



# Windows Workflow Foundation

**Josip Šaban**

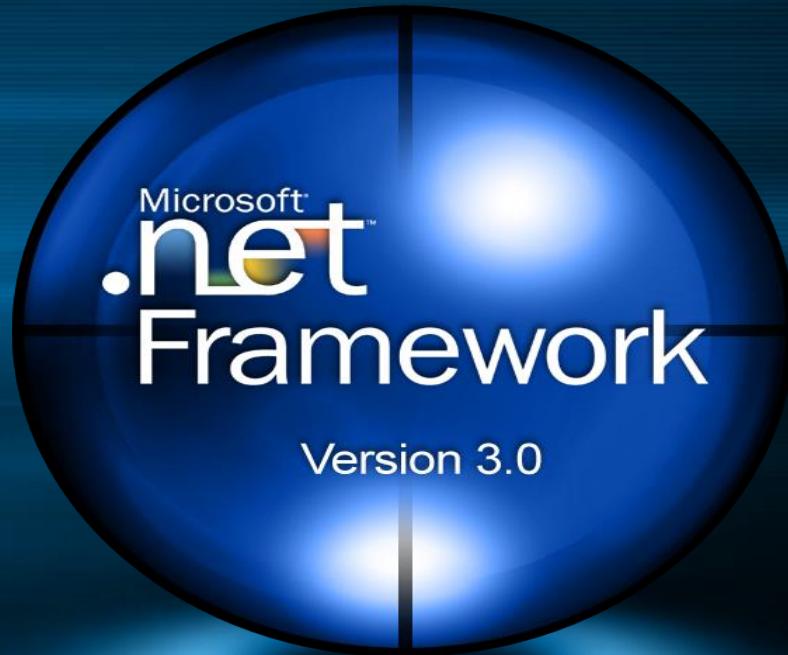
BI konzultant, crmT d.o.o  
Comptia Project+  
MCSD, MCDBA, MCPD: Windows, Web, Enterprise  
MCTIP:Developer, Administrator

# WF != WWF



```
101000100100101001010101001100  
10101010010101010101010101010101  
10101010101111110110100001010110  
101000100100101001010101001100  
10101010010101010101010101010101  
10101010101111110110100001010110
```

# U pozadini se nalazi .NET 3.0



101000100100101001010101001100  
10101010010101010101010101010101  
101010101011111101101000010101  
101000100100101001010101001100  
10101010010101010101010101010101  
101010101011111101101000010101

# Windows Workflow Foundation

Programski model,  
engine i alati  
za izgradnju Workflow  
osposobljenih aplikacija  
na Windows platformi.



# Workflow izazovi

“Narudžbe se potvrđuju unutar 48 sati i šalju unutar 30 dana”

“Većina dobavljača potvrđuje naše narudžbe, ali neki zaborave”

“Koji je status naše narudžbe i koji je slijedeći korak?”

## Long Running & Stateful

Workflow-i se mogu izvršavati do 30 dana i u tom periodu moraju čuvati stanje

## Require Flexible Control Flow

Fleksibilnost koja omogućava ljudima da promjene ili preskoče korake

## Must Provide Transparency

Renderiranje trenutnog stanja u vizualnom obliku kontrole workflow-a

# BizTalk Server

Dizajn.  
alati

Accelerators

Or **Workflow** Foundation

Messaging

Transformation

Adapters

Business  
Activity  
Monitor.  
i  
Admin.  
alati

- Serverski proizvod
- Korišten u B2B, EAI, BPM scenarijima
- Instalirano rješenje
- Upravljanje, scale-out
- Buduće verzije će migrirati na Workflow Foundation za orkestracije

Visual Studio Designer

Windows Workflow  
Foundation

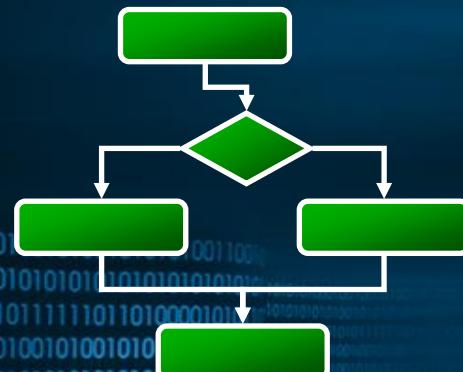
.NET Framework 3.0

- Dostupan preko .NET-a 3.0
- Širok set scenarija
- Korišten u izgradnji rješenja
- Omogućava održavanje i scale-out rješenja
- Korišteno za ugradnju workflowa u aplikacije

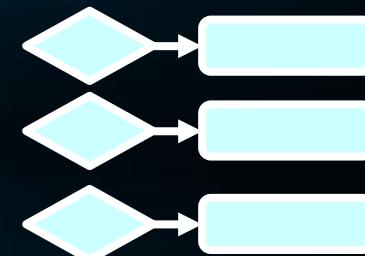
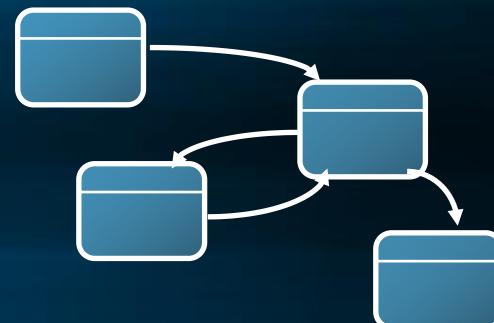
# Što je workflow?

