Översikt

- Arbeta med Unit-test.

.easec

Lektion 1: Arbeta med Unit tester

- Unit-tester.
- Tester.
- Tre faser.
- Automatiskt testning.
- Test Driven Development.
- Principer för Test Driven Development.
- Loosely Coupled Application.
- Tightly Coupled Application.
- Skriva unit-test för komponenter i MVC.
- Använda unit-test.

.easec

Lektion 1: Arbeta med Unit tester (forts.)

- Specificera rätt mål.
- Mocking Framework.

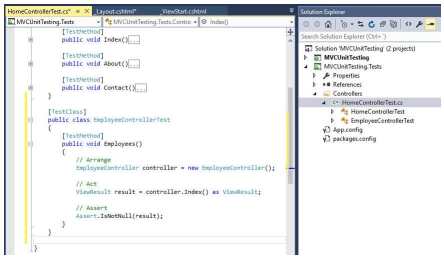
Unit-tester

- Lektionen fokuserar på unit-tester, denna typ av tester utförs under utveckligen.
- Andra typer av tester, t ex tester för integration eller acceptans kan bara utföras när ett antal komponenter finns på plats.
- I MVC webb applikation kan unit-test instansiera klasser för model eller controllers, kalla på dess metoder och händelser.
- Du kan skriva unit-test som skapar ett objekt för produkt, testar att metoden köp, returnerar ett chart objekt.

Tester

- *Unit-test*, kontrollerar små aspekter för funktionen. Exempelvis: att viss metod returnerar rätt värde. Genom att arbeta med flera unit-tester kan du försäkra dig om att dessa täcker hela din applikation.
- *Integrationstest*, kontrollerar hur två eller flera komponenter fungerar tillsammans, kan även användas för att testa hela siten.
- *Acceptanstest*, fokus på funktion eller tekniska krav som måste fungerar för att beställaren skall acceptera.

Tre faser



.eavec

Tre faser (forts.)

- *Arrange:*
  - Testet skapas instans av klass som skall testas, tilldelar även egenskaper som krävs och skapar alla de objekt som behövs.
  - Bara egenskaper och objekt som behövs för testet skapas.
- *Act:*
  - Testet har den funktion som skall kontrolleras.
  - Oftast en enda procedur och sparar resultat i en variabel.

.eavec

Tre faser (forts.)

- *Assert:*
  - Test kontrollerar resultat med förväntat resultat.
  - Matchas resultatet har programkod passerat testet, om inte kommer fel att visas.
- Efter skapandet av test, kommer test att köras automatiskt när du startar med debugging.

.eavec

Unit-test hjälper till att fixa buggar

- Eftersom unit-test testar en liten och specifik del av programkoden, är det lätt att diagnosiera problem om ett test skulle fallera.
- Vanligtvis använder vi testet på en enda klass och isolerar denna från de andra klasserna där det är möjligt.
- Om inblandning av annan klass behövs, skall ett minimerat antal klasser skapas i fasen Arrange.
- Att strukturera på detta sätt, ger möjlighet att fixa fel snabbt pga att källor för fel är litet.

.easec

Unit-test hjälper till att fixa buggar (forts.)

- Unit-test skall kontrollera programkod som du skriver och inte infrastruktur som eventuell programkod är beroende av.
- Exempelvis:
  - Unit-test skall köras utan att ansluta till databas eller webbtjänst eftersom nätverksproblem eller att tjänst är nere kommer att orsaka fel.
  - Genom att arbeta på detta sätt, kan du särskilja på buggar från programkod, vilket kan fixas med omskrivning av programkod, från buggar från infrastruktur, som du kan fixa till genom att ändra hårdvara, omkonfiguration av webbserver eller anslutningssträng.

.easec

Automatiskt testning

- Det är viktigt att förstå, att efter definition av unit-test, kan tester köras som kvickt och lätt under hela livscykeln för projektet.
- Visual Studio kör om testet automatiskt när ett projekt startas med debugging. Testet kan även köras manuellt.
- Detta är viktigt, för ny programkod kan orsaka buggar var som helst i utvecklingsfasen.

.easec

Automatiskt testing -exempel

- Ett exempelprojekt har tre stycken faser, ett unit-test definierat för fas 1 hjälper dig att observera problem som kan uppstå i fas 2 och fas 3, som annars kanske hade passerat ouppmärksammat.
- Fas 1, enkel controller som innehåller två händelser, Index() och Details(), bägge returnerar views. Ett unit-test skrivs, Test_Index_Action, testet testar om händelsen Index fungerar med en integer parameter. När händelsen anropas med integer, skall view visas det antalet produkter som definieras av integer. Testet kommer att passeras.

.eosec

Automatiskt testing –exempel (forts.)

- Fas 2, utvecklare modifierar händelsen så den returnerar partial view, för att följa namnkonvention så ändras namnet för händelse till _index. Testet kommer att fallera, eftersom namnet har ändrats. Utvecklare modifierar testet så testet kallar på den namnändrade händelsen istället och test kommer att passeras.
- Fas 3, utvecklar modifierar händelsen Index så att en LINQ-fråga används, för att uppfylla ett nytt krav. Men utvecklare kommer att skriva fel och händelsen returnerar inga poster. Testet kommer att fallera.

.eosec

Automatiskt testing –exempel (forts.)

- Unit-test hjälper utvecklare med att fokusera på lösningar för att gå igenom testet och på sätt ser till att ny funktionalitet inte orsakar fel i programkod som är skriven tidigare.
- Unit-test kan användas med vilken projektmetodik som helst, för att uppmärsamma buggar och förbättrar programkvalité.

.eosec

Test Driven Development

- I Test Driven Development (TDD) är unit-test en central funktion.
- TDD är oftast beskriven som en separat utvecklingsmetodik, en del betraktar det som en viktig del i metodiken Extreme Programming.

.eorec

Test Driven Development (forts.)

- I TDD arbetar utvecklare med korta cyklar för att implementera krav för en funktion:
  - *Skriv test,* utvecklare börjar med att skriva ett test, kräver att utvecklaren förstår kraven, dessa fås från user stories och user cases. Därefter körs testet, eftersom ingen programkod är skriven kommer testet att fallera.
  - *Passera testet,* skriver snabbt enkel programkod som passerar testet.
  - *Refactor,* rensar upp i koden, duplicerad kod tas bort, förbättrad läsbar. Kontinuerligt körs testet om.

.eorec

Principer för Test Driven Development

- *Skriva test innan programkod,* test skapas som första steg, innan programkod skrivs. Testet måste fallera första gången. Testet är egentligen en specifikation för funktionalitet som du bygger. Genom att skriva testet först, försäkra du dig om att du har förstått kraven.
- *Gå framåt i små steg,* genom att bryta upp en stor applikation i små bitar av funktionalitet ökar produktiviteten. Utvecklar kan lösa funktion snabbt och få förståelse omgivningen de körs i.
- *Skriv bara tillräckligt med programkod för att passera testet,* även om det är frestande att skriva in ytterligare funktionalitet, så skriv bara programkod för att testet skall gå igenom.

.eorec

Principer för Test Driven Development (forts.)

- Utvecklare kan referera till testerna som exempel på hur klasser och metoder skall användas.
- Testerna kan även användas för att demonstrera dessa.

.easec

Loosely Coupled Application

- Konceptet att skriva applikation så att alla klasser kan fungerar oberoende av varandrar, kallas för Loosley Coupled Application.
- I MVC har vi tre delar:
  - Model, klasser och logik.
  - View, gränssnitt som baseras på information i Model.
  - Controller, svarar på förfrågningar och väljer View.
- De tre delar i MVC kan förändras individuellt utan att påverka de andra delarna.
- Lätt att skriva unit-test, då det är lätt att ersätta full funktionalitet i en klass, med en förenklad variant.

.easec

Tightly Coupled Application

- ASP.NET WebForms kallas för Tightly Coupled Application.
- Varje .aspx sida är beroende av underliggande programkod.
- Är svårt att testa.

.easec

Skriva unit-test för komponenter i MVC

- Models är enkla att testa, pga dessa är separata klasser. Dessa klasser kan instanieras och konfigureras under fasen arrange.
- Controllers är en enkel klass som du kan testa, men kan vara komplex om du använder information, speciellt om denna information kommer från en databas.
- Unit-test skall inte blanda in infrastruktur, skall bara testa programkod.
- Eventuell information som behövs, måste finnas i minnet. Detta kallas för test double.

.easec

Använda unit-test

- Lägg till ett nytt projekt till din befintliga programkod, baserat på mallen Unit Test Project.
- Referenser måste läggas till så programkod får tillgång till klasser i ditt MVC-projekt.
- I testprojektet markeras klasser med [TestClass], metoder med [TestMethod].
- Unit-test returnera vanligtvis void, men kalla på metod i Assetklass, t ex Assert.AreEqual för att kontrollera resultatet.

.easec

Specificera rätt mål

- När test double används för i unit-test för att testa controllers, är det viktigt att du har rätt mål, i två scenarions: unit-test och utanför testet.
- Unit-test används test double, i din programkod används Entity Framework.
- För att specificera rätt förråd att användas för test, använd controller constructor.
- Du kan även använda Invension of Control container, ett ramverk som kan läggs till din applikation och som instansierar klasser.

.easec

Mocking Framework

- Under fasen Arrange, måste testinformation skapas.
- Du kan göra detta manuellt, eller genom att arbeta med ramverk för detta.
- Mocking, ger möjlighet att isolera programkod under text, eventuella beroende förutsätts att dessa fungerar.

Övning Skapa Unit-test

Repetitionsfrågor