Projektuppgift: Automatiserad RSS-läsare

Introduktion

Bolaget AutomateEverything tror att det finns en marknad för automatiserade RSS-läsare för privatpersoner. ”Tid = Pengar” tänker de, ”och det här systemet kommer våra kunder spara tid på!”. Bolaget tänker också att det är dyrt att anställa en konsult för att utföra jobbet.

- Vi anställer några studenter istället! De är billiga i drift, menar chefen på ett möte med utvecklingsteamet, och alla håller med.

Chefen planerar att projektet ska genomföras av grupper på 3 studenter, och de kommer få drygt två veckor på sig att genomföra uppgiften. Lansering av produkten sker 25 november så projektet måste vara inlämnat, med källkod och rapport, **senast tisdag 6 november kl. 23:59** för att all funktionalitet ska hinna verifieras. Återkoppling och omdöme kommer att meddelas senast 23 november innan lanseringen.

Rapport

En rapport skall bifogas med färdigt projekt, fokus är dock på projektet och koden. Rapporten ska skrivas gemensamt av gruppen, dvs. en rapport per projekt, och bör inte vara längre än 3 sidor. Det finns inga krav på formalia i rapporten – det räcker med att helt enkelt svara på frågorna nedan – men kom ihåg att referera korrekt om ni använder annat material i rapporten.

Vi vill att rapporten innehåller:

- Erfarenheter och reflektioner från projektet
    - Vad har varit bra eller dåligt?
    - Vad kunde ni ha gjort bättre?
- Kodreflektioner:
    - Vad skulle ni vilja gjort annorlunda om ni haft mer tid?
    - Vilka genvägar har ni varit tvungna att ta?

Projektupplägg

I ert projekt får ni välja om ni vill använda Windows Forms eller WPF. På föreläsningarna har vi gått igenom Windows Forms men inte WPF (och WPF tas inte heller upp praktiskt i kursboken). WPF är således valfritt för de som vill utmana sig själva. För de som vill använda WPF finns en grundstruktur för projektet på blackboard, väljer ni att arbeta med Windows Forms skapar ni projektet själva så som gåtts igenom på föreläsningarna.

Krav

Uppgiften är alltså att skapa en automatiserad RSS-läsare, som används för att prenumerera på podcasts. Det ska göras genom en desktop-applikation för Windows, där en användare kan mata in en URL till ett RSS-flöde för en podcast-prenumeration, namnge flödet samt ange hur ofta data ska hämtas (uppdateringsintervall).

Detta flöde ska sedan hämtas in och visas i en lista, som också ska lagras som XML på den egna datorn. Under tiden programmet körs ska det enligt ett angett uppdateringsintervall se om det kommit någon ny podcast i RSS-flödet, och i så fall spara ner detta.

Er applikation behöver inte kunna spela upp nedladdade podcasts.

För att utveckla systemet ska ni använda Visual Studio och programmeringsspråket C#.

Arkitektur

Projektet ska struktureras i en tre-lagersarkitektur. Boken tar upp detta kortfattat på sidan 362 och i föreläsningen för KT13 ges exempel på det. För mer inspiration kan ni läsa på om designmönstret ”repository pattern”. Genom att använda repository pattern kan man minska den duplicering av kod som annars ofta uppstår. Det ger även ytterligare ett abstraktionslager mellan affärslogik och datalogik och gör att samma funktioner kan användas för att hämta olika typer av data.

Ni ska också arbeta enligt Single Responsibility Principle (SRP) när ni designar er kod. Detta innebär att varje klass/metod ska ha en uppgift, inte fler.

Tekniska krav

Chefen vill testa era färdigheter i C# och kräver att ni använder följande:

- Virtual
    - Virtual är ett nyckelord som anger att en metod eller property i en klass kan implementeras av dess subklass(er).
- Interface
    - Ett interface representerar ett kontrakt -- det definierar en eller flera metoder eller egenskaper som en implementerande klass måste ha.
- Async
    - Att göra asynkrona anrop är viktigt när anropen kan ta lång tid, till exempel att läsa in en Feed eller ladda ner en podcast. Tips: Kolla på klasserna SyndicationFeed, XMLReader och Task för att lösa detta.
- Linq
    - Linq är ett sätt att hantera Collections. Man kan filterera/sortera/räkna beroende på olika villkor.

