Uživatelské rozhraní - Události, Kontrola vstupů, Layout, Ovládací prvky

*Předmluva: Velký část ohledně uživatelského rozhraní se dá jednoduše vymyslet. Stačí se jenom zamyslet nad tím, v jakých uživatelských rozhraních jste kdy pracovali a následně se z toho dají poměrně jednoduše vyvodit nějaké rozdíly a závěry. Tudíž i kdyby jste se na tuhle otázku úplně vysrali, tak to na 4 dáte.*

# Uživatelské rozhraní

Uživatelským rozhraním máme na mysli interakci uživatele s programem.

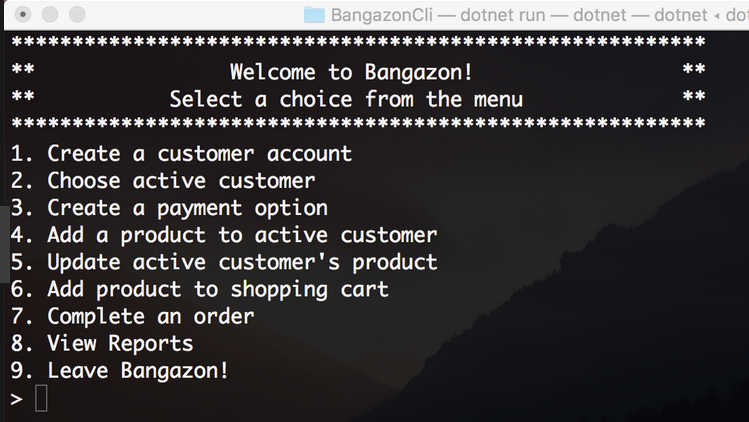
Existuje několik druhů uživatelských rozhraní a několik různých technologií.

Uživatelské rozhraní by mělo být uživatelsky přívětivé (přehledné, intuitivní, funkční)

## CLI (Command Line Interface)

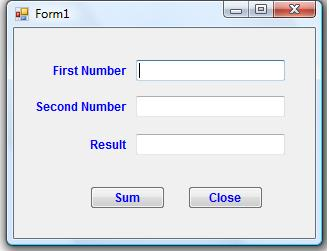
Jedná se o konzolovou aplikaci s uživatelským rozhraním ve formě výpisu textu. Od uživatele je očekáván vstup. Můžeme se ocitnout v situaci, kdy ani žádné menu není potřeba. V tomto případě nám poslouží tzv. Argumenty aplikace. Jedná se o array (string) vstupů při spuštění. Známe ji například z konzolových aplikací, a je velmi podobná klasickému linuxu bez GUI (Graphical User Interface).

Názorným příkladem je zde MacOS bash terminál s běžící Dotnet console aplikací.



## Windows Forms (Project - Windows Form Application)

Jedná se o plnohodnotné GUI (Graphical user interface). Obsahuje velice mnoho funkčních prvků(Label,TextBox,Button,RichTextBox,PictureBox,DataTable…). Využívá namespace System.Windows.Forms. Uživatelské rozhraní tvoříme pomocí designeru, který pouze upravuje hodnoty a přidává kód do třídy Designer.cs. Ta uchovává všechny informace o GUI. Jednotlivé elementy můžeme nalézt v tzv. toolboxu.



## WPF (Windows Presentation Foundation)

WPF (Windows Presentation Foundation) je framework pro komplexní tvorbu bohatých formulářových aplikací, který je součástí .NET frameworku od verze 3.0. Jedná se o nástupce Windows Forms, přičemž disponuje širokou paletou formulářových prvků a také umožňuje bohaté stylování vzhledu aplikace. Několik zásadních výhod představuje WPF například v lepším renderování grafiky, podpoře SVG obrázků.

### Windows Forms vs Windows Presentation Foundation

Dále jen “WF” a “WPF”.

#### Pokročilost

Z pohledu historie a vývoje zde vede WPF, je modernější, lépe pracuje s moderním hardwarem a vzhledem k faktu, že jde o nejmodernější a nejvíce podporovaný a propagovaný systém ze strany Microsoftu.

#### Jednoduchost použití

Jak jsem již zmiňoval, WF je, narozdíl od WPF, primárně určen pro jednodušší aplikace. V praxi to především znamená, že i práce s ním je tedy výrazně jednodušší. WPF na druhou stranu nabízí širší škálu všech možných ovládacích prvků za cenu složitějšího nastavení a uvedení do chodu.

