

2017-05-22

Namn

## **Examensarbete för Gymnasieingenjör**

Inriktning: Informationsteknik

Utgång: Mjukvarudesign

Titel

# Agil utveckling av anpassningsbara funktioner i webb-applikationer, resursbedömning och kompromisser

Namn

Huddinge

Maj, 2017

Handledare

Carina Envall

# Rapportskrivning för T4

## Abstract

Developers need critical thinking in order to be more efficient. Usually, there are several solutions for one same problem which means that it's better not to take the first solution but search and make choice for a more effective way to resolve a problem.

This report describes an example of how an agile developer uses freedom to choose a less complicated solution for a problem, how the choice was made and what consequences it led to. The goal with this report is give an example of how such choice is made in a real situation so that other developers may use similar methods in suitable situations.

The describing uses a product as an example – a web application with user adjustable UI elements in a web application.

The development was done in Visual Studio 2015, ASP.NET, HTML, JavaScript, CSS.

## Sammanfattning

Utvecklare behöver ha kritiskt tänkande för att vara effektiva. Ofta finns det flera lösningar för ett problem, därför behöver man inte välja den första lösningen, utan hitta och välja mer effektiva sätt att lösa ett problem.

Rapporten beskriver ett exempel på hur en agil utvecklare använder sin frihet för att försöka uppnå produktägarens krav utan att skapa överkomplicerade lösningar, det beskrivs hur valet gjordes och vilka konsekvenser det har lett till. Syftet med rapporten är att ge ett exempel på hur utvecklare väljer lösning i en konkret situation så att andra utvecklare kan använda liknande metoder vid andra tillfällen.

Beskrivning sker med en produkt som utvecklingsexempel – en webbapplikation med gränssnittselement som användare kan anpassa efter sina behov.

Utvecklingen skett i Visual Studio 2015, SQL, EF, ASP.NET, HTML, JavaScript, CSS.

## Innehållsförteckning

<b>Abstract</b> .....	2
<b>Sammanfattning</b> .....	2
<b>Inledning</b> .....	4
Bakgrund .....	4
Syfte .....	4
Frågeställningar .....	5
Material och metoder.....	5
<b>Resultat</b> .....	6
Utgångspunkten.....	6
Vidareutveckling – kravanalys och Lösningsval .....	8
Implementering av lösningen: .....	11
Förbättringar till den implementerade lösningen: .....	16
<b>Diskussion/slutsats</b> .....	19
Referenser.....	22
Bilagor .....	23

2017-05-22

Namn

## Inledning

### Bakgrund

Team Mountain Development i T4 gymnasieingenjörer i Östra gymnasiet fick ett uppdrag under praktiken – att skapa en applikation som hjälper att beskriva sporthändelser mycket snabbare än man kan göra det normalt i textform.

Applikationen borde vara tillgänglig till så många användare som möjligt, vilket uppnås enklast genom att skapa applikationen med webbteknologierna, och team Mountain Development har lärt sig att arbeta med Microsoft MVC i Visual Studio 2015 utvecklingsmiljö, därför har applikationen utvecklats med användning av dessa. Mountain Development använde **agilt** arbetssätt under utvecklingen. Bedömning och lösningsval gjordes av utvecklarna.

En av kärnfunktionerna i appen är **frasknappar** – funktionella knappar som användare kan lägga till in i gränssnittet för att sedan klicka på dem och få fram ord, fraser eller fullständiga meningar som beskriver händelser. I den här rapporten beskrivs frasknapparna och deras utveckling som ett exempel på **anpassningsbart gränssnitt** samt hur utvecklare valt att implementera dem och varför just på det sättet.

Se bilaga 1 för en bild på applikationen.

### Syfte

Beskriva utvecklingsprocessen för en funktion i en specifik webb-applikation som användaren kan anpassa efter sina önskemål. Förklara hur utvecklare analyserade krav och hittade kompromiss enligt agilt arbetssätt och vidareutvecklade anpassningsmöjligheter med avseende på effektivitet.

### Frågeställningar

- Hur hittar man olika lösningar till ett problem?
- Hur väljer man en passande lösning för sin situation?
- Hur implementeras lösningen?
- Hur håller man gränssnittet minimalistiskt och funktionell även när nya funktioner läggs till?
- Vilken nytta får användaren av förbättringarna?

### Material och metoder

VS2015, SQL, Entity Framework, ASP.NET MVC, AJAX, HTML, CSS, JavaScript

Gymnasieingenjörer i T4 i Östra Gymnasiet 2016/2017 lärde sig att använda de flesta av ovanstående programmeringsspråk/verktyg/ramverk.

Visual Studio 2015 är utveckelmiljö från Microsoft, huvudspråket är C#.

SQL är ett språk för databashantering.

Entity Framework är ett ramverk som underlättar datahantering

ASP.NET är ett ramverk med inloggningsfunktioner.

2017-05-22

Namn

## Resultat

*OBS: Rapporten innehåller tekniska fördjupningar, dessa markeras med kursiv text eller består av kodexempel. Läsare utan programmeringsbakgrund kan hoppa över dessa.*

### Utgångspunkten

Team Mountain Development från Östra gymnasiets T4 klass 2016-2017 gick på praktik i 10 veckor. Uppdraget var att skapa en applikation med funktioner som låter användaren beskriva händelser under en pågående rugbymatch. Det är väldigt svårt att beskriva matchhändelser i realtid i textform utan stödfunktioner. Frasknappar är en av stödfunktionerna som ingår i programmet.