Method overloading ska också användas, vilket innebär att man kan definiera metoder med samma namn men med olika parametrar. Vid metodanropet kallas den metod som har en överensstämmande parameterlista.

Vidare får det färdiga projektet inte ge några byggfel eller -varningar vid körning och all felhantering ska hanteras med Exceptions. Det ska finnas minst en egen (kom ihåg SRP!) klass som sköter all validering av det data användaren matar in, och den klassen ska kasta passande Exceptions om inmatningen är felaktig (tänk på att ni kan skapa egna Exceptions).

Funktionella krav

Projektet kommer främst att hantera tre typer av entiteter: feeds, avsnitt och kategorier. Feeds och kategorier ska vara persistenta: de ska sparas på användarens dator och laddas in igen när man startar applikationen på nytt.

En feed är en kanal där webbinnehåll publiceras där användare via en länk kan prenumerera på innehållet. I vårt fall ska applikationen kunna hantera RSS-feeds i form av XML, där innehållet är podcast-avsnitt. Varje feed hör till en kategori samt har ett antal avsnitt knutna till sig. Följande funktionalitet ska vara möjlig:

Podcast-feeds

- Lägg till en ny podcast-feed som:
    - Har en URL (dvs. den adressen som feeden är tillgänglig via).
    - Hör till en kategori
    - Har ett uppdateringsintervall (dvs hur ofta applikationen ska kolla på länken för att se om det publicerats några nya podcasts)
    - Har ett antal avsnitt.
- Ändra en feeds URL, kategori samt uppdateringsintervall
- Ta bort en feed med all tillhörande data
- Visa alla avsnitt i en feed

När programmet körs ska applikationen kolla efter nya avsnitt i feeden enligt det tidsintervall som användaren angett. Hittas nya så ska dessa läggas till i feed:en.

Podcastavsnitt

Avsnitt-objekten ska visas i en lista av valfritt utseende. Det ska också gå att visa mer information om varje avsnitt, till exempel genom att man klickar på podcast-objektet i en lista för att se en detaljvy av innehållet.

Kategorier

Kategorier används för att sortera podcasts-feeds efter ämne, till exempel hälsa, musik, programmering. Det ska vara möjligt att:

- Lägga till en kategori
- Ändra en kategori
- Ta bort en kategori
- Visa alla feeds som hör till en viss kategori

Övrigt

Ni ska skriva er kod i enlighet med principen Clean Code, vilket innebär att koden ska vara tydlig och lätt att förstå, utan att några kommentarer behövs. Det handlar inte om att bara sluta kommentera koden, utan att se till så namnsättningen är tydlig och metoder och klasser är korta, koncisa och har ett tydligt syfte.

Det räcker alltså inte med att det fungerar utan ni måste se till så det inte finns onödig redundans, alltför komplexa metoder, klasser som har fler ansvarsområden osv i er kod. Läs på om Clean Code, SRP och tre-lagersarkitektur (gärna även om Repository-mönstret) innan ni börjar koda!

Instruktioner för inlämning

Hela er solution ska komprimeras till en zip/rar-fil och skickas in tillsammans med rapporten senast 6 november kl. 23:59. Skicka projektet via meddelande på Blackboard till **Kalle Räisänen**. Sent inskickade projekt kommer inte bedömas under ordinarie examination utan ni får då vänta till om-examination av projektet! Som slutbetyg på projektet ges betyg godkänt (G) eller underkänt (U). Rest kan ges en gång på projektet, om resten inte är godkänd får studenten göra om projektet nästa gång kursen ges.

Tänk på att:

- Snygga till koden innan ni skickar in projektet! Vi vill inte se filer som inte hör till projektet, funktioner som inte används och liknande.
- Bra kod är oftast enkel kod.
- Det finns inga krav på den grafiska designen av programmet, försök dock att skapa ett tydligt och lättanvänt GUI till er applikation. Men kom ihåg: Hellre snygg kod än snyggt gränssnitt.