#### Rozvoj aplikace

Primární záměr aplikace napsané ve Windows Form byl primárně “One-time development”. V praxi to znamená, že vyvinete jednoduchou aplikaci za jediným účelem a o budoucí vývoj se moc nestaráte. WPF je ale určen pro vývoj platforem, které vývojáři plánují nadále udržovat a rozvíjet. Je tomu tak možné z důvodu možné komplexity aplikací, napsaných ve WPF, nebo například podpora prvků od třetích stran, SVG (Scalable Vector Graphics) grafiky a mnoho dalšího.

#### Bezpečnost

Vzhledem ke stáří WF je docela očekávané, že bezpečnost zde nebude nikterak dobře řešená. WF ve zkratce nabízí méně různých opatření proti útokům a možnému využití tzv. “exploitů”. Je tomu proto, že WF aplikace nikdy nebyly určeny ke tvorbě komplikovaných systémů, které například komunikují s databázemi apod. *- Respektive byly v době, kdy nebyl k dispozici WPF.*

WPF disponuje více bezpečnostními prvky a využívá moderních technologií na ochranu před hrozícími útoky.

#### Design

WF nebyl nikdy primárně vytvořen ani zamýšlen jako platforma pro tvorbu aplikací obsahující komplexní design. Je zde podpora pouze rastrové grafiky, která je pro moderní dobu nedostačující. WF je ale také možné “naskinovat”, takže vypadá poměrně k světu. Nejde ale o velmi šťastné řešení, jak si poradit s problémem ohledně grafiky aplikace. WPF naopak nabízí moderní, jednoduché a velmi dobře zpracované metody, pro použití grafiky libovolného typu.

#### Rychlost

WF pracuje pomaleji oproti WPF, WF je prostě a jednoduše starší a méně efektivní.

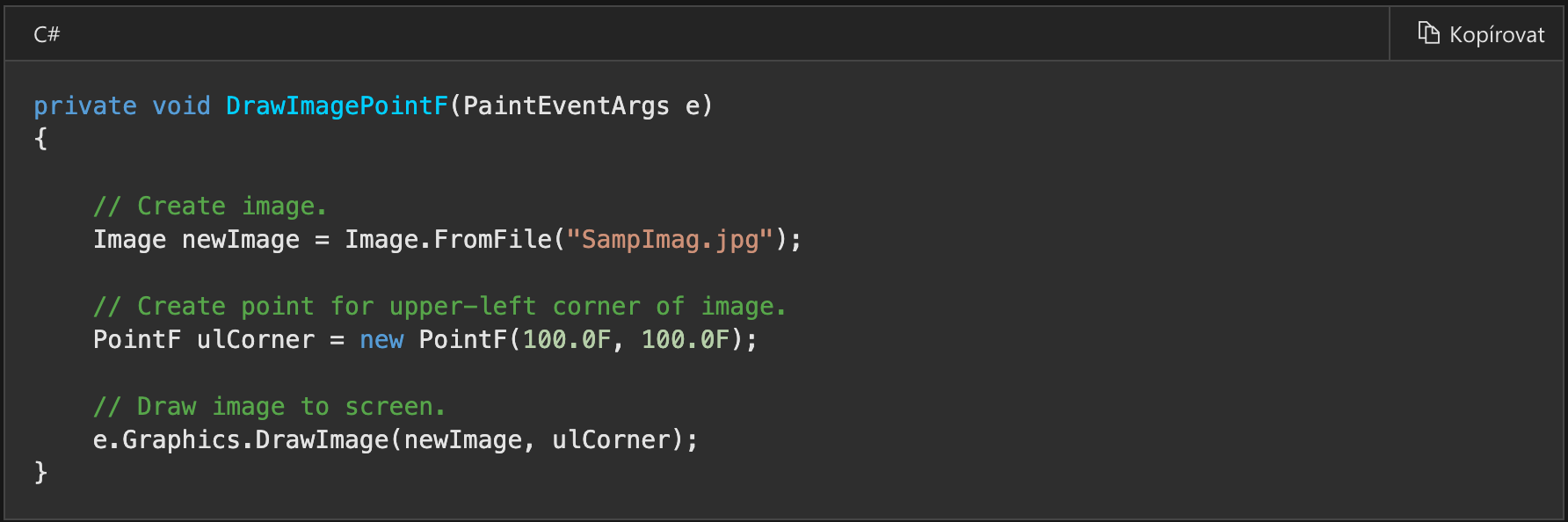
## System.drawing

Následující příklad kódu je navržen pro použití s model Windows Forms a vyžaduje objekt