- Program izrađen kao niz aktivnosti
  - Koordinira ljudi i softver
  - Ima kontrolu toka podataka u stvarnom vremenu
  - Izvršava se transparentno
  - Dopoljuje dinamičke promjene

Kao flowchart....



ili dijagram stanja.... Ili temeljen na pravilima.



# Windows Workflow Foundation

*Workflow platforma za Microsoftove  
proizvode*

- Jedinstvena workflow tehnologija za Microsoft
- Framework za ugradnju workflow-a u aplikacije
  - Dio .NET Framework 3.0
- Uvodi deklarativni workflow “među mase”

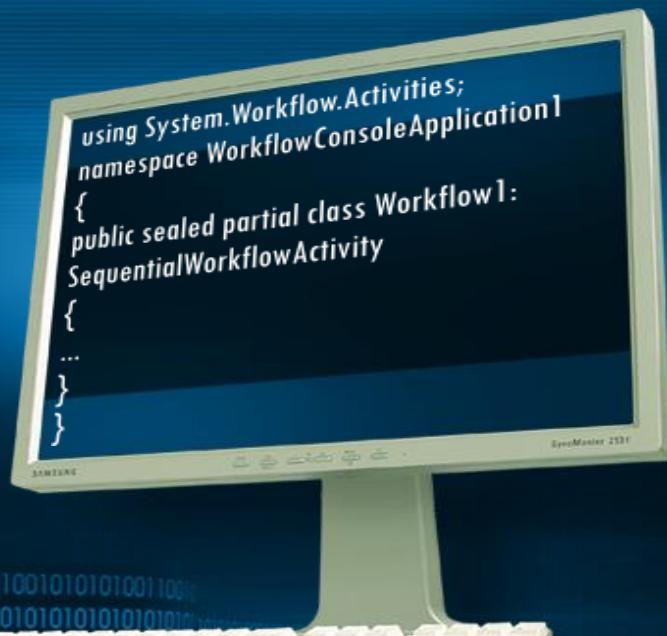
# WF arhitektura



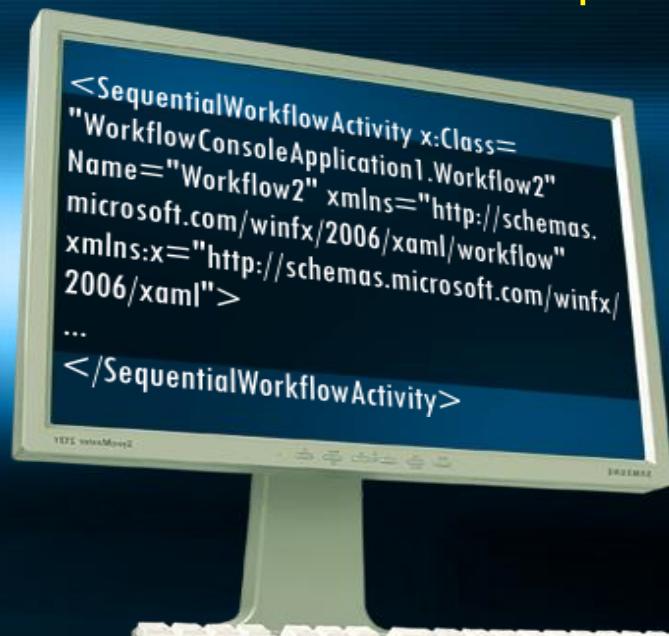
Vizualni  
dizajner

# Workflow osnove

Workflow je klasa



Workflow klasa se može definirati i u markupu

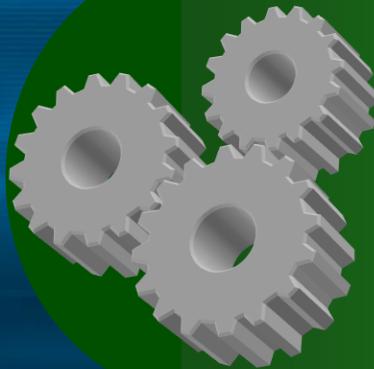


# Nova filozofija razvoja

Forme



Podaci



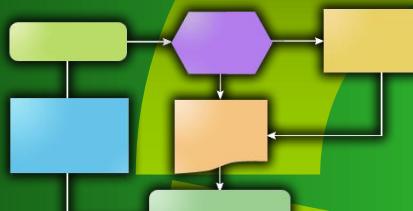
Programska  
logika

Model-driven razvoj

Pravila



Programski  
kod



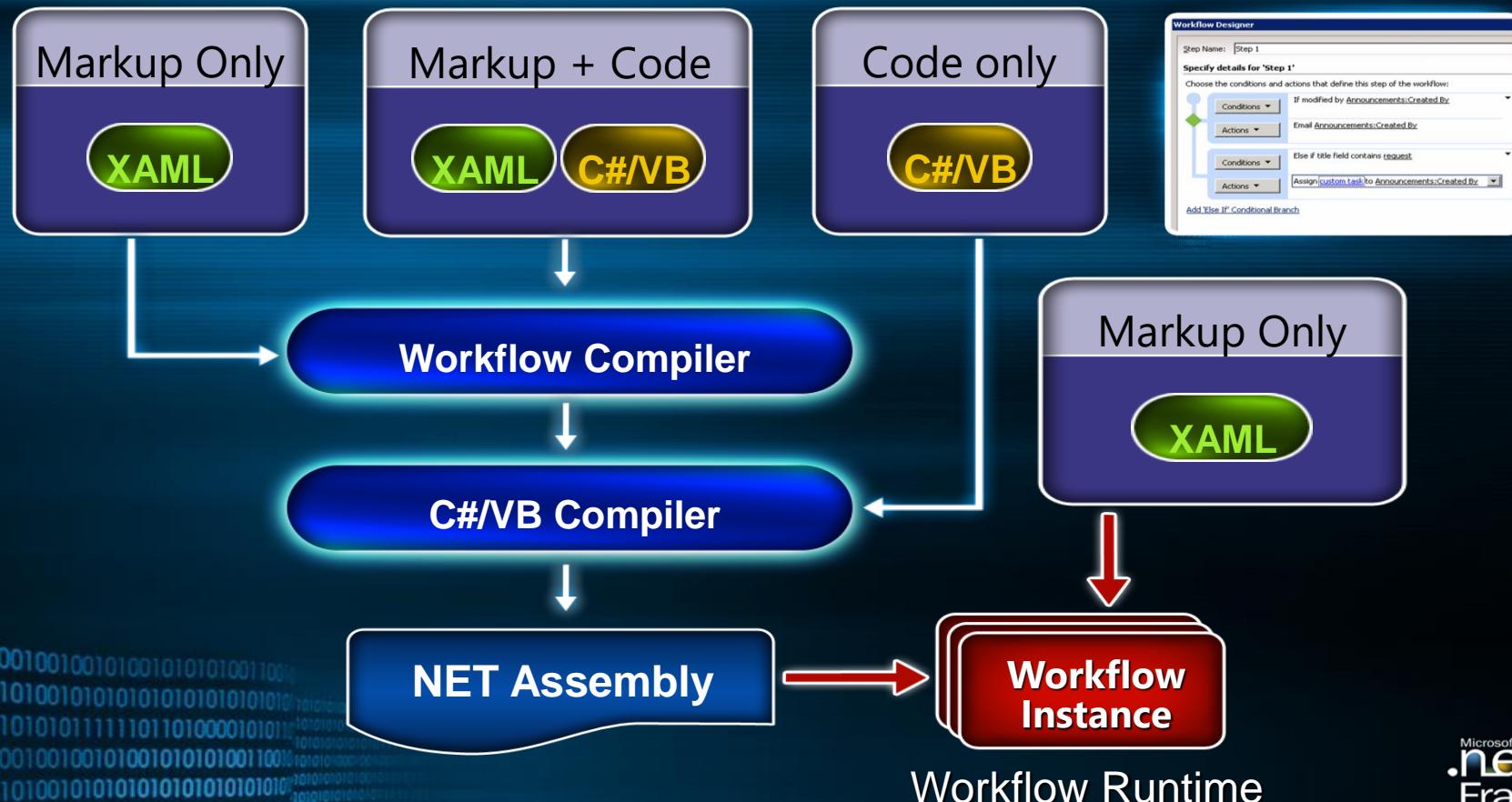
Workflow  
model



Usluge

Framework

# Načini izrade



# Workflow-i i aktivnosti

## Activity

```
public class Activity  
{  
    override Execute()  
    { ...  
    }  
}
```

## Workflow

```
public class MyWorkflow  
{  
    MyWorkflow()  
    {  
        Activities.Add(...);  
    }  
}
```

## Activity

```
public class Activity  
{  
    override Execute()  
    { ...  
    }  
}
```

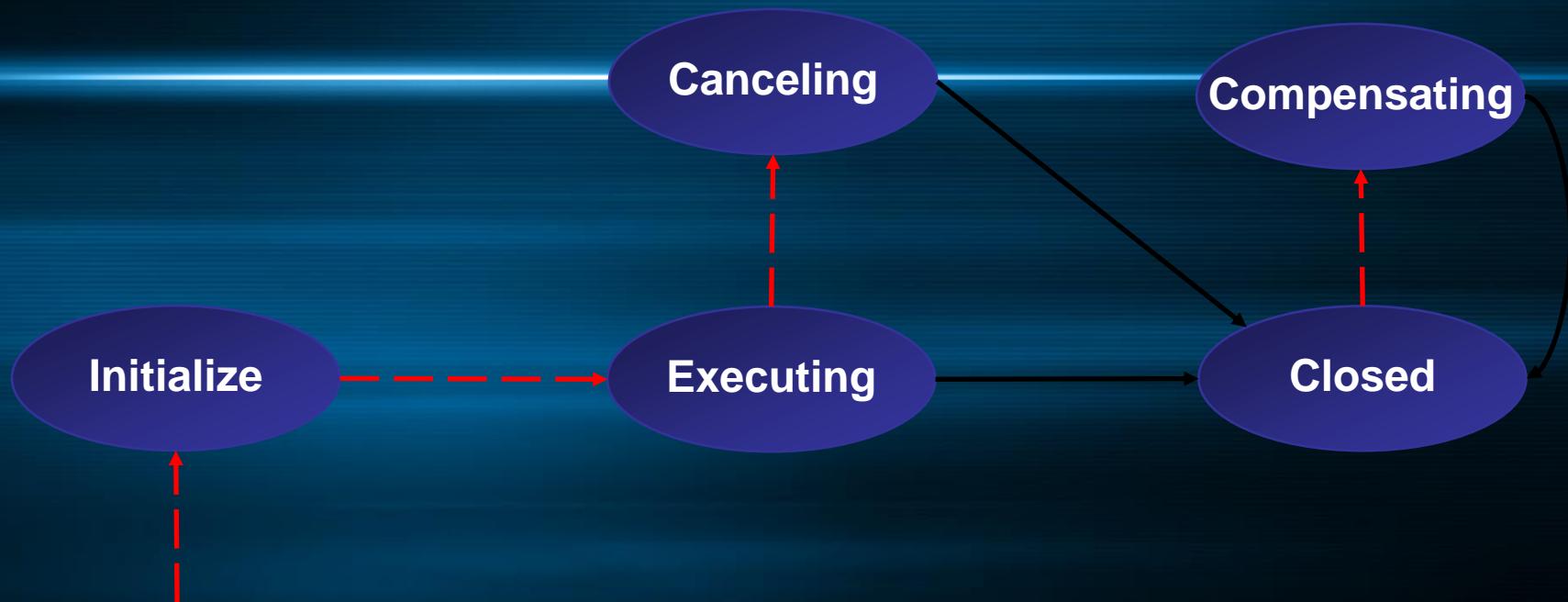
- Workflow je .NET klasa

- Sastoji se od aktivnosti ( isto .NET klasa )

- Aktivnosti se mogu sastojati od još klase

- Ugrađene su u asemblerije i referencirane kao i svaka druga klasa

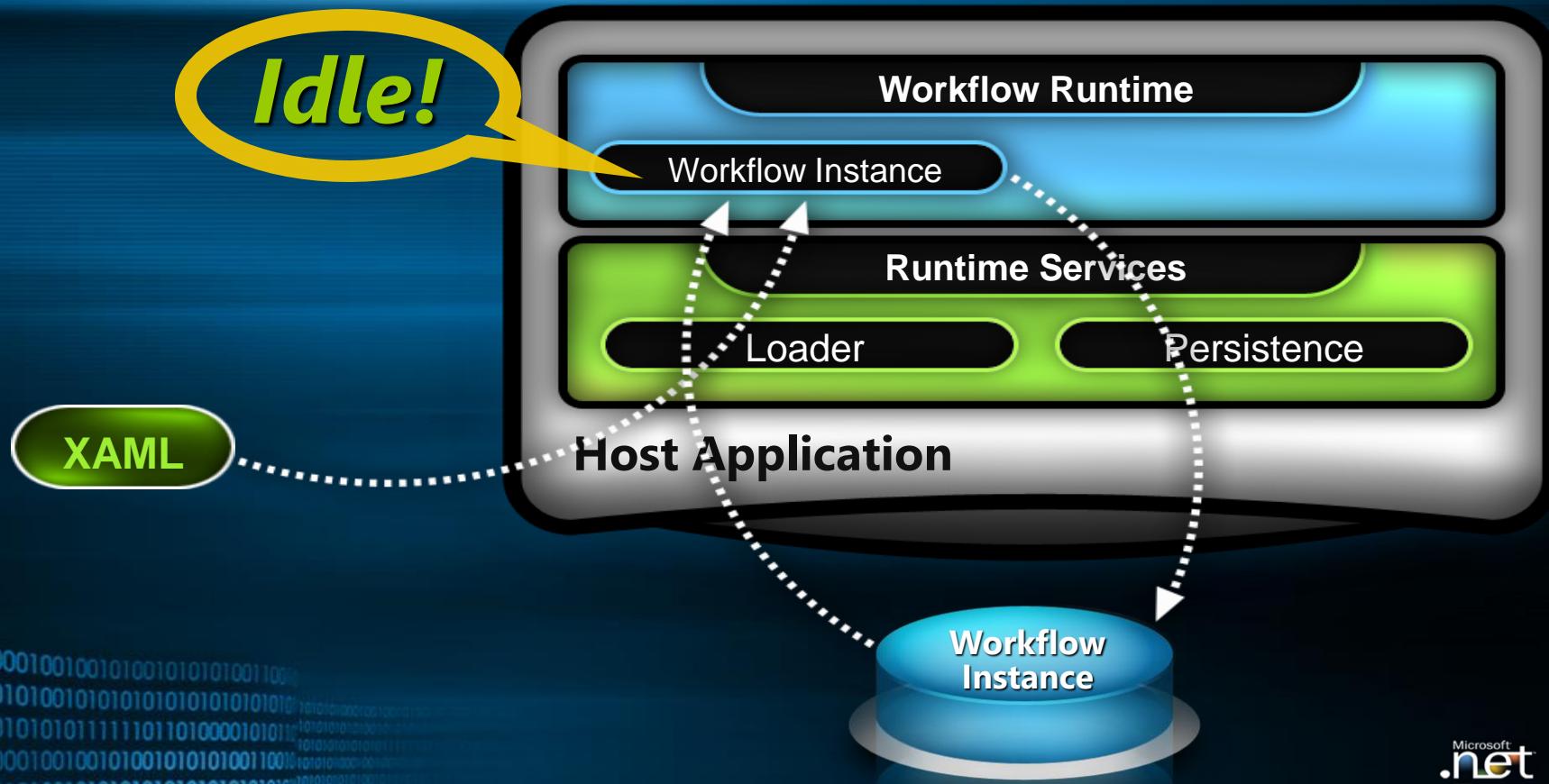
# Stanja izvršavanja aktivnosti



Vrste prijelaza:

- > Runtime
- > Activity

# Izvršavanje workflow-a



# Osnove aktivnosti

Aktivnosti su osnovni dio svakog workflow-a

- Jedinica izvršavanja i ponovnog korištenja
- Osnovne aktivnosti su koraci u workflowu
- Kompozitne aktivnosti se sastoje od drugih aktivnosti
- Base Activity Library pruža out-of-the-box aktivnosti
- Partneri i kupci mogu stvarati vlastite aktivnosti

# Uvod u prvi demo

- Validacija poštanskog broja
- Biramo “Sequential Workflow Console Application”
- Osnovni primjer da se pokaže princip rada
- Razvojna okolina:
  - Windows XP SP3
  - Visual Studio 2008 ( +SP1 )
  - SQL Server 2008 Express
- Pretpostavka predavanja:
  - Iskustvo sa .NET-om

# DEMO



Prikaz arhitekture kroz jednostavni primjer

1010101010001001001010010100100  
110010101010100101010101010101001  
1010101010101010111110110000101010  
1010101010001001001001010101001100  
11001010101010010101010101010101010

# Workflow runtime

- Objekt koji nadgleda izvršavanje aktivnosti, okidanje evenata i praćenje aktivnosti dodatnih servisa koji se povezuju na radnu okolinu
- WF i aplikacija se izvršavaju paralelno – WF koristi aplikaciju kao host
- Do zadnje verzije mogla je biti samo jedna instanca *Workflow Runtime*-a po AppDomain-u
- Svako pozivanje workflow-a stvara jednu *instancu*

# Uvod u drugi demo

- Radimo aplikaciju bez primjene wizarda ( za razumijevanje )
- Gradimo workflow
- Referenciramo WF assembly
- Pišemo kod koji pokreće *WorkflowRuntime*, i upravljamo s njime ( što znači da obrađujemo iznimke koje runtime ponekad aktivira – kao npr. kada *runtime* ode u *Idle* stanje ili kada instanca baci iznimku )
- Dodajemo ručno referencu

# Uvod u drugi demo

- *WorkflowInstance* objekt
  - (Method)AddService
  - (Method>CreateWorkflow
  - (Method)GetWorkflow
  - (Method)StartRuntime
  - (Method)StopRuntime
  - (Property)IsStarted
  - (Property)Name
- Koristimo *Singleton pattern* (samo jedna instanca klase) i *factory pattern*

# Uvod u drugi demo

- *WorkflowInstance* objekt
  - (Event)Started
  - (Event)Stopped
  - (Event)WorkflowCompleted
  - (Event)WorkflowIdle ( moguće akcije su čišćenje iz memorije, spremanje bazu i kasniji povrat u memoriju )
  - (Event)WorkflowTerminated ( Host može terminirati sa *Terminate* naredbom, sa *Terminate* aktivnosti ili ako *WorkflowRuntime* uhvati neobrađenu iznimku )

# Uvod u drugi demo

- Sada smo prošli većinu stvari koje je napravio čarobnjak, ali nam još nedostaje workflow koji bi pokrenuli
- Ako sada pokrenemo demo, on će ostati “visiti”, to jest neće se “okinuti” niti jedan od evenata, a s time se neće ni *waitHandle* nikad aktivirati

101000100100101001010101001100  
10101010010101010101010101010101  
10101010101111101101000010101  
10100010010010101010101001100  
10101010010101010101010101010101  
10101010101111101101000010101

# DEMO



Osnovni prikaz *Workflow Runtime-a*

1010101010001001001010010100100  
110010101010100101010101010101001  
101010101010101011111011000010101  
10101010100010010010010101011001  
1100101010101001010101010101010101

# Workflow instance

- Pokretanje workflow instanci, sa ili bez ulaznih parametara
- Određivanje statusa pokrenutih workflow instanci
- Zaustavljanje workflow instanci
- Određivanje razloga zašto su instance zaustavljenje ili otišle u *idle* stanje

# Workflow instance

- Workflow instance se sastoje od jedne korijenske aktivnosti, koja se naziva *workflow definicijom*
  - Workflow definicija je ono što tražimo runtime da izvede, a instanca je ono što se izvodi – no u praksi se koristi samo termin instanca, i to ćemo nadalje koristiti
- Workflow instance su kao i svaki drugi dio softvera – izvode se od početka do kraja, a mogu ih zaustaviti iznimke tokom izvođenja

# Workflow instance

- Workflow instance se mogu izvoditi dugo vremena ( danima, tjednima.... )
- Workflow instance sudjeluju u transakcijama ( dapače, upravljanje transakcijama je ključni dio upravljanja instancama )
  - Kako upravljamo transakcijama vidjet ćemo u jednom od kasnijih demo-a
- *WorkflowInstance* je WF objekt koji daje pojedinačnim workflow-ima kontekst u kojem se izvodi

# Uvod u treći demo

- Simuliramo dugi ( engf. *long-running* ) task korištenjem sekvensijalnog workflowa koji ima *Delay* aktivnost
- Prikazujemo MessageBox koji nam kaže da je posao (*delay*) započeo i jedan da je završio ( iako to možemo to dobiti iz host-a iz *WorkflowCompleted* eventa, ovo je pravilniji pristup )
- Koristimo kod iz prošlog demo-a da pokrenemo ovaj workflow

# Uvod u treći demo

- Kada dodamo referencu od projekta za treći demo u drugi, neće proći kompilacija
  - Potrebno je ručno dodati još dvije reference
    - System.Workflow.Activities
    - System.Workflow.ComponentModel
- Workflow-i koji prihvataju ulazne parametre
  - Prihvataju ih kao najobičnije public property-e
  - Prenose se upotrebom Dictionary-a
  - Nadopunjujemo postojeći workflow
    - Dodajemo provjeru da li je workflow već aktivan
    - Dinački modificiramo vrijednost activity komponente

# Uvod u treći demo

- Određivanje statusa instance
  - Bazna klasa *Activity* ima property *ExecutionStatusProperty* koji je dio *ActivityExecutionStatus* enumeracije
- Terminiranje instance se vrši pozivanjem metode *Terminate* koja je dio osnovnog objekta aktivnosti
  - Poruka koja se šalje kao parametar metode je tekst exceptiona, ali kako smo mi taj event već prije obradili u delegatu, on će samo ispisati poruku koju mu mi pošaljemo

# DEMO



Workflow instance

1010101010001001001010010100100  
1100101010101001010101010101010100  
101010101010101011111011000010101  
1010101010100010010010010101001100  
1100101010101001010101010101010100

# Uvod u aktivnosti i vrste workflowa

- Aktivnosti su “Lego kocke” WF-a koje služe da se neki poslovni proces podijeli na dijelove
  - Neke aktivnosti služe kao *container-i* za druge aktivnosti
  - Postoji osnovni *container* koji sadržava sve ostale
    - Sekvencijalne aktivnosti
    - *State-based* aktivnosti
- Aktivnosti se izvršavaju ili redoslijedom kojim su definirane ili u slučaju okidanja nekih evenata

# Uvod u aktivnosti i vrste workflow-a

- *Activity* objekt sadržava samo osnovne funkcionalnosti, no pojedini *activity*-i imaju na raspolaganju širok raspon funkcionalnost
- Osnovna metoda je *Execute* kojom počinje izvršavanje poslovne logike pojedine aktivnosti
- Tri osnovne vrste workflow-a:
  - Sekvencijalni ( odvija se definiranim redoslijedom )
  - State machine ( aktivnosti se dešavaju u odnosu na okidanje evenata )
  - Sekvencijski temeljen na pravilima

# Uvod u aktivnosti i vrste workflow-a

- Razlike među vrstama workflow-a
  - Sekvencijalni – workflowi se izvršavaju autonomno sa malo vanjskog upravljanja
  - State machine – ovise o vanjskoj kontroli za svoje izvršavanje
  - Sekvencijalni temeljen na pravilima – postoje pravila za rješavanje složenih problema te ne pripadaju direktno u niti jednu od gornjih kategorija
- Razlika između njih je u potrebama procesa koji želimo modelirati

# Uvod u aktivnosti i vrste workflow-a

- Iako se ovdje neće pokazivati, mogući je dinamički stvarati workflow-e dodavanjem aktivnosti
- U ovome trenutku Visual Studio ne omogućava dodavanje workflow-a direktno u Windows/Web aplikacije, stoga je za njihovu primjenu potrebno napraviti posebni projekt koji će ih saržavati ( kao u slučaju
  - To je, u krajnjoj liniji, i pravilan programerski pristup

# Workflow servisi

- Osim hijerarhije objekata koju smo dosad promatrali dostupni su i dodaci ( *plugin-ovi* ) u obliku servisa koje možemo koristiti zajedno sa stvorenim workflowima
  - Workflow Persistence Service
  - Workflow Queuing Service
  - Workflow Runtime Service
  - Workflow Scheduler Service
  - Workflow Subscription Service
  - Workflow Transaction Service
  - Tracking Service

# Workflow Tracking

- Detaljno ćemo kao primjer ( a i najčešće korišteni servis ) promotriti Workflow Tracking servis
- Kao implementaciju baznih klasa navedenih na prošlom slide-u, u sklopu WF-a dolazi *SQLTrackingService*
- Kao dodatni download dostupni su *ConsoleTrackingService* i *SimpleFileTrackingService*

# Workflow tracking

- Pratimo workflow proces dodavanjem tracking servisa na workflow runtime
- Kako se okidaju praćeni eventi, Wf stvara i upravlja tracking zapise ( *tracking records* )
- Upiti nad tim podacima se mogu pregledavati upotrebom *WorkflowMonitor* alata – alat koji dolazi sa WF-om kao primjer, zajedno sa kompletним izvornim kodom
- Svi podaci se nalaze u SQL Server bazi te se mogu, naravno, promatrati i odvojeno

# Workflow tracking

- Postoje tri glavne kategorije evenata koji se prate:
  - Activity eventi
  - Workflow eventi
  - Korisnički eventi
- Iako je, naravno, moguće napraviti vlastiti traking modul, u slijedećem demo-u ćemo prikazati ugrađeni modul za praćenje koji podatke sprema u SQL Server 2005/2008 bazu

