `uniqid()` para evitar conflitos, e armazená-lo fora de diretórios públicos sempre que possível.
- Por fim, no **armazenamento**, os arquivos podem ser salvos em diretórios locais, serviços em nuvem para escalabilidade, ou diretamente no banco de dados em formato BLOB, dependendo das necessidades de acesso e centralização de dados.

Inclusão de arquivos

Em aplicações PHP OO, o código é dividido em classes, funções e componentes separados que são armazenados em arquivos diferentes. Esses arquivos podem ser incluídos quando necessários usando os comandos `include` ou `require`.

Essa abordagem permite a **modularização**, possibilitando que cada componente do sistema seja desenvolvido e testado de forma isolada, o que simplifica a identificação e a correção de problemas. Além disso, inclui a vantagem da **reutilização**, pois funções ou classes podem ser compartilhadas entre diferentes partes do código, reduzindo redundâncias.

Inclusão de arquivos

A **organização** do projeto também é aprimorada, garantindo um código mais limpo e legível, especialmente em aplicações de maior escala. Por fim, a **manutenção** se torna mais prática e centralizada, uma vez que atualizações realizadas em um único arquivo automaticamente impactam todas as áreas que o utilizam, promovendo consistência e agilidade no desenvolvimento.

Comandos para inclusão de arquivos

- **`include`:** inclui o arquivo especificado. Se o arquivo não for encontrado, ele gera um aviso e continua a execução.
- **`require`:** similar ao include, mas se o arquivo não for encontrado, ele gera um erro fatal e interrompe a execução.
- **`include_once`:** inclui o arquivo apenas uma vez. Mesmo que o comando seja repetido, o arquivo não será incluído novamente.
- **`require_once`:** combina as características do require e do include_once, garantindo que o arquivo será incluído uma única vez e interromperá a execução se não for encontrado.

Arquivos texto

Arquivos texto armazenam dados legíveis organizados em linhas, sendo usados em PHP para armazenamento de dados temporários ou permanentes sem banco de dados. Eles são importantes por oferecer:

- **Persistência de dados:** permitem salvar informações para uso posterior, como logs, configurações e caches.
- **Interoperabilidade:** formatos como `.txt` e `.csv` são amplamente suportados e facilitam a troca de dados entre sistemas.
- **Geração de relatórios:** possibilitam a exportação de dados estruturados para formatos como CSV ou PDF, permitindo análises e visualizações.

Arquivos texto

- Para manipular arquivos em PHP, usamos a função `fopen()`, que permite abrir arquivos em diferentes modos.

MODO	DESCRIÇÃO
r	Abre para leitura. O ponteiro é posicionado no início. O arquivo deve existir.
w	Abre para escrita, sobrescrevendo o conteúdo existente. Se não existir, o arquivo é criado.
a	Abre para escrita, mantendo o conteúdo existente e adicionando ao final. Cria o arquivo caso não exista.
x	Cria um arquivo para escrita. Se o arquivo já existir, a operação falha.

Arquivos texto

- Para ler um arquivo em PHP OO, é comum utilizar uma estrutura de laço que percorre cada linha do arquivo.
- Para escrever dados em um arquivo, o PHP usa `fwrite()`.
- Erros ao abrir, ler ou escrever arquivos são comuns. Utilizar `try-catch` para capturar exceções ajuda a evitar interrupções inesperadas no sistema.

Geração de relatórios

- CSV (*Comma-Separated Values*) é um formato comum para exportação de dados, comumente utilizado em planilhas e bancos de dados.
- PDF é o formato ideal para relatórios prontos para impressão. Usando bibliotecas como FPDF, é possível criar relatórios customizados e formatados.

Smarty

Ferramenta poderosa para desenvolvedores PHP que permite separar completamente a lógica da aplicação (PHP) da interface do usuário (HTML), ou seja, atua como uma ponte entre o código PHP e a estrutura visual do site.

Seu funcionamento baseia-se em três etapas principais: inicialmente, são criados arquivos de ***template***, geralmente com extensão `.tpl`, contendo a estrutura HTML da página com *placeholders* para dados dinâmicos. Em seguida, no código PHP, os **valores são atribuídos** a essas variáveis e o *template* é passado para o Smarty. Por fim, o *Smarty* **processa o *template***, substitui os *placeholders* pelos valores fornecidos e gera o HTML final, que é enviado ao navegador do usuário.

Smarty

Apresenta diversas **vantagens**, como a separação entre lógica PHP e design, permitindo que desenvolvedores e designers trabalhem de forma independente. Ele facilita alterações na interface diretamente nos *templates*, promove a reutilização de código, oferece mecanismos de segurança, melhora o desempenho ao compilar *templates* em PHP e conta com uma grande comunidade com diversos recursos disponíveis.