PaintEventArgs. Objekt PaintEventArgs má název e a je parametrem obslužné rutiny

události Paint. Kód provádí následující akce:

* Vytvoří obrázek ze souboru JPEG. Soubor má název SampImag. jpg a je umístěn ve složce v příkladu.
* Vytvoří bod, ve kterém se má vykreslit levý horní roh obrázku.
* Nakreslí obrázek bez měřítka na obrazovku pomocí objektu Graphics.



# Kontrola vstupů

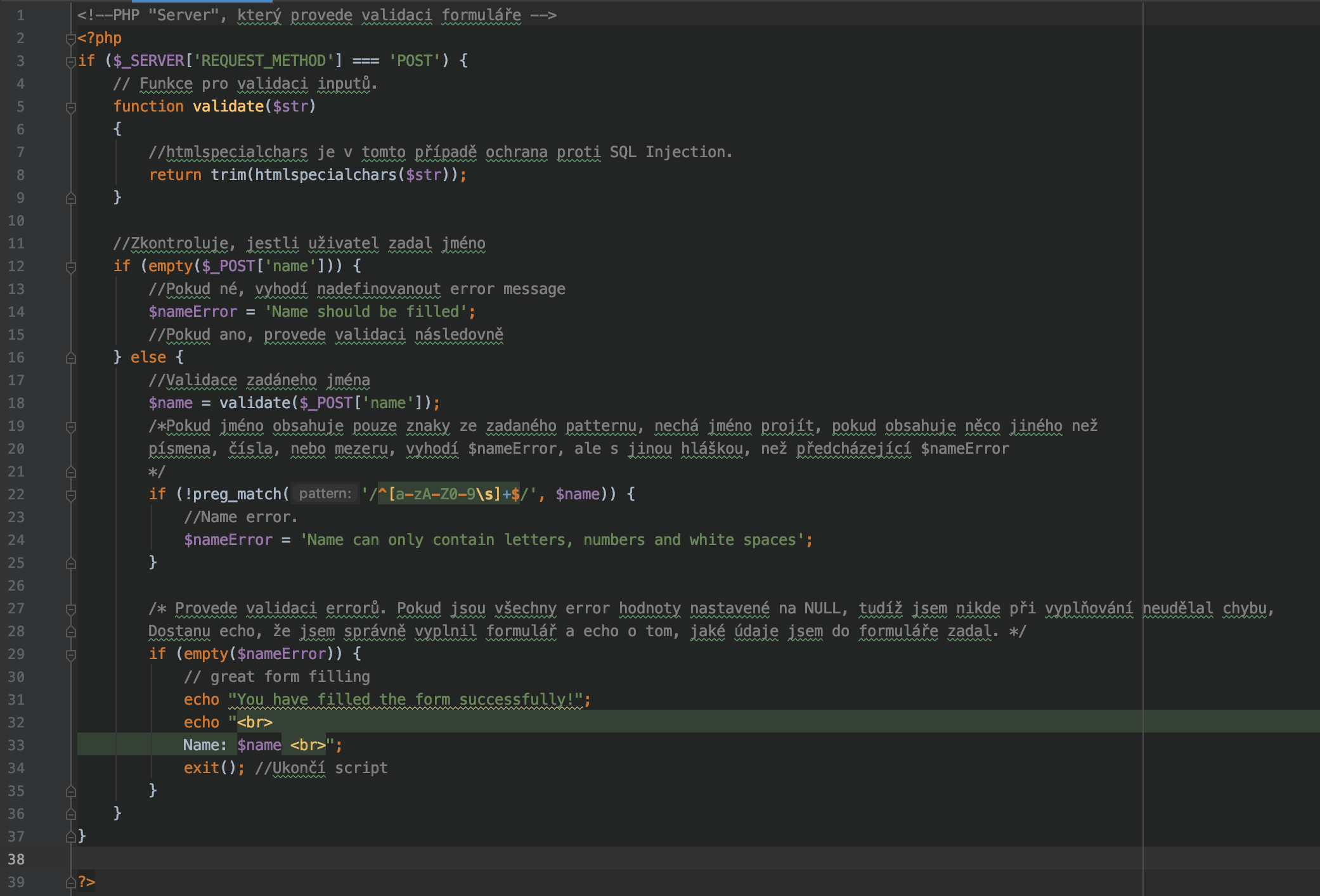
*Lépe dohledáte pod názvem Ošetření vstupů. Kontrolu vstupů budu ukazovat na jednoduchém HTML formuláři a validaci budu provádět přes PHP - jednak zabijete hodně času vysvětlováním, jak to vlastně funguje a je to docela dost opakujícího se kódu, takže když bude nejhůř, začnete programovat. Jde taky o docela fajn příklad, jak funguje taková validace reálného formuláře. Pokud někdo nechce dělat příklad v PHP, tak sem může dodat příklad v jiném jazyce.*

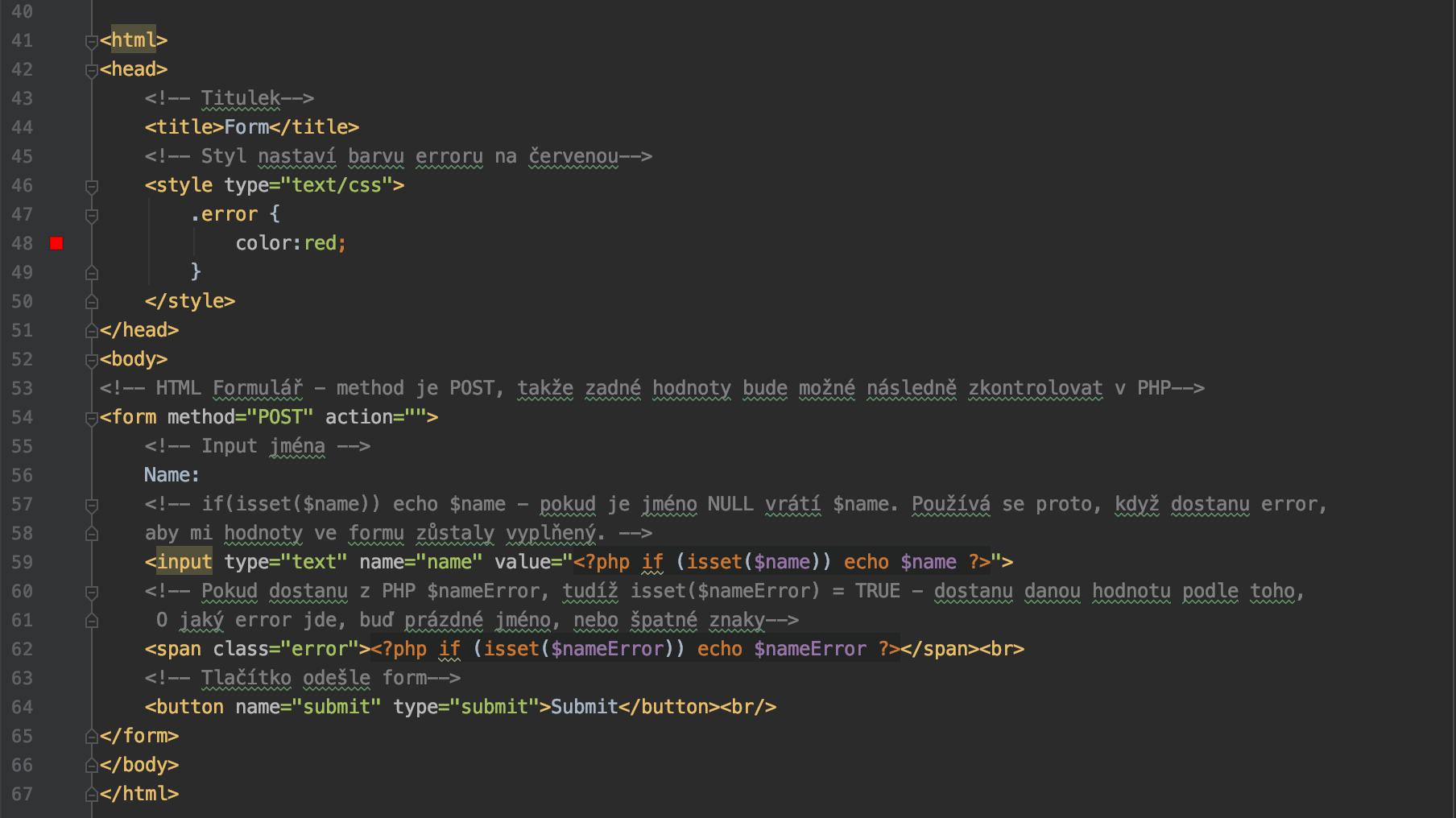
*Další disclaimer je ten, že dneska se už nepoužívá klasické PHP, ale frameworky. Nette, Symphony, atd.* Kontrola vstupů je velice jednoduchá záležitost, avšak dá se o ní poměrně dlouho mluvit.

Jedná se o to, že uživatel vám na vstupu zadá nějaké údaje, může jít o jméno, příjmení, email, heslo, apod. A úkolem kontroly vstupů je postarat se o to, zda-li zadané údaje korespondují s formátem, který je od uživatele očekáván.

Jednoduchý příklad je kolonka **jméno.** Od uživatele očekávám text, popřípadě číslo (může se jednat o arabštinu, nebo hebrejštinu psanou latinkou, která obsahuje čísla). Poté daný text “proženeme” filtrem, který například u jména zkontroluje, zda obsahuje pouze znaky v zadaném rozsahu (viz. screenshoty dole) a poté buď vrátí $nameError, nebo nechá input projít a $nameError zůstane **NULL.**

Dole jsou k dispozici screenshoty jednoduché validace inputu (pouze jméno). Celý PHP formulář s dokumentací (index.php) k dispozici v main directory (PV - Otázky).





# 

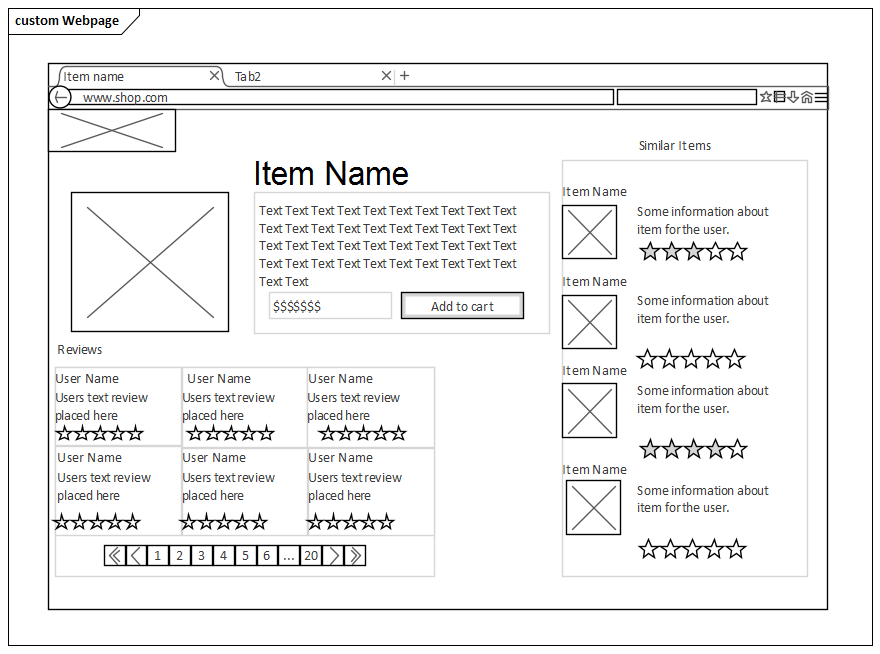
# Layout

*Nemám nejmenší tušení co k tomuhle pojmu napsat, tak prostě vysvětlím, co to je layout z hlediska UI.*

Layout znamená grafické rozvržení tiskové nebo elektronické stránky, případně i jiné plochy. Zahrnuje obvykle: formát stránky a tiskové zrcadlo, rozvržení textu, obrázků a prázdných ploch, typ, velikost a barvy písma, záhlaví a nadpisy další grafické prvky výběr barev nebo motivů pozadí, případně ještě další. [Wikipedie](https://cs.wikipedia.org/wiki/Layout).

V případě programování aplikace musíme navrhnout takzvaný Wireframe, což je vlastně rozvržení jednotlivých prvků na obrazovce (nevím proč bychom to sice dělali jako programátoři, když to mají na práci UX a UI designéři, ale whatever).

Takhle zhruba vypadá Wireframe. Můžeme na něm vidět, že jde o webovou aplikaci a následně jednotlivá místa, kde má být text, pro obrázky se pak využívá proškrtnutý čtverec/obdélník. Dále vidíme tlačítka, hvězdičky apod. Takovýto Wireframe slouží jak pro designéra, který pro daný wireframe navrhne design, tak pro programátora, který tak díky němu vidí, jak má aplikace fungovat, různé návazností, apod.



Při programování můžeme využít například Bootstrap, který obsahuje už různé hotové layouty například pro vyhledávač, menu, apod.

Ovládací prvky

V první řadě můžeme ovládací prvky rozdělit na virtuální a fyzické.

**Fyzické ovládací prvky** jsou například klávesnice, myš, různé stream decky, fyzické audio controllery, apod.

Mám pocit, že jste všichni myš a klávesnici viděli, tak sem střihnu obrázek Loupedecku (Cool fyzickej interface pro Adobe Lightroom)



**Virtuální ovládací prvky** (označovány také jako widgety) jsou prvky, se kterými se můžete setkat prakticky všude. Virtuálních prvků je vážně hrozně moc, tak sem dám jen některé, ale asi všichni tušíte, o co jde.

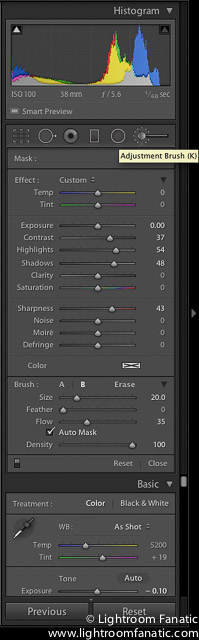
Tlačítko (button)

Šoupátko (slider)

Nabídka (menu)

Posuvník (scrollbar)

Ten Loupedeck nahoře je fyzický ovládací prvek a toto je jeho virtuální ekvivalent.

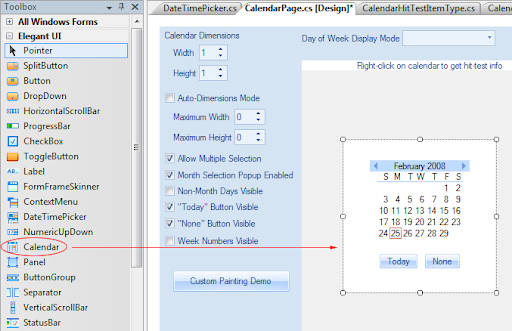


Fyzický ovládací prvek můžeme využít k jednoduššímu ovládání virtuálního prvku (viz ty obrázky). Když pohnu sliderem na Loupedecku, pohne se mi i slider v Lightroomu.

Tohle je tady jen tak abyste mohli o něčem cool kecat. Ovládací prvky jsou v naprosto každé GUI aplikaci, na každém elektronickém zařízení, fakt úplně všude, ukažte příklady na něčem, co bude po ruce.

## Ovládací prvky ve Windows Form

Tohle je fakt jednoduchý, pokud chcete do vaší WF aplikace přidal nějakej ovládací prvek, prostě ho vyberete z Toolboxu, přetáhnete do Designéra a nastavíte velikost, barvu atd. Pak už pro něj jen naprogramujete funkčnost.



Události

Občas je v programu potřeba reagovat na nějakou událost. Ideálním praktickým příkladem jsou již zmíněné formulářové aplikace. Když se klikne na tlačítko, spustí se obslužná metoda, která vykoná nějakou akci. Událostí bychom vymysleli mnoho, např. minimalizace okna, pohyb myší nebo třeba výpadek připojení k internetu. Bez znalosti principu událostí (a delegátů) bychom asi takovou aplikaci zvládli naprogramovat, ale byla by velmi nepřehledná a bylo by složité události přidávat nebo měnit.

Pro vizualizaci mi přijde tenhle obrázek asi nejlepší