# Uvod u četvrti demo

- Prvo stvaramo bazu u koju ćemo spremati podatke
  - U našem primjeru to je TrackingDb
- Zatim treba izvršiti skripte za stvaranje potrebnih objekata u bazi
  - One se nalaze u <%WinDir%>/Microsoft.NET/v3.0/Windows Workflow Foundation/SQL/EN/[Tracking\_Shema.sql i Tracking\_Logic.sql]

# Uvod u četvrti demo

- Projektu je potrebno dodati *System.Configuration* referencu – koristimo je za pristupanje ConnectionStringu koji je spremljen u konfiguracijskoj datoteci
- Dodajemo konfiguracijsku datoteku ( App.config )
- Dodajemo slijedeće reference:
  - *System.Workflow.Runtime.Tracking*;
  - *System.Configuration*;

# Uvod u četvrti demo

- Dodajemo kod koji dodaje tracking u runtime
- Glavni podaci se nalaze u *ActivityInstance* tablici
- Iako postoje objekti kojima je aplikacijski moguće dohvatiti logirane podatke, puno je jednostavnije dohvatiti ih direktno iz baze
  - Ovaj pristup ( sa ugrađenim objektima ) se uglavnom koristi ukoliko želimo prikazati ili pratiti neki određeni događaj
- Izrađujemo korisničke podatke koje želimo pratiti
  - Ti podaci se nalaze u *UserEvent* tablici

# Uvod u četvrti demo

- Moguće je izrađivati takozvane *tracking profile* u kojima možemo definirati koji se dio podataka logira
  - Neće biti prikazano u demo-u radi količine potrebnog koda i vremenskog ograničenja predavanja
  - Konceptualno, radi se o XML datoteci u kojoj se definira što a što neće biti uključeno u praćenje

# DEMO



Workflow servisi i tracking

1010101010001001001010010100100  
1100101010101001010101010101010100  
10101010101010101111101101000010101  
1010101010100010010010010101001100  
110010101010100101010101010101010100

# Učitavanje i iščitavanje instanci iz memorije

- Većina poslovnih procesa su dugotrajni, a sigurno nisu mjereni mikrosekundama na način na koji rade strojevi
- No, poslovni procesi koji zahtijevaju ljudsku intervenciju spadaju u kategoriju tzv. *long-running* procesa
- Mora postojati mehanizam koji će te procese, dok čekaju na prelazak na sljedeći korak, iščitati iz memorije i spremiti u neko spremište podataka dok se ne stvore uvjeti za njihovo daljnje izvršavanje

# Učitavanje i iščitavanje instanci iz memorije

- Kao što smo imali servis za praćenje ( *tracking* ) tako WF dolazi sa dediciranim servisom za spremanje instanci na SQL Server 2005/2008 bazu
  - Puni naziv je *SqIWorkflowPersistanceService* te služi serijalizaciji workflowa u bazu
- To se dešava kada WF engine “odluči” da neki dugotrajni workflowi bespotrebno zauzimaju memoriju, no postoji i niz evenata koji nam omogućava programsku kontrolu nad ovim procesom

# Učitavanje i iščitavanje instanci iz memorije

- lako naizgled jednostavno, ako malo detaljnije analiziramo ovaj problem počnemo dolaziti do značajnog komplikiranja situacije
  - Najčešća situacija je da imamo jednu bazu za spremanje instanci, no što ako se one izvode na više strojeva paralelno, ili ako čak i na istom, ali pod više odvojenih procesa?
  - Postoji ugrađen mehanizam koji osigurava da se instanca vrati u stanje u kojem je bila prije spremanja

# Učitavanje i iščitavanje instanci iz memorije

- Instanca se mora spremiti neovisno o tome da li je bila blokirana ( čekala na neku akciju ), njen trenutni status ( *executing, idle,...* ), trenutne podatke koje sadržava, vlasnika procesa koji ju je pokrenuo...
  - Ukoliko se ti i još mnogi drugi podaci pravilno i sigurno ne mogu spremiti, instanca se neće moći povratiti u početno stanje
  - Ovaj proces je ključan za razumijevanje rada orkestracija u Biztalk Serveru ( hidracija/dehidracija ) , a i ovdje predstavlja pozadinu na kojoj počiva WF

# Učitavanje i isčitavanje instanci iz memorije

- Nakon što instanci “dodamo” servis za spremanje u bazu, postoje 3 metode u objektu *WorkflowInstance* kojima kontroliramo ovaj proces:
  - Load
  - Unload ( blokira trenutni thread dok se operacija ne izvrši )
  - TryUnload ( ne blokira trenutni thread )

# Uvod u peti demo

- Ponovno postavljamo bazu za spremanje instanci kao i u prošlom demo-u
  - Nalaze se u <%WinDir%>/Microsoft.NET/v3.0/Windows Workflow Foundation/SQL/EN/ [SqlPersistanceService\_Shema/ SqlPersistanceService\_Logic].sql
- Iako naizgled jednostavno, ako malo detaljnije analiziramo ovaj problem počnemo dolaziti do značajnog komplikiranja situacije

# Uvod u peti demo

- Gradimo jednostavnu Windows aplikaciju koja ima nekoliko kontrola kojima ćemo kontrolirati učitavanje i spremanje instanci
- Forsirati ćemo *long-running* workflow, ali ne preko *Delay* aktivnosti ( koja ima svoj posebni način spremanja stanja ), nego ćemo glavnom threadu reći da se zaustavi na 10 sekundi što nam daje dovoljno vremena da aktiviramo kontrole u Windows aplikaciji

