Min Kursrapport

DV1610
Objektorienterade webbteknologier
(mvc)

Johan Kristoffersson
joki20@student.bth.se

v1.0.0
April 03, 2021

Läsperiod 4
Våren 2021

Institutionen för datavetenskap (DIDA)
Blekinge Tekniska Högskola (BTH)
Sverige

Innehåll

Kapitel 1

Objektorientering

1.1 Bakgrund och fördelar

Jag är bekant med objektorientad programmering genom kursen i objektorienterad Python tidigare under våren. Det har underlättat förståelsen för klasser i detta kursmoment, trots annat programmeringsspråk. Alexander Petkovs artikel på hemsidan freeCodeCamp tar upp viktiga principer kopplat till objektorienterad programmering [3]:

1. Inkapsling - möjlighet att skapa unika objekt som användaren kan interagera med via publika metoder. 2. Abstraktion - det underliggande arbetet som objektet utför via de publika metoderna hålls dolda för användaren 3. Arv - möjlighet att skapa subklasser som ärver bas- och föräldraklasser.

Säg du jobbar med ett stort projekt med olika moduler. För att undvika att variabler krockar i globala scopet kan inkapsling göra att var och en kan skapa metoder genom inkapsling där publika metoder kan kommunicera med de andra klasserna. Gällande abstraktion så kan vi ta användare som går in på SMHI eller gör en beställning från en webbshop. Då är det viktigaste med ett tydligt gränssnitt där fokus ligger på att användaren kan slutföra en uppgift (beställa, boka, se lokala vädret etc). Arv i sin tur är viktigt för att kunna skapa exempelvis nya användare på en sida där basklassen innehåller grundläggande info, och där ärvda subklasser arv gör att användarens attribut/egenskaper kan specificeras.

1.2 Klasser och objekt i PHP

Vid skapandet av klasser är det viktigt att utgå från en grund. I den officiella manualen för PHP [4] tas viktiga kodstandarder och grunder upp, likaså på Tutorials Points artikel om objektorienterad PHP [2]

Klasser innehåller variabler (kallas medlemsvariabler) och funktioner. Klassen fungerar som en mall som kan instansiera/skapa objekt utifrån sin mall genom nyckelordet **new** och klassens namn. Objekten tilldelas klassens variabler (kallas då medlemsvariabler/attribut/properties) medans objektens metoder (medlemsfunktioner) delas mellan alla objekt. Om du har definierat en konstruktorfunktion kan du skicka med argument när objekt skapas som då tilldelas attributvärden som du valt ut. Objekt ska komma åt sina egna metoder och attribut används inom objektet syntaxen **this**, medan utanför objektet används objektnamnet som referens. Om du har definierat en klasskonstant nås den inom klassen med **self::konstant** och utanför med klass::konstant Attribut och metoder kan vara antingen **privata** (åtkomliga endast inom klassen), **publika** (åtkomliga även utanför klassen) eller **protected** (för ärvda klasser, åtkomliga inom basklassen och den ärvda klassen). De metoder som du vill att användare ska kunna komma sätter du som publika, medan medlemsvariabler bör vara privata.

1.3 Tankar om kodbasen

Fördelen som jag ser med **Namespace** är att jag kan skapa ett projekt under ett visst namespace, och sedan ett annat projekt under ett annat namespace och inte oroa mig för att skriva över variabler. Användningen av **use** för att definiera om referensen sina klasser kan vara fördelaktig om man vill komprimera sin kod, så länge det inte blir på bekostnad av tydligheten. Gällande autoloader (som inkluderar filer) och composer känns det som områden man inte helt förstår hur de fungerar i bakgrunden men som har ett värde för att exempelvis kunna hantera objekt i sessioner. Strukturen i övrigt med view och Router påminner en del om tidigare kurser som exempelvis databas-kursen där vi använde express och .ejs-filer. Däremot känns Router-filen lite rörig med alla elseif-satser och det blir svårt att få en bra överblick över alla sina views.

1.4 Tankar om game 21

Game21-klassen anropar publika metoder från övriga klasser. Jag använde funktionen **isset** som kontrollerade vilket formulär som postats. I början kollas om knappens name är startgenom **isset och POST[start"]** som nollställer al-

la resultat och visar formulärknappar för 1 eller 2 tärningar. Genom multipla elseif-satser som kollade vilken knapp som trycktes i en bestämd ordning från start till slut, och genom att spara spelarens poäng och antal rundor i sessionsvariabler, så kunde poängen föras vidare när användaren trycker på POST. Koden ligger i klassen Game21 och anropas genom publika metoden game21() i vyn.

Såhär i efterhand borde jag kanske använt mig av switch case-satser för att skapa en tydligare kod-struktur. Jag kunde då också använt mig mer av en sessionsvariabel som definierar vilket läge sessionen befinner sig i. Har försökt kommentera koden noggrannt för att jag och andra ska kunna gå tillbaka till den och förstå vad som händer stegvis. Det underlättade att börja skriva koden i vyn, för att sedan flytta över den till en klass. Då kunde jag fokusera mer på kodens struktur i början när jag löste uppgiften. För denna uppgift använde jag inte graphic-metoden vilket blivit mer tilltalande visuellt. Anledningen till detta var att jag inte på ett lätt sätt kunde få ut och använda siffrorna för tärningsslagen.

1.5 Tankar om PHP The Right Way

Dokumentet PHP The Right Way"[1] är en communitybaserad online-handbok inom PHP. De kanske viktigaste och mest intressanta områdena i dokumentationen är de om sessioner, databaser och inloggning med säkra lösenord. Hur man gör för att bygga upp säkradatabassystem med användare och lösen känns högst relevant att behärska som programmerare. Väl ute på arbetsmarknaden kommer det att ställas krav på att vi ska behörska dessa delar och PHP är fortfarande ett vanligt språk som hängt med länge och som många arbetsplatser fortfarande sitter med. Att kunna skapa nya användare utifrån klasser och lägga in dem i databasen känns som en bra fortsättning.

1.6 Lärtillfälle från kursmoment 1

Jag tar med mig vikten av att låta logisk kod tillhöra klasserna för att skapa renare vyer. Det är också viktigt att endast göra metoder som användaren ska komma åt publika. Det är viktigt att då inte ändra metodernas namn om koden ska ändras, utan att istället ändra på klassens kod.

Kapitel 2

Controller

Här skriver du din redovisningstext för detta kursmoment.

Kapitel 3

Enhetstestning

Här skriver du din redovisningstext för detta kursmoment.

Kapitel 4

Ramverk

Här skriver du din redovisningstext för detta kursmoment.

Kapitel 5

Autentisering

Här skriver du din redovisningstext för detta kursmoment.

Kapitel 6

To be defined

Här skriver du din redovisningstext för detta kursmoment.

Kapitel 7

Projekt & Examination

Här skriver du din redovisningstext för detta avslutande kursmoment.

7.1 Projektet

Här skriver du din redovisningstext rörande projektet.

7.2 Avslutningsvis

Här skriver du de avslutande orden om kursen.

Litteratur

[1] Phil Sturgeon et al Josh Lockhart. "PHP The Right Way". I: (2021). URL: https://phptherightway.com/. (besökt: 04.03.2021).

[2] "Object Oriented Programming in PHP". I: (2021). Dnr 2019:00860, Promemoria, Beskrivande statistik. URL: https://www.tutorialspoint.com/php/php_object_oriented.htm%22. (besökt: 04.03.2021).

[3] Alexander Petkov. "How to explain object-oriented programming concepts to a 6-year-old". I: (2018). URL: https://www.freecodecamp.org/news/object-oriented-programming-concepts-21bb035f7260/. (besökt: 04.03.2021).

[4] "PHP: The Basics - Manual". I: (2021). URL: https://www.php.net/manual/en/language.oop5.basic.php. (besökt: 04.03.2021).