Teamet fick skapa en egen uppfattning om hur frasknapparna kunde implementeras; medlemmarna bestämde sig för att låta användaren sätta knapparna i separata grupper för att kunna sortera olika sorters frasknappar. Applikationens användbarhet berodde på användarens möjlighet att lägga ut så många knappar som möjligt för att kunna beskriva nästan alla sorts händelser som sker under rugby matcher. Det var självklart att gruppssortering behövdes för att hålla knappar särade från varandra för att göra det enklare för användaren att hitta rätt knapp.

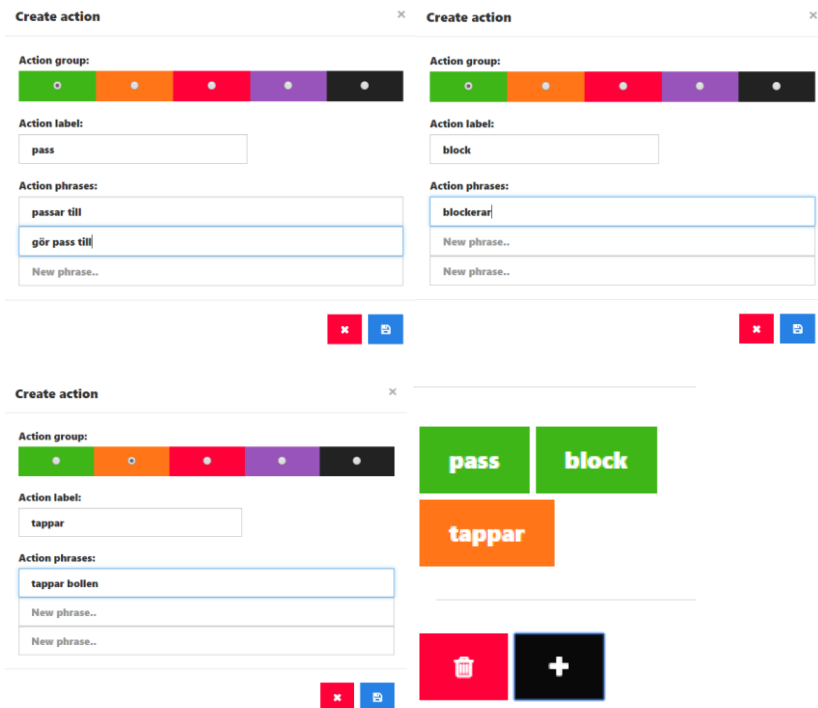
Vid slutet av praktiktiden har Mountain Development skapad en helt fungerande prototyp av applikationen; frasknapparna ingick och fungerade väldigt smidigt vid provkörning när produktägaren testade applikationen.

I sista versionen fick man placera knappar i 5 olika grupper. Knapparna visades i den ordningen som de lades till (kronologisk sortering). Här nedan finns det ett exempel på tre frasknappar man skapar. Den fjärde bilden visar hur knapparna har lagts.

Observera att gränssnitt i applikationen är på engelska; frasknappar kallas för "actions" (ungefär "händelser") både i gränssnitt och i koden.

2017-05-22

Namn



The image shows three instances of the 'Create action' dialog box. Each dialog has a title bar with a close button (X). Below the title bar is an 'Action group' section with a row of five colored circles (green, orange, red, purple, black). The 'Action label' is a text input field. The 'Action phrases' section contains a list of phrases with a 'New phrase...' button at the bottom.

- First dialog:** Action label is 'pass'. Action phrases are 'passar till' and 'gör pass till'.
- Second dialog:** Action label is 'block'. Action phrases are 'blockerar' and 'New phrase...'.
- Third dialog:** Action label is 'tappar'. Action phrases are 'tappar bollen' and 'New phrase...'.

Below the third dialog, there is a visual representation of the action buttons: a green button labeled 'pass', a green button labeled 'block', an orange button labeled 'tappar', and a black button with a white plus sign. There are also red and blue buttons with icons (a trash can and a plus sign) below the 'tappar' button.

Här nedan finns det en del av CSHTML filen som placerar ut frasknapparna i respektive grupp – koden är delvis C#, delvis HTML. Separata grupper har olika färger och börjar från ny rad. Det uppnås genom att köra if-satser i vyn som är ansvarig för frasknappar. Det används en vymodell (datamall för information som skickas fram till användarenhet) som innehåller PlayerActionsList (lista på frasknappar). Just den här koden kan placera ut fraser som tillhör grupper 1 och 2, men det är ändå samma kodstruktur för de resterande grupperna.

*OBS: Det används ramverk "Bootstrap" och "Font Awesome" som hjälper att bygga webbsidor i HTML. Deras funktionalitet beskrivs inte inom rapporten eftersom de är mindre relevanta.*

```
<div class="row">
  @foreach (RugbyTweetApp.Entities.Action item in Model.PlayerActionsList)
  {
    if (item.Group == 1)
    {
      <a class="btn btn-success btn-action-input action pull-left action-left"
        onclick="setActionToPlayer('@item.Phrases', '@item.Text')">@item.Text</a>
      <a class="btn btn-danger btn-action-remove action pull-left action-left"
        onclick="removeActionByID(@item.Id)"
        style="display: none;">@item.Text</a>
    }
  }
</div>
<div class="row">
  @foreach (RugbyTweetApp.Entities.Action item in Model.PlayerActionsList)
  {
    if (item.Group == 2)
    {
      <a class="btn btn-warning btn-action-input action pull-left action-left"
        onclick="setActionToPlayer('@item.Phrases', '@item.Text')">@item.Text</a>
      <a class="btn btn-danger btn-action-remove action pull-left action-left"
        onclick="removeActionByID(@item.Id)"
        style="display: none;">
        @item.Text
      </a>
    }
  }
</div>
```

*Eftersom objekt (frasknappar) har nyckelegenskap "ID" och de kopplas till vymodelen med hjälp av metod "ToList()" då visas frasknapparna i den ordningