Métodos de Envio de Formulários HTML - GET vs. POST

1. Introdução aos Métodos de Envio de Formulários

Ao construir seus formulários em HTML, um dos passos mais importantes é definir como os dados inseridos pelo usuário serão transmitidos do seu navegador para o servidor. Esse processo é controlado pelo **método de envio** do formulário. Entender a diferença entre os métodos disponíveis é fundamental para que você trate os dados de forma adequada, seja para uma simples busca ou para o envio de informações sensíveis.

Por padrão, os formulários HTML utilizam um método específico se nenhum for declarado. No entanto, o desenvolvimento web nos oferece duas opções principais: **`GET`** e **`POST`**. Cada um possui características distintas que os tornam adequados para diferentes cenários. A escolha correta não é apenas uma questão técnica, mas uma decisão que impacta a usabilidade, a segurança e a funcionalidade da sua aplicação. Vamos começar analisando o método padrão, o **`GET`**.

2. O Método `GET`: O Padrão Visível

O método **`GET`** é a forma padrão de envio de dados de um formulário HTML. Sua principal característica, e a mais notável, é que todos os dados coletados nos campos são anexados diretamente à **URL** de destino, tornando-os completamente visíveis na barra de endereço do navegador.

Na prática, quando um usuário preenche um formulário e o envia, o navegador pega os dados de cada campo (por exemplo, `nome=José` e `sobrenome=Soares`) e os adiciona ao final do endereço do arquivo de destino, separados por um ponto de interrogação (`?`). Um exemplo clássico que você vê todos os dias é a busca no Google. Ao pesquisar por `curso em vídeo`, a **URL** resultante mostra claramente o termo pesquisado, o que permite que essa busca específica seja salva nos favoritos ou compartilhada com outras pessoas.

O poder do **`GET`** fica ainda mais claro quando percebemos que essa **URL** pode ser editada. Você pode copiar o endereço completo da busca, apagar todos os parâmetros extras e enviar a um colega apenas o link essencial da pesquisa (como `https://www.google.com/search?q=vírus+e+bactérias+diferença`), que funcionará perfeitamente. Isso é impossível de se fazer com o **`POST`**. Para definir explicitamente o uso do método **`GET`** em seu formulário, você utiliza o atributo `method`. Se este atributo for omitido, o navegador assumirá **`GET`** como padrão.

```
<form action="cadastro.php" method="get">
  <!-- Campos do formulário aqui -->
</form>
```

Embora a visibilidade dos dados na **URL** seja útil para buscas e compartilhamento, ela se torna um problema em situações que exigem mais discrição. É por isso que existe uma alternativa para enviar dados de forma oculta.

3. O Método `POST`: Enviando Dados de Forma Oculta

O método **`POST`** é a principal alternativa ao **`GET`**. Sua função é enviar os dados do formulário de uma maneira diferente: em vez de anexá-los à **URL**, ele os envia de forma "invisível", como se estivessem dentro de um envelope (o corpo da requisição HTTP), em vez de escritos no lado de fora (a **URL**). Como resultado, os dados não ficam visíveis na barra de endereço do navegador para o usuário comum.

Ao utilizar o **`POST`**, se um usuário preencher um formulário com os dados `Mariana Gonçalves` e o enviar para um arquivo chamado `cadastro.php`, a **URL** no navegador exibirá apenas `cadastro.php`, sem nenhum dado adicional. Essa característica o torna a escolha ideal para formulários que lidam com informações que não devem ser expostas publicamente. A sintaxe para implementar o método **`POST`** é muito simples, bastando declarar o atributo `method` na tag `<form>`:

```
<form action="cadastro.php" method="post">
  <!-- Campos do formulário aqui -->
</form>
```

Apesar de ocultar os dados da **URL**, é crucial que você desmistifique uma concepção errada muito comum: o método **`POST`** não é, por si só, uma ferramenta de segurança.

4. Desmistificando a Segurança: `POST` não é Criptografia

É um erro comum acreditar que, por ocultar os dados da barra de endereço, o método **`POST`** é inerentemente seguro. Entender os limites de segurança dos métodos de formulário é um passo estratégico para qualquer desenvolvedor. O **`POST`** apenas muda por onde os dados trafegam, mas não os protege contra interceptação. Qualquer pessoa com conhecimento técnico mínimo pode visualizar os dados enviados via **`POST`**. Para ver isso em ação, siga estes passos no seu navegador:

1. Abra as Ferramentas de Desenvolvedor (geralmente com a tecla F12).
2. Clique na aba "Rede" (`Network`).
3. Preencha e envie o formulário que utiliza o método **`POST`**.
4. Na lista de requisições que aparecer, clique no arquivo de destino (ex: `cadastro.php`).
5. Nos detalhes da requisição, procure pelos dados enviados (como `Glauber Pereira`), que estarão visíveis em texto puro.

A conclusão é direta: **`POST`** apenas oculta os dados, não os criptografa. Para proteger informações verdadeiramente sensíveis, como senhas e dados de cartão de crédito, é indispensável o uso do protocolo **`HTTPS`**. O **`HTTPS`** cria uma camada de criptografia que protege os dados durante toda a transmissão entre o navegador e o servidor. Agora que desmistificamos essa ideia de segurança, a pergunta natural é: afinal, quando devo escolher cada método? Vamos criar um guia prático para resolver essa dúvida de uma vez por todas.

5. Guia de Decisão: Quando Usar `GET` vs. `POST`

A escolha entre **`GET`** e **`POST`** não se baseia em qual método é "melhor", mas sim em qual é o mais adequado para a tarefa que você precisa realizar. Antigamente, uma grande discussão era sobre a velocidade, com o **`GET`** sendo considerado mais rápido. Hoje, com a evolução da tecnologia, essa diferença é imperceptível. Portanto, não baseie sua decisão nisso. Guie-se pela finalidade e pela natureza dos dados.

Utilize o método `GET` quando:

- Os dados **não forem sensíveis** (ex: termos de busca, filtros de pesquisa, nomes).
- Você desejar que a **URL seja compartilhável** ou possa ser adicionada aos favoritos (ex: resultados de uma busca ou uma página de produto com filtros).

- A quantidade total de dados enviados for pequena, com um limite de até **3.000 bytes** (aproximadamente 3.000 letras).

Utilize o método `POST` quando:

- Os dados forem **sensíveis** (senhas, cartões de crédito). **Neste caso, o uso de `HTTPS` não é opcional, é obrigatório para garantir a segurança real dos dados.**
- Você **não** quiser que os dados fiquem visíveis no histórico do navegador ou na **URL**.
- A quantidade de dados a ser enviada **ultrapassar 3.000 bytes**.
- O formulário permitir o **envio de arquivos** (fotos, documentos, etc.).

Tomar uma decisão informada sobre qual método utilizar é um sinal de maturidade no desenvolvimento e garante que sua aplicação se comporte da maneira esperada e segura.

6. Conclusão

A lição mais importante que você deve levar desta apostila é que a sua escolha entre **`GET`** e **`POST`** depende de uma análise cuidadosa do que você precisa. Eles não são concorrentes, mas ferramentas com propósitos distintos. **`GET`** oferece transparência e URLs compartilháveis, ideal para interações não sensíveis. **`POST`** proporciona discrição e maior capacidade de dados, sendo a escolha certa para informações sensíveis e uploads de arquivos, sempre em conjunto com **`HTTPS`**.

O próximo passo é seu: aplique este conhecimento, crie seus próprios formulários e teste os dois métodos. A prática é o que solidificará seu aprendizado e o transformará em um desenvolvedor mais completo e seguro.