Objektově orientované programování

Přednáška č. 6

Ing. Lukáš Slánský

Ústav elektrotechniky a informatiky, Univerzita Pardubice

23. 11. 2005

Obsah

Trojvrstvý model aplikace

- Dobře napsaný program (informační systém) se skládá z několika logicky oddělených částí:

Trojvrstvý model aplikace

- Dobře napsaný program (informační systém) se skládá z několika logicky oddělených částí:
  - Vizualizace údajů uživateli (UI, WEB, tenký klient, . . .),

Trojvrstvý model aplikace

- Dobře napsaný program (informační systém) se skládá z několika logicky oddělených částí:
  - Vizualizace údajů uživateli (UI, WEB, tenký klient, . . .),
  - ukládání a správa dat (databáze, filesystém, . . .),

Trojvrstvý model aplikace

- Dobře napsaný program (informační systém) se skládá z několika logicky oddělených částí:
  - Vizualizace údajů uživateli (UI, WEB, tenký klient, ...),
  - ukládání a správa dat (databáze, filesystém, ...),
  - zpracování dat.

Trojvrstvý model aplikace

- Dobře napsaný program (informační systém) se skládá z několika logicky oddělených částí:
  - Vizualizace údajů uživateli (UI, WEB, tenký klient, ...),
  - ukládání a správa dat (databáze, filesystém, ...),
  - zpracování dat.
- Aby byl program dobře spravovatelný, je vhodné jednotlivé logické části oddělit do nezávislých částí – bloků, vrstev.

Trojvrstvý model aplikace

- Dobře napsaný program (informační systém) se skládá z několika logicky oddělených částí:
  - Vizualizace údajů uživateli (UI, WEB, tenký klient, ...),
  - ukládání a správa dat (databáze, filesystém, ...),
  - zpracování dat.
- Aby byl program dobře spravovatelný, je vhodné jednotlivé logické části oddělit do nezávislých částí – bloků, vrstev.
- Jednotlivé vrstvy spolu komunikují pomocí rozhraní.

Prezentační vrstva

- *Prezentační vrstva* je zodpovědná za zobrazení obsahu uživateli, tisky atd.

Prezentační vrstva

- *Prezentační vrstva* je zodpovědná za zobrazení obsahu uživateli, tisky atd.
- Poskytuje nástroje pro spolupráci mezi uživatelem a aplikací

Prezentační vrstva

- *Prezentační vrstva* je zodpovědná za zobrazení obsahu uživateli, tisky atd.
- Poskytuje nástroje pro spolupráci mezi uživatelem a aplikací – umožňuje uživateli zadávat příkazy pro IS.

Prezentační vrstva

- *Prezentační vrstva* je zodpovědná za zobrazení obsahu uživateli, tisky atd.
- Poskytuje nástroje pro spolupráci mezi uživatelem a aplikací – umožňuje uživateli zadávat příkazy pro IS.
- V systémech typu server-klient jde o

Prezentační vrstva

- *Prezentační vrstva* je zodpovědná za zobrazení obsahu uživateli, tisky atd.
- Poskytuje nástroje pro spolupráci mezi uživatelem a aplikací – umožňuje uživateli zadávat příkazy pro IS.
- V systémech typu server-klient jde o klientskou část.

Business vrstva

- *Bsiness vrstva* implementuje hlavní logiku aplikace.

Business vrstva

- *Bsiness vrstva* implementuje hlavní logiku aplikace.
- Z prezentační vrstvy přebírá příkazy z UI a reaguje na ně.

Business vrstva

- *Bsiness vrstva* implementuje hlavní logiku aplikace.
- Z prezentační vrstvy přebírá příkazy z UI a reaguje na ně.
- Jsou zde prováděny všechny výpočty, validace dat atd.

Business vrstva

- *Bsiness vrstva* implementuje hlavní logiku aplikace.
- Z prezentační vrstvy přebírá příkazy z UI a reaguje na ně.
- Jsou zde prováděny všechny výpočty, validace dat atd.
- Koordinuje a zprostředkovává tok dat mezi prezentační vrstvou a datovou vrstvou.

