Objektno orijentirano programiranje - praktikum u .NET okolini

Predavanje 04

Kreiranje WindowsForms kontrola

- Osim kontrola koje dolaze s .NET frameworkom, moguće je kreirati i vlastite kontrole
- Postoji dva tipa kontrola koje možemo kreirati:
  - **Korisničke** kontrole (engl. *user controls*)
    - Najčešće se koriste
  - **Vlastite** kontrole (engl. *custom controls*)

Korisničke kontrole

- Korisničke kontrole nasljeđuju klasu **UserControl**
- Sastoje se od bilo koje **kombinacije postojećih kontrola i komponenti**
  - Zbog toga se nazivaju i **složene** ili **kompozitne** kontrole
- Korisničku kontrolu dodajemo na formu na iste načine kao i ostale kontrole
- Sve kontrole koje se nalaze u korisničkoj kontroli su privatne
- Korisničkoj kontroli možemo dodavati članove:
  - Metode, svojstva, događaje, …

Vlastite kontrole

- Nasljeđuju klasu **Control**
- Pružaju najveću mogućnost podešavanja
- Najzahtjevnije za razvoj
- Ključna karakteristika: kontrola se mora sama iscrtati
  - Iscrtavanje treba napraviti **premošćivanjem** (engl. *override*) metode **OnPaint()**

Validacija

- **Validacija** je postupak provjere ispravnosti podataka koje korisnik unosi u formu
- Komponenta **ErrorProvider** zadužena je za prikaz poruke o neispravnom unosu
- Postupak validacije:
  1. Kontroli koja prima korisnikov unos (npr. TextBox) postavimo svojstvo **CausesValidation** na **true** (predefinirana vrijednost)
  2. Na istoj kontroli implementiramo metodu za obradu događaja **Validating**
     1. Ako nema greške, na instanci ErrorProvidera pozovemo metodu **SetError()** s praznim stringom
     2. Ako ima greške, na instanci ErrorProvidera pozovemo metodu **SetError()** s porukom greške

Globalizacija i lokalizacija

- **Globalizacija**
  - Prikaz podataka (vrijeme, datum, valuta, broj, ...) u formatima prilagođenim određenoj kulturi
  - Primjerice:
    - U Hrvatskoj broj pišemo kao **1.800,00**
    - U SAD-u se isti broj piše kao **1,800.00**
- **Lokalizacija**
  - Prikaz podataka na jeziku određene kulture (prijevod)
  - Primjerice:
    - Naslov forme na hrvatskom može biti "**Boja**"
    - Naslov iste forme na engleskom može biti "**Color**"

Kultura

- Pojam **kulture** (engl. *culture*) u .NET-u podrazumijeva sljedeće elemente:
  - Jezik
  - Pismo (opcionalno)
  - Regiju (opcionalno)
- **Neutralna kultura**: sadrži samo informaciju o jeziku
  - Jezik: **hr, sr, en, ...**
- **Specifična kultura**: specificira jezik i regiju (može i pismo)
  - Jezik i regija: **hr-HR, hr-BA, en-US, en-UK, en-CA, en-AU, ...**
  - Jezik, pismo i regija: **sr-Cyrl-BA, sr-Cyrl-CS, sr-Latn-BA, sr-Latn-CS, ...**

Promjena kulture

- Kultura je implementirana u klasi **CultureInfo**
  - Sadrži informacije o formatu vremena, datuma, valute, ...
- **Globalizacija** se postavlja pomoću:
  - Thread.CurrentThread.**CurrentCulture**
  - Predefinirano se primjenjuje kultura odabranih regionalnih postavki
- **Lokalizacija** se postavlja pomoću:
  - Thread.CurrentThread.**CurrentUICulture**
  - Predefinirano se primjenjuje kultura (jezik) operativnog sustava
  - Određuje koji resursi će biti učitani u lokalizirane forme

Lokalizirane forme

- Svaka forma može imati više verzija - po jednu za svaku željenu kulturu
  - Kulturu biramo pomoću **CurrentUICulture**
  - Nakon odabira, aplikacija će učitati resurse zadane kulture
  - Ako resursi ne postoje, učitat će resurse podrazumijevane kulture
  - Odabir kulture potrebno je odraditi prije prikaza forme
    - Moguće je i dinamički promijeniti lokalizirane stringove na formi
- Bitna svojstva forme za lokalizaciju:
  - **Localizable** – ako je postavljeno na vrijednost **true**, dizajner svojstva forme i kontrola čuva u resursnim datotekama
  - **Language** – dizajner prikazuje odabranu lokaliziranu verziju forme

Ispis (printanje)

- Komponenta zadužena za ispis (engl. *print*) je **PrintDocument**
- Princip rada je sljedeći:

1. Na instanci klase **PrintDocument** definira se metoda za obradu događaja **PrintPage**
2. Na instanci klase **PrintDocument** poziva se metoda **Print()**
3. Podići će se događaj **PrintPage** za prvu stranicu
   - U njemu se radi ispis na papir pomoću **Graphics** klase
   - Na kraju ispisa postavlja se vrijednost svojstva **HasMorePages** na instanci klase **PrintPageEventArgs**
4. Dok god je **HasMorePages** jednako **true**, podiže se događaj **PrintPage**
5. Kad je dokument poslan na ispis, podiže se događaj **EndPrint**

Pomoćne kontrole za ispis

- Prilikom ispisa možemo koristiti pomoćne kontrole:
  - **PrintDialog**: predstavlja dijalog za odabir printera, stranica koje želi printati, broj kopija, ...
  - **PageSetupDialog**: predstavlja dijalog za odabir veličine papira, orijentacije stranice, margina, ...
  - **PrintPreviewDialog**: predstavlja dijalog za pregled dokumenta prije ispisa
  - **PrintPreviewControl**: koristi se za razvoj vlastitog dijaloga za pregled dokumenta prije ispisa
- Sve navedene kontrole povezuju se s **PrintDocument** komponentom preko svojstva **Document**