

Návrhové vzory (Design Patterns)

Návrhové vzory jsou standardizovaným řešením pro typické situace v [objektově orientovaném návrhu](#). Pomáhají vývojářům psát kód, který je srozumitelný, snadno testovatelný a především udržitelný.

Tento koncept zpopularizovala v roce 1994 kniha skupiny autorů známé jako „**Gang of Four**“ (GoF).

1. Klasifikace vzorů

Návrhové vzory se tradičně dělí do tří základních kategorií podle jejich účelu:

A) Vzory tvořivé (Creational)

Řeší mechanismy vytváření objektů. Cílem je oddělit proces tvorby objektu od systému, který ho používá.

- **Singleton (Jedináček):** Zajišťuje, aby třída měla pouze jednu instanci a poskytuje k ní globální přístup.
- **Factory Method (Tovární metoda):** Definuje rozhraní pro vytváření objektů, ale o konkrétní třídě rozhoduje až podtřída.
- **Builder:** Umožňuje postupnou konstrukci složitých objektů.

B) Vzory strukturální (Structural)

Zabývají se tím, jak se třídy a objekty skládají do větších struktur.

- **Adapter:** Umožňuje spolupráci tříd, které mají nekompatibilní rozhraní.
- **Facade (Fasáda):** Poskytuje zjednodušené rozhraní k celému podsystému tříd.
- **Decorator:** Umožňuje dynamicky přidávat objektům nové vlastnosti bez změny jejich struktury.

C) Vzory chování (Behavioral)

Zaměřují se na komunikaci mezi objekty a rozdělení zodpovědnosti.

- **Observer (Pozorovatel):** Definuje závislost 1:N, kdy při změně stavu jednoho objektu jsou automaticky informováni všichni závislí.
- **Strategy:** Umožňuje definovat rodinu algibritmů a zaměřovat je za běhu programu.
- **Command:** Zapouzdřuje požadavek jako objekt, což umožňuje parametry požadavků ukládat nebo vracet zpět.

2. Detailní pohled: Singleton (Jedináček)

Singleton je jeden z nejznámějších (a někdy nadužívaných) vzorů. Používá se například pro správce konfigurace nebo připojení k databázi.

Klíčové vlastnosti:

- Soukromý (private) konstruktor.
- Statická metoda, která vrací instanci.
- Instance je vytvořena pouze tehdy, když je poprvé potřeba (Lazy initialization).

3. Detailní pohled: Observer (Pozorovatel)

Tento vzor je základem pro asynchronní systémy a uživatelská rozhraní (GUI).

- **Subject (Předmět):** Drží seznam pozorovatelů a informuje je o změnách.
- **Observer (Pozorovatel):** Čeká na zprávu od předmětu a reaguje na ni (např. překreslení grafu po změně dat).

4. Výhody a nevýhody

Výhody	Nevýhody
Srozumitelnost: Vývojáři používají společný „jazyk“ (např. „Tady použijeme Fasádu“).	Překombinovanost: Použití složitého vzoru na jednoduchý problém (Overengineering).
Osvědčená praxe: Řešení eliminují chyby, které už někdo udělal před vámi.	Učební křivka: Správné pochopení a implementace vyžaduje zkušenost.
Snadná údržba: Vzory podporují principy SOLID .	

5. Vztah k dalším tématům

Návrhové vzory nejsou totéž co **architektonické vzory** (např. MVC – Model-View-Controller). Architektonické vzory mají globální dopad na celou aplikaci, zatímco návrhové vzory řeší konkrétní problémy v rámci struktury tříd.

Související články:

- [Základy objektově orientovaného programování](#)
- [Principy SOLID](#)

- [Architektura MVC](#)

Tagy: *programming software-engineering design-patterns oop architecture*

From:

<https://www.serviceit.cz/> - **IT ENCYKLOPEDIE**

Permanent link:

https://www.serviceit.cz/doku.php?id=it:sw:design_patterns

Last update: **2026/01/02 18:54**