Business vrstva

- *Bsiness vrstva* implementuje hlavní logiku aplikace.
- Z prezentační vrstvy přebírá příkazy z UI a reaguje na ně.
- Jsou zde prováděny všechny výpočty, validace dat atd.
- Koordinuje a zprostředkovává tok dat mezi prezentační vrstvou a datovou vrstvou.
- V systémech typu server-klient

Business vrstva

- *Bsiness vrstva* implementuje hlavní logiku aplikace.
- Z prezentační vrstvy přebírá příkazy z UI a reaguje na ně.
- Jsou zde prováděny všechny výpočty, validace dat atd.
- Koordinuje a zprostředkovává tok dat mezi prezentační vrstvou a datovou vrstvou.
- V systémech typu server-klient je z části na klientské a z části na serverové straně.

Datová vrstva

- *Datová vrstva* je zodpovědná za správu dat.

Datová vrstva

- *Datová vrstva* je zodpovědná za správu dat.
- Poskytuje business vrstvě veškerá zpracovaná data, která poté mohou být zobrazena uživateli pomocí prezentační vrstvy.

Datová vrstva

- *Datová vrstva* je zodpovědná za správu dat.
- Poskytuje business vrstvě veškerá zpracovaná data, která poté mohou být zobrazena uživateli pomocí prezentační vrstvy.
- Přebírá od business vrstvy vložená (změněná) data a další požadavky na manipulaci s daty.

Datová vrstva

- *Datová vrstva* je zodpovědná za správu dat.
- Poskytuje business vrstvě veškerá zpracovaná data, která poté mohou být zobrazena uživateli pomocí prezentační vrstvy.
- Přebírá od business vrstvy vložená (změněná) data a další požadavky na manipulaci s daty.
- V systémech typu server-klient jde o

Datová vrstva

- *Datová vrstva* je zodpovědná za správu dat.
- Poskytuje business vrstvě veškerá zpracovaná data, která poté mohou být zobrazena uživateli pomocí prezentační vrstvy.
- Přebírá od business vrstvy vložená (změněná) data a další požadavky na manipulaci s daty.
- V systémech typu server-klient jde o serverovou část.

Požadavky na trojvrstvou aplikaci

- Jednotlivé vrstvy jsou vyměnitelné.

Požadavky na trojvrstvou aplikaci

- Jednotlivé vrstvy jsou vyměnitelné.
  - Datová vrstva, která pracuje s daty na pevném disku,
  - datová vrstva pracující s databází Oracle,
  - datová vrstva pracující s páskovou pamětí,
  - ...

Požadavky na trojvrstvou aplikaci

- Jednotlivé vrstvy jsou vyměnitelné.
  - Datová vrstva, která pracuje s daty na pevném disku,
  - datová vrstva pracující s databází Oracle,
  - datová vrstva pracující s páskovou pamětí,
  - ...
- Při výměně je nutné dodržet rozhraní.

Požadavky na trojvrstvou aplikaci

- Jednotlivé vrstvy jsou vyměnitelné.
  - Datová vrstva, která pracuje s daty na pevném disku,
  - datová vrstva pracující s databází Oracle,
  - datová vrstva pracující s páskovou pamětí,
  - ...
- Při výměně je nutné dodržet rozhraní.
  - Nejlépší je vytvořit abstraktní třídu, která specifikuje rozhraní.

Požadavky na trojvrstvou aplikaci

- Jednotlivé vrstvy jsou vyměnitelné.
  - Datová vrstva, která pracuje s daty na pevném disku,
  - datová vrstva pracující s databází Oracle,
  - datová vrstva pracující s páskovou pamětí,
  - ...
- Při výměně je nutné dodržet rozhraní.
  - Nejlépší je vytvořit abstraktní třídu, která specifikuje rozhraní.
  - Z abstraktní třídy vydědit třídu, která implementuje přesné chování dané vrstvy.

Dynamické vytváření objektů

- Rozšířená syntaxe procedury `New` pro použití s objekty:

Dynamické vytváření objektů

- Rozšířená syntaxe procedury `New` pro použití s objekty:

```
procedure New(p:^Třída, Init:Constructor);
```

Dynamické vytváření objektů

- Rozšířená syntaxe procedury `New` pro použití s objekty:

  ```
  procedure New(p:^Třída, Init:Constructor);
  ```

  - Alokuje objekt z třídy `Třída` na haldě a automaticky zavolá konstruktor.

Dynamické vytváření objektů

- Rozšířená syntaxe procedury `New` pro použití s objekty:

  ```
  procedure New(p:^Třída, Init:Constructor);
  ```

  - Alokuje objekt z třídy `Třída` na haldě a automaticky zavolá konstruktor.
- Rozšířená syntaxe procedury `New` pro použití s objekty:

Dynamické vytváření objektů

- Rozšířená syntaxe procedury `New` pro použití s objekty:

  ```
  procedure New(p:^Třída, Init:Constructor);
  ```

  - Alokuje objekt z třídy `Třída` na haldě a automaticky zavolá konstruktor.
- Rozšířená syntaxe procedury `New` pro použití s objekty:

  ```
  function New(T:^Třída, Init:Constructor):^Typ;
  ```

Dynamické vytváření objektů

- Rozšířená syntaxe procedury `New` pro použití s objekty:

  ```
  procedure New(p:^Třída, Init:Constructor);
  ```

  - Alokuje objekt z třídy `Třída` na haldě a automaticky zavolá konstruktor.
- Rozšířená syntaxe procedury `New` pro použití s objekty:

  ```
  function New(T:^Třída, Init:Constructor):^Typ;
  ```

  - Alokuje objekt z třídy `Třída` a automaticky zavolá konstruktor.

Dynamické vytváření objektů

- Rozšířená syntaxe procedury `New` pro použití s objekty:

  ```
  procedure New(p:^Třída, Init:Constructor);
  ```

  - Alokuje objekt z třídy `Třída` na haldě a automaticky zavolá konstruktor.
- Rozšířená syntaxe procedury `New` pro použití s objekty:

  ```
  function New(T:^Třída, Init:Constructor):^Typ;
  ```

  - Alokuje objekt z třídy `Třída` a automaticky zavolá konstruktor.
  - Využití s dědičností. Lze vytvořit objekt z třídy potomka a přiřadit ho do proměnné typu předka:

Dynamické vytváření objektů

- Rozšířená syntaxe procedury `New` pro použití s objekty:

  ```
  procedure New(p:^Třída, Init:Constructor);
  ```

  - Alokuje objekt z třídy `Třída` na haldě a automaticky zavolá konstruktor.

- Rozšířená syntaxe procedury `New` pro použití s objekty:

  ```
  function New(T:^Třída, Init:Constructor):^Typ;
  ```

  - Alokuje objekt z třídy `Třída` a automaticky zavolá konstruktor.
  - Využití s dědičností. Lze vytvořit objekt z třídy potomka a přiřadit ho do proměnné typu předka:

    ```
    Predek:=New(PPotomek, Init);
    ```

Dynamické vytváření objektů

- Rozšířená syntaxe procedury `New` pro použití s objekty:

  ```
  procedure New(p:^Třída, Init:Constructor);
  ```

  - Alokuje objekt z třídy `Třída` na haldě a automaticky zavolá konstruktor.

- Rozšířená syntaxe procedury `New` pro použití s objekty:

  ```
  function New(T:^Třída, Init:Constructor):^Typ;
  ```

  - Alokuje objekt z třídy `Třída` a automaticky zavolá konstruktor.
  - Využití s dědičností. Lze vytvořit objekt z třídy potomka a přiřadit ho do proměnné typu předka:

    ```
    Predek:=New(PPotomek, Init);
    Predek:=New(PPotomek, Init(10));
    ```