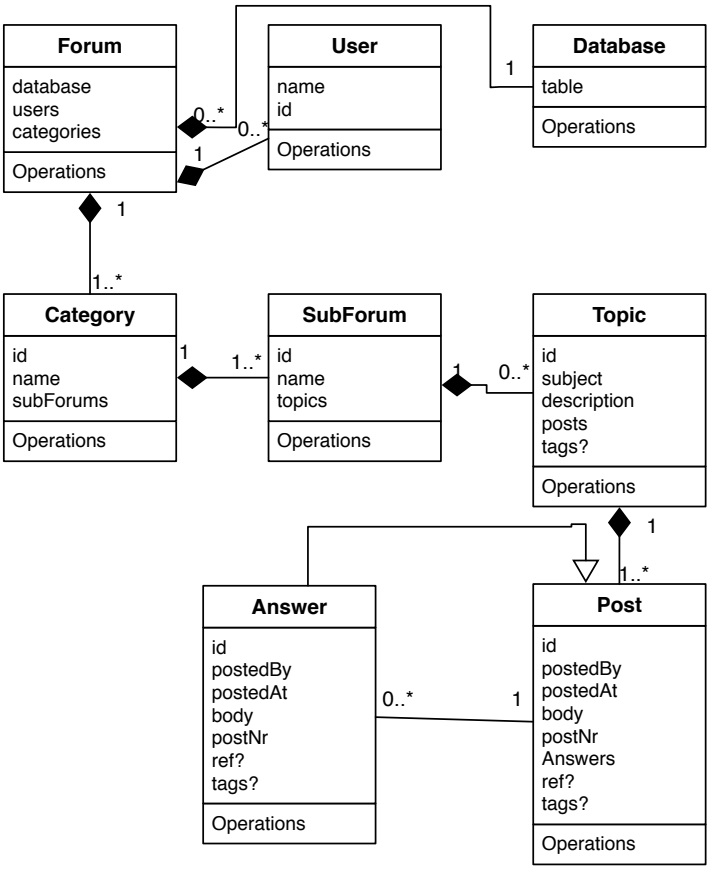
Design

2012-03-22

Innehåll

Versioner

Version	Datum	Ändringar	Utförda av
0.3	2012-03-22	Ändringar efter kommentarer och utförligare text	gusah849
0.2	2012-02-27	Uppdaterad kodstandard.	nicol271
0.1	2012-02-22	Första versionen.	gusah849



Figur 1: UML-klassdiagram över systemet

1 Designstruktur

Övergripande UML-diagram över forumet kan ses i figur 1

2 Undersystem

2.1 LiveDB

Databasen som kunden Visiarc bidragit med. Den hanteras genom ett API som följer med databasen. Denna databasen används för att spara ner informationen i forumet. Så som trådar och poster.

2.2 Enyo

Javascript ramverket Enyo 2.0 kommer att användas för att bygga upp forumappen. Det är ett ramverk för att enkelt kunna skapa objekt och olika vyer för forumet. Man bygger upp kontroller med hjälp utav något som enyo kallar för "kind". Det kan liknas med en klass.

2.3 Databasstrukturen

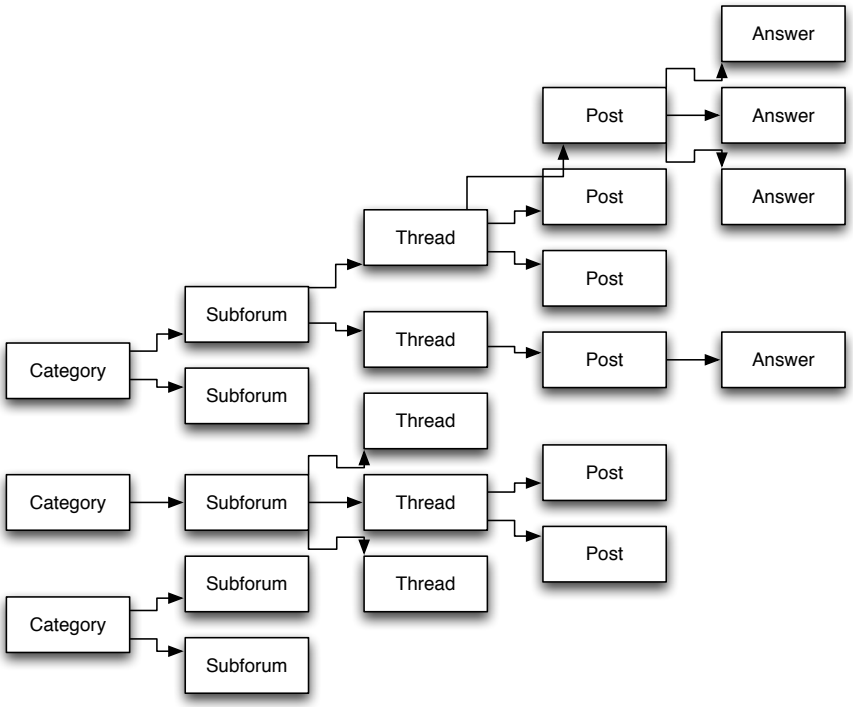
Databasen kommer vara uppbyggd så att första noden kommer vara en kategori, en kategori har sedan ett underforum. Underforum har trådar som sedan kan ha poster i sig. Varje post kan ha ett svar. Ett exempel på hur databasen kan se ut ses i figur 2.