# Uvod u peti demo

- Nakon što sve prikažemo kako je implementirano u kodu, pokrećemo na dva načina:
  - Prvo pokrećemo “Start Workflow” i cekamo 10 sekundi prije nego što se vrati u početno stanje
  - Zatim pokrećemo ponovno i u vremenu od tih 10 sekundi kliknemo na “Unload workflow” te gledamo u bazu u tablicu “InstanceState”
  - Klikom na “Load Workflow” učitavamo ga iz baze i vraćamo u izvršavanje

# Uvod u peti demo

- Također, moguć je “automatski” sustav spremanja workflow-a koji se automatski aktivira korištenjem *Delay* aktivnosti (naravno, ako je aktivan SqlPersistance servis)
- Dodajemo još jedan workflow sa *Delay* aktivnosti i dorađujemo aplikaciju
  - Dodajemo namespace *System.Collections.Specialized*

# DEMO



Učitavanje i spremanje instanci

1010101010001001001010010100100  
110010101010100101010101010101001  
101010101010101011111011000010101  
10101010100010010010010101011001  
1100101010101001010101010101010101

# Objavljivanje workflowa kao web servisa

- Prvo što treba napomenuti kod web servisa ( ali o tome nećemo pričati jer nije tema predavanje ) je da oni sa sobom donose sigurnosne i konfiguracijske probleme
- No ovdje je problem druge prirode
  - Kako se *WorkflowRuntime* izvršava paralelno sa aplikacijom (asinkrono), što pruža prednosti u lokalnom načinu rada, ovdje postoji vrlo realna mogućnost da dođe ASP.NET zahtjev koji pokrene workflow, i nakon toga se stranica počne učitavati i prikaže se prije nego što workflow završi svoje izvođenje

# Objavljivanje workflowa kao web servisa

- Radi navedene situacije potrebno je “natjerati” WF da se izvodi paralelno ( u istom threadu ) sa ASP.NET procesom što predstavlja problem sa *long-running* workflowima
  - Ovo su dva osnovna problema rada WF-a u ASP.NET aplikacija ( bilo web aplikacijama ili web servisima ) i rješavanje ovog problema predstavlja osnovu ovog demo-a

# Uvod u šesti demo

- Razvijamo Console aplikaciju koja poziva deployani web servis
- Radi količine stvari koje treba objasniti detaljnija objašnjenja će ići kroz kod
  - Jedino što ovdje treba reći da je rješenje ovih problema *persistiance*, to jest spremanje stanja workflowa i učitavanje u idućem ciklusu pozivanja ( bilo ASP.NET aplikacije, bilo web servisa

101000100100101001010101001100  
10101010010101010101010101010101  
10101010101111101101000010101  
10100010010010101010101001100  
10101010010101010101010101010101  
10101010101111101101000010101

# DEMO



Objavljivanje workflowa kao web servisa

1010101010001001001010010100100  
110010101010100101010101010101001  
1010101010101010111110110000101010  
1010101010001001001001010101100100  
11001010101010010101010101010101010

# Što još treba pogledati?

- WF nudi veliki broj predefiniranih aktivnosti, no prolaziti na ovome predavanju kroz specifičnosti svake aktivnosti je potpuno besmisleno
  - Na kraju predavanja su dani prijedlozi literature koju sam ja koristio u proučavanju WF platforme i koja opširno pokriva sve specifičnosti dostupnih aktivnosti kao i izradu korisničkih

# Što još treba pogledati?

- *State-based workflows*
  - Iako je bilo moguće ubaciti demo sa primjerom ovakvog sustava, teorija koja leži iza toga i potpuno objašnjenje bi zahtijevalo previše vremena, no ovo područje se svakako isplati proučiti
- Transakcije i workflow-i
  - Kombinacijom transakcija ( i povezanih aktivnosti *TransactionScope*, *Compensate*, *CompensatableTransactionScope* ) i workflowa ( i sekvensijalnih i stanja ) moguće je simulirati najrazličitije procese koje zahtijevaju veliku dozu sigurnosti izvršenja – simulacije ATM-a, procesa web narudžbi, ...

# Što još treba pogledati?

- Korelacije i workflow komunikacije
  - Što se dešava kada jedna aplikacija poziva više instanci istog workflowa, te na koji način se osigurava integritet podataka ( u WF-u se to naziva *korelacija* )
- Pozivanje vanjskih web servisa direktno iz workflowa
  - Što se dešava kada workflow želi pozvati vanjski servis, kako se radi autentifikacija, koji su problemi i prednosti ovakvog pristupa
- Deklarativne workflow-e ( XAML )
- I još puno toga... ☺

# Literatura i kontakt

- Literatura:
  - MS Press: Microsoft Windows WF Step By Step
  - Wrox: Professional Windows Workflow Foundation
  - Apress: Pro WF in .NET 3.0
  - Apress: Foundations Of WF - An Introduction To Windows Workflow Foundation
- Kontakti:
  - [josipsaban@gmail.com](mailto:josipsaban@gmail.com)
  - <http://www.linkedin.com/in/jsaban>

# Q&A



Hvala na pažnji i puno sreće

**Microsoft®**  
*Your potential. Our passion.™*

10101010100010010010100100100100  
1100101010101001010101010101010100  
101010101010101011111011000010101  
101010101010001001001001010100100  
1100101010101001010101010101010100