Dynamické stránky s přístupem do databáze (formuláře a jejich zpracování, zabezpečení uživatelského vstupu, zápis do databáze, výpis dat z databáze)

Dynamické stránky s přístupem do databáze

Kromě prezentování nějakých informací umožňují dynamické stránky pracovat s nějakými daty či je zpracovávat. Je nutné využít jazyk, který toto umožňuje, dnes nejčastěji PHP. Data jsou uložena v databázi, prostřednictvím jejíž vrstvy je k nim pak také přistupováno.

Formuláře a jejich zpracování

Formuláře slouží k odesílání dat na server, kde jsou následně data zpracována. Data z formuláře je možné odeslat dvěma metodami: GET a POST, které se od sebe poměrně zásadně liší.

GET

Metoda GET odesílá veškerá data jako součást URL adresy – v tzv. parametrech. URL pak obvykle končí například následující částí: ?name=Tom&id=805&pohlavi=m. V té jsou obsaženy veškeré údaje, které byly formulářem odeslány. Neumožňuje odesílat jiná data než textová. Není určena pro odesílání citlivých dat (např. hesel), protože jsou data v adrese viditelná. Některé znaky je třeba převádět na přenositelné znaly (např. mezeru na %20).

K datům lze v PHP přistupovat pomocí superglobální proměnné $_GET[$key] kde $key odpovídá konkrétnímu vstupu z odeslaného formuláře, např. "name".

Používá se například při vyhledávání, kdy je následně dotaz uveden v parametru URL adresy pro snadné sdílení výsledků vyhledávání atp.

POST

Metoda POST odesílá veškerá data v hlavičkách HTTP dotazu. Data tak nejsou navenek viditelná, je tedy vhodná pro přenos citlivých dat jako jsou hesla. Oproti metodě GET umožňuje odeslat kromě textových dat také soubory a délka není omezena takovým způsobem jako délka GET. Pro omezení délky POST dotazu slouží na serveru v PHP funkce post_max_size celkově, případně upload_max_filesize pro nahrávané soubory.

Zabezpečení uživatelského vstupu

Je třeba zabezpečit, aby server od uživatele dostal taková data, která očekává. Byť můžeme pro ověření tvaru dat použít JavaScript na straně klienta, tato ochrana lze obejít a někdy nemusí být funkční. Nejspolehlivější je tak zabezpečení na straně serveru, kde po odeslání formuláře každý odeslaný vstup (prvek) zkontrolujeme – zda odpovídá potřebám – např. zda je e-mail či datum zadáno ve správném tvaru.

Kromě toho je třeba zabezpečit aplikaci proti různým útokům, jako je např. SQL injection, což je útok, při němž je prostřednictvím webového formuláře odeslán SQL dotaz, který je, namísto, aby

s ním server pracoval jako s obyčejným textem, zpracován, a to právě z důvodu špatného nebo žádného zabezpečení takových situací.

K ověřování správného tvaru dat se používají filtry, které přineslo PHP 7, či vlastní funkce, které obvykle využívají porovnání s regulárním výrazem. SQL injection útokům lze zabránit přidáním funkce `bind_param('ssi', $a, $b, $c)` před vložením hodnoty do takových proměnných. Při odeslání SQL dotazu se ověří, zda proměnné obsahují požadovaný datový typ. Funkce zároveň zabrání SQL injection. První parametr 'ssi' znamená, že první a druhá proměnná budou datového typu string a třetí typu integer.

Různé knihovny, které komunikaci s DB usnadňují (např. dibi), většinou mívají toto zabezpečení implementováno samy v sobě.

Zápis do databáze

Zápis do databáze se provádí pomocí nějakého databázového dotazu – odesláním příslušného požadavku na databázi – tedy odesláním požadavku pro uložení dat společně s takovými daty.

Například u databáze MySQL se používá jazyk SQL a příkaz pro uložení nějakých dat do tabulky user může vypadat například takto:

```
INSERT INTO user (username, password, email, admin) VALUES ('tomas', 'hashovaneheslo',
'tom@tom.tom', 0);
```

Výpis dat z databáze

Výpis dat z databáze se provádí taktéž odesláním příslušného požadavku na databázi. Ten může vypadat například následovně (provede načtení všech uživatelů z tabulky user):

```
SELECT * FROM user;
```

Informace o uživatelích, které se načtou, je následně možné procházet cyklem a postupně tak utvořit například tabulku s takovými informacemi. Cyklem proto, protože byly vybrány všechny záznamy a v takovém případě jsou uloženy v poli.

Následujícím příkazem můžeme z databáze zase získat například pouze e-mail nějakého jednoho konkrétního uživatele:

```
SELECT email FROM user WHERE username = 'tomas';
```