2.4 Uppbyggnad

Tänk att varje klass i uml-diagramet (figur 1) är en enyo kind som sedan kommer bestå utav componenter utav de andra klasserna. Själva forumappen kommer vara den första klassen Forum som sedan kommer bestå utav en komponent som innehåller subforums. Komponenten är även htmltaggen som kommer ritas ut på hemsidan.

Ungefär hur det kan se ut i kod:

```
1  enyo.kind({
2    name: "Forum",
3    kind: enyo.Control,
4    tag: div,
5
6    components: [
7      {tag: "ul", name: "categoriesContainer"}
8    ]
9  });
10
11 enyo.kind({
12   name: "category",
13   kind: enyo.Control,
14   tag: "li",
15
16   components: [
17     {tag: "span", name: "catTitle" }
18     {tag: "ul", name: "threadContainer" }
19   ]
20
21   published: {
22     title: "",
23     description: ""
```



Figur 2: Exempel på hur databasen kan se ut

```
24    },
25
26    create: function(){
27      this.inherited(arguments);
28      this.titleChanged();
29      this.descriptionChanged();
30    },
31
32    titleChanged: function(){
33      this.$.catTitle.setContent(this.title);
34    },
35
36    descriptionChanged: function(){
37
38    }
39
40  });
```

Databasobjektet kommer dock inte att baseras på en enyo kind utan det kommer vara ett vanligt javascript objekt.

2.5 Språkuppbyggnad

Språket kommer att fungera så att vi kommer ha en klass för språk och sedan spara statiska variabler till den klassen för de olika språksträngarna.

```
1  theLang = function(){};
2  theLang.helloWorld = "Hej varlden";
```

3 Designmönster

När vi skriver koden för projektet ska vi använda oss utav bland annat följande designmönster

3.1 Model-View-Controller

3.1.1 Överblick

Detta designmönster separerar vyn och datan. Den använder sedan kontrollen för att få över datan till vyn.

3.1.2 Struktur

Det går till så att de saker som berör vyn kommer att läggas i separata filer och det som hanterar datan i egna filer. Vi kommer sedan att ha funktioner

som skickar datan till vyn där den presenteras.

3.1.3 Beteende

Detta gör att ifall man ändrar till exempel representationen utav datan behöver man inte ändra om i vyn för att kunna presentera datan.

4 Kodstandard

4.1 JavaScript

4.1.1 Namn

Använd camelcase till variabelnamn och funktionsnamn. All kod på engelska.

```
1 function functionName(firstVar, secondVar) {
2     return;
3 }
```

4.1.2 Indentering

Ha alltid en tab när du går ner en nivå för att få lättöverskådlig kod. Börja alltid måsvingepar på ny rad.

```
1 function foo() {
2     return;
3 }
```

4.1.3 Kommentarer

Kommentera koden väl, men endast där det är nödvändigt. En enkel kommentar om vad funktionen gör eller vad variabeln har för betydelse. Kommentarer är alltid på engelska.

```
1 var calls; // number of calls to the printNumbers function
2
3 // Prints 0 to max
4 function printNumbers(max) {
5     ++calls;
6     for(int i=0;i<=max;++i) {
7         console.log(i);
8     }
9 }
```

4.1.4 Operatorer

Ha alltid ett mellanslag på varje sida om en operator vid till exempel strängsammanslagning eller aritmetiska operationer.

```
a =  1 +  2;
foo = "hej" + "san";
b += 7;
```

4.1.5 Switch-satser

En switch-sats ska ha all a sina case på en ny rad, break efter en case-del ska ha samma indentering som koden.

```
switch(foo) {
    case 1:
        foo = bar;
        break;
    case 2:
        bar = foo;
        break;
    default:
        bar = foobar;
        break;
}
```

4.2 HTML

4.2.1 DOCTYPE

Doctypen ska alltid vara för HTML 5 och den ser ut som följande:

```
<!DOCTYPE html>
```

En standardsida i HTML är uppbyggd enligt följande mall:

```
<!DOCTYPE html>
<html>
<head>
<title>Title of the document</title>
<!-- inkludera scripts och stilfiler -->
</head>

<body>

<!-- Kroppen av sidan -->

</body>

</html>
```