No entanto, possui algumas **desvantagens**, como a necessidade de aprender uma nova sintaxe, o potencial overhead em aplicações muito complexas, maior complexidade na configuração em projetos pequenos e sua adequação limitada dependendo do caso de uso.

Web Service

É um método de comunicação entre diferentes aplicações, geralmente através da internet. Imagine-os como serviços que uma aplicação pode "contratar" de outra. Por exemplo, uma aplicação de *e-commerce* pode usar um *web service* de um serviço de pagamento para processar transações.

Como principais características, podemos citar que eles podem ser acessados por qualquer aplicação, comumente utilizam protocolos como SOAP e REST, e geralmente utilizam XML ou JSON para trocar informações.

API

Uma Interface de Programação de Aplicativos é um conjunto de regras e especificações que define como diferentes *softwares* podem interagir entre si. É como um contrato que define quais são as funcionalidades que um *software* oferece e como outros *softwares* podem acessá-las.

De forma resumida, uma API **é a porta de entrada para os recursos de um sistema**.

RESTful APIs

São **um tipo específico de API que seguem os princípios da arquitetura REST**. Essa arquitetura enfatiza a simplicidade, a escalabilidade e a utilização de protocolos HTTP padrão.

Possuem características essenciais que as tornam eficientes e flexíveis. Cada informação é tratada como um recurso, identificado por um URI (*Uniform Resource Identifier*), e as operações nesses recursos são realizadas utilizando os **métodos HTTP**, como **GET, POST, PUT e DELETE**, para obter, criar, atualizar ou deletar dados. Elas são *stateless*, ou seja, cada requisição contém todas as informações necessárias para ser processada, sem que o servidor mantenha estado entre as requisições. Além disso, os dados retornados são representados em formatos como JSON ou XML, facilitando a interoperabilidade.

Por que usar APIs?

O uso delas traz diversos benefícios, como a reutilização de código já existente, evitando a necessidade de recriar funcionalidades, e a integração eficiente entre sistemas, facilitando a troca de dados.

Além disso, aceleram o desenvolvimento ao permitir que os desenvolvedores foquem nas funcionalidades específicas de suas aplicações, promovendo a inovação ao possibilitar a criação de novos serviços e produtos.

Por fim, as APIs são projetadas para serem escaláveis, atendendo a um grande número de usuários e requisições sem comprometer o desempenho.

Não confundam!

- **Todos os *web services* são APIs:** pois eles expõem uma interface para que outros sistemas possam se comunicar com eles.
- **Nem todas as APIs são *web services*:** uma API pode ser local, por exemplo, para comunicação entre diferentes componentes de um mesmo aplicativo.
- **RESTful APIs são um tipo específico de *web service*:** elas utilizam o protocolo HTTP e os princípios REST para expor funcionalidades e dados.

Não confundam!

- *Web services* são a categoria mais ampla, representando qualquer mecanismo de comunicação entre sistemas.
- APIs são interfaces que permitem o acesso a recursos de um sistema.
- RESTful APIs são um tipo de API que utiliza os princípios REST e o protocolo HTTP para oferecer uma forma simples e eficiente de comunicação entre sistemas.

Métodos HTTP

São essenciais para a **comunicação entre clientes e servidores** na *web*, definindo as ações realizadas sobre os recursos disponíveis. O **método GET** serve para obter informações do servidor, sendo usado em operações de leitura, como acessar listas ou detalhes específicos de itens. Ele **não altera o estado do servidor e os parâmetros de consulta são incluídos na URL**.

O **POST**, por outro lado, envia dados ao servidor com o objetivo de criar ou processar novos recursos, como ao cadastrar um item. Os dados **são enviados no corpo da requisição, e cada execução pode gerar resultados diferentes, como a criação de registros duplicados**.

Métodos HTTP

O **PUT** é empregado para atualizar ou substituir informações existentes. Diferentemente do POST, o PUT **garante que a mesma ação repetida não cause mudanças adicionais, pois atualiza os dados fornecidos de forma consistente**.

Já o **DELETE** é utilizado para excluir recursos, removendo itens identificados no servidor. Assim como o PUT, repetir o DELETE **não traz efeitos extras após a remoção**.

Esses métodos são a base das operações CRUD (*Create, Read, Update e Delete*), indispensáveis para manipulação de dados em sistemas web.

Dúvidas?

jessica.oliveira@fbr.edu.br