|  |
| --- |
| Laborationsanvisning |

Gissa det hemliga talet

Steg 1, laborationsuppgift 1

|  |
| --- |
| *Författare:* Mats Loock  *Kurs:* ASP.NET MVC  *Kurskod:*1DV409 |

**Upphovsrätt för detta verk**

Detta verk är framtaget i anslutning till kursen ASP.NET MVC (1DV409) vid Linnéuniversitetet.

**Du får använda detta verk så här:**

Allt innehåll i detta verk av Mats Loock, förutom Linnéuniversitetets logotyp, symbol och kopparstick, är licensierad under:

|  |  |
| --- | --- |
|  | Creative Commons Erkännande-IckeKommersiell-DelaLika 2.5 Sverige licens. <http://creativecommons.org/licenses/by-nc-sa/2.5/se/> |

**Det betyder att du i icke-kommersiella syften får:**

* kopiera hela eller delar av innehållet
* sprida hela eller delar av innehållet
* visa hela eller delar av innehållet offentligt och digitalt
* konvertera innehållet till annat format
* du får även göra om innehållet

Om du förändrar innehållet så ta inte med Linnéuniversitetets logotyp, symbol och/eller kopparstick i din nya version!

Vid all användning måste du ange källan: ”Linnéuniversitetet – ASP.NET MVC” och en länk till <https://coursepress.lnu.se/kurs/aspnet-mvc> och till Creative Common-licensen här ovan.

**Innehåll**

[Problem 5](#_Toc370382239)

[Modell 5](#_Toc370382240)

[Den uppräkningsbara typen Outcome 6](#_Toc370382241)

[Strukturen GuessedNumber 6](#_Toc370382242)

[Klassen SecretNumber 6](#_Toc370382243)

[Vymodell 7](#_Toc370382244)

[Controller 8](#_Toc370382245)

[Vy 8](#_Toc370382246)

[Körexempel 9](#_Toc370382247)

[Krav 13](#_Toc370382248)

# Problem

Skriv webbapplikation med hjälp av ASP.NET MVC och C# där användaren ska ha sju försök på sig att gissa ett hemligt tal i det slutna intervallet mellan 1 och 100.

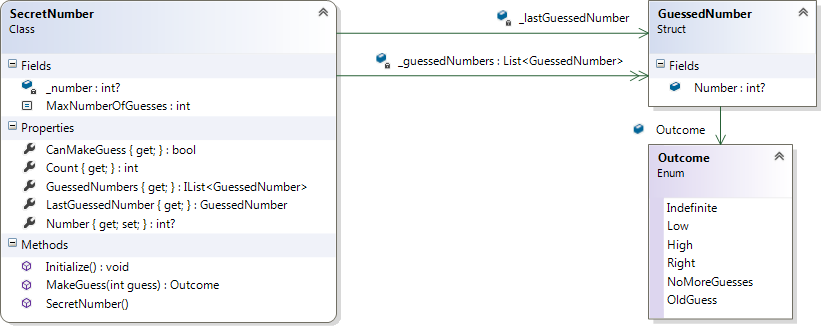


Figur 1.

När en användare gjort en gissning ska resultatet av gissningen presenteras, det vill säga om gissningen var för låg, för hög eller rätt. Har användaren gissat rätt, eller förbrukat samtliga gissningar, ska det inte gå att göra fler gissningar innan ett nytt hemligt tal slumpats.

## Modell

I klassdiagrammet nedan ser du ett förslag på hur du kan utforma typerna som ska utgöra modellen i applikationen. Du behöver inte utforma dina typer på samma sätt utan kan skapa en egen om du så önskar så länge du ser till att alla krav uppfylls.



Figur 2. Förslag på modelltyper.

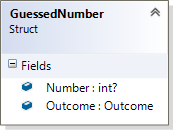
### Den uppräkningsbara typen Outcome

Typen Outcome används för att representera utfallet av en gissning.

|  |  |
| --- | --- |
| Figur 3. | * Innan en gissning gjort är utfallet obestämt (Indefinite). * En gissning på ett tal mindre än det hemliga talet är för låg (Low). * En gissning på ett tal större än det hemliga talet är för hög (High). * En gissning på ett tal lika med det hemliga talet är för rätt (Right). * Har sju gissningar redan gjorts och fler försök att gissa görs saknas fler möjligheter att gissa (NoMoreGuesses). * En gissning på ett tal överstämmande med en tidigare gissning är en gammal gissning (OldGuess). |

### Strukturen GuessedNumber

Instanser av strukturen GuessedNumber används för att lagra information om genomförda gissningar.



Figur 4.

#### Fältet Number

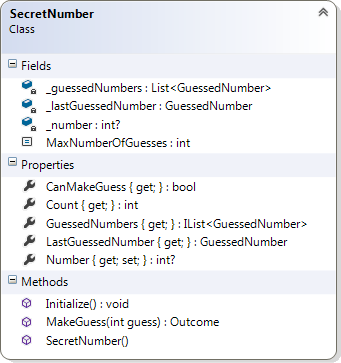
Publikt fält av typen int? som innehåller en gissnings värde.

#### Fältet Outcome

Publikt fält av typen Outcome som innehåller utfallet av en gissning.

### Klassen SecretNumber

Klassen måste implementeras så den som minst innehåller medlemmarna enligt klassdiagrammet i Figur 5 och har den funktionalitet som beskrivs för respektive medlem för att klara samtliga tester.



Figur 5. Klassdiagram över SecretNumber. Observera att set-blocken för egenskapen Number ska vara privat.

#### Fältet \_guessedNumbers

Privat fält, av typen List<GuessedNumber>, som innehåller godkända gissningar där utfallet är High, Low eller Right sedan det hemliga talet slumpats fram. Kapslas in av egenskapen GuessedNumbers.

#### Fältet \_lastGuessedNumber

Privat fält, av typen GuessedNumber, som innehåller den senaste gissningen med tillhörande utfall. Kapslas in av ”read-only”-egenskapen LastGuessedNumber.

#### Fältet \_number

Privat fält av typen int? som innehåller det hemliga talet. Kapslas in av egenskapen Number.

#### Konstanten MaxNumberOfGuesses

Publik konstant, av typen int, med värdet 7 som definierar hur många gissningar en användare har på sig att gissa rätt.

#### Egenskapen CanMakeGuess

Publik ”read only”-egenskap, av typen bool, vars värde ger om användaren kan gissa eller inte.

Så länge användaren kan göra en gissning ska egenskapen ge värdet true. Egenskapen ska ge värdet false då en användare förbrukat sju gissningar eller lyckats gissa rätt.

#### Egenskapen Count

Publik ”read-only”-egenskap, av typen int, vars värde ger antalet gjorda gissningar sedan det hemliga talet slumpades fram.

#### Egenskapen GuessedNumbers

”Read-only”-egenskap, av typen IList<GuessedNumber> som ger en ”read-only”-referens till det privata fältet \_guessedNumbers. OBS! För att undvika en *”privacy leak”* får under inga omständigheter en referens till det privata fältet \_guessedNumbers returneras av egenskapen. (Ett tips är att använda metoden AsReadOnly() som löser alla(?) problem.)

#### Egenskapen LastGuessedNumber

”Read-only”-egenskap, av typen GuessedNumber, vars värde ger utfallet av den senaste gissningen, oavsett utfallet.

#### Egenskapen Number

Egenskap, av typen int?, där get är publik och set är privat, som kapslar in det hemliga talet. Så länge som det går att gissa ska egenskapen ge värdet null. Går det inte att gissa ska egenskapen ge värdet den privata egenskapen \_number har.

#### Metoden Initialize

Publik metod som initierar klassens fält och egenskaper.

* List-objektet \_guessedNumbers refererar till ska återanvändas och måste därför rensas på gjorda gissningar.
* Number ska tilldelas ett slumpat heltal i det slutna intervallet mellan 1 och 100.

#### Metoden MakeGuess

Publik metod som anropas för att göra en gissning av det hemliga talet. Beroende om det gissade talets värde är för högt, lågt eller överensstämmer med det hemliga talet ska lämpligt värde av typen Outcome returneras. En gissning på ett tidigare gissat tal ska inte räknas.

Om värdet parametern guess har inte är i det slutna intervallet mellan 1 och 100 ska metoden kasta ett undantag av typen ArgumentOutRangeException.

#### Konstruktorn

Konstruktorn har till uppgift att se till att ett SecretNumber-objekt är korrekt initierat. Det innebär att fält och egenskaper har blivit tilldelade lämpliga värden, vilket enklast görs genom att låta konstruktorn anropa metoden Initialize().

Konstruktorn ska även ansvara för att instansiera List-objektet som håller ordning på gjorda gissningar.

## Vymodell

Beroende på hur du väljer att utforma modellklassen kan det vara lämpligt att skapa vymodellklasser som hanterar datat vyer presenterar. Exempelvis låter du valideringen av inmatat värde hanteras av en vymodellklass istället för att låta en modellklass implementera detta.

## Controller

Tänk på att det hemliga talet, eller snarare objektet som utgör modellen, ska kommas ihåg mellan postningar. I en controller har du tillgång till egenskapen Session…

Undersök vad egenskapen Session har att erbjuda till hjälp för att kontrollera om en session gått ut eller inte.

## Vy

Hur många vyer behöver du? Fler än en? Använder du flera vyer och behöver visa samma data i flera kan det vara en god idé att placera gemensam kod i en eller flera ”*partial views*” så att du bara har koden på ett ställe (bryt inte mot DRY-principen, *Don’t Repeat Yourself*).

Inte bara ”*partial views*” kan vara av intresse utan även ”*helper methods*” av olika slag.

# Körexempel

Figurerna nedan är ett förslag på hur gränssnittet kan utformas. Det står dig fritt att utforma det som du finner lämpligt.



Figur 6. Användaren står i begrepp att göra en sjätte gissning.



Figur 7. Användaren gissar på ett tal användaren redan gissat på.



Figur 8. Användaren har gissat rätt hemligt tal.



Figur 9. Användaren misslyckas med att gissa det hemliga talet.



Figur 10. Användaren gissar på ett tal större än 100.



Figur 11. Användaren gissar då textfältet är tomt.



Figur . Användaren väntat för länge med att gissa varför session gick ut.

# Krav

Du får fritt utforma formuläret, där inmatningen av en gissning görs. Följande krav måste uppfyllas:

|  |  |
| --- | --- |
|  | Användaren har sju (7) försök på sig att gissa rätt tal vilket ska lagras i en namngiven konstant i modellen. |
|  | Det ska bara gå att genomföra en gissning då användaren gissar på ett heltal mellan 1 och 100. Validering av en gissning innan den skickas till servern måste göras i största möjligaste mån. Självklart ser du till att validera på servern också. |
|  | Användaren ska informeras om den senaste gissningen var för hög eller för låg jämfört med det hemliga talet. |
|  | Användaren ska informeras om en gissning är korrekt och hur många gissningar som krävdes samt därefter kunna välja att få gissa på ett nytt hemligt tal. |
|  | Antalet gissningar som gjorts sedan det hemliga talet slumpats fram ska kunna kommas åt med en egenskap, exempelvis Count. |
|  | Gissar användaren på ett tal som redan han/hon gissat på ska det inte räknas som en gissning utan användaren ska informeras om att han/hon redan gissat på talet. |
|  | Användaren ska kunna se en historik över tidigare gissningar och om de var för höga eller för låga jämfört med det hemliga talet. |
|  | Klienten ska inte på något sätt kunna komma åt det hemliga talet, som slumpas fram. Det hemliga talet får under inga omständigheter lämna servern (om det inte är så att användaren förbrukat alla gissningar, se punkt 12). |
|  | Alla hantering av det hemliga talet placerar du i en klass som får utgöra modellen. Klassen ansvara för att på lämpligt sätt informera om användaren gissat rätt tal, gissat för lågt, gissat för högt, redan gissat på talet och om användaren förbrukat alla gissningar. |
|  | En gissning görs lämpligen genom att en metod i modellklassen anropas. Metoden kan exempelvis returnera, och/eller sätta en egenskap till, lämpligt värde (enligt punkten ovan) beroende på resultatet av gissningen. Mycket lämpligt är att använda en uppräkningsbar typ (enum) som returvärde.  Metoden ska returnera lämpligt värde, förslagsvis av typen Outcome, beroende på resultatet av gissningen.   1. Är det gissade talet inte i det slutna intervallet mellan 1 och 100 ska ett undantag av typen ArgumentOutOfRangeException kastas. 2. Användaren har sju (7) försök på sig att gissa rätt tal. Det ska inte gå att göra fler gissningar utan att ett nytt hemligt tal slumpas fram. Görs ytterligare försök utöver de stipulerade sju ska värdet Outcome.NoMoreGuesses returneras. 3. Är det gissade värdet samma som en tidigare gissnings värde ska inte gissningen räknas och värdet Outcome.OldGuess returneras. 4. Är det gissade värdet lägre än det hemliga talet ska värdet Outcome.Low returneras. 5. Är det gissade värdet högre än det hemliga talet ska värdet Outcome.High returneras. 6. Är det gissade värdet samma som det hemliga talet ska och värdet Outcome.Right returneras. |
|  | Resultatet av den senaste gissningen ska kunna kommas åt via en egenskap, exempelvis LastGussedNumber. |
|  | Går det att genomföra en gissning ska det hemliga talet inte kunna kommas åt publikt. |
|  | Går det inte att genomföra en gissning ska det hemliga talet finnas publikt tillgängligt via en egenskap, exempelvis Number. |
|  | Några ”*privacy leaks*” får under inga omständigheter förekomma i modellklassen. |
|  | Tar användaren så lång tid på sig att skicka in en gissning så att sessionen går ut ska ett felmeddelande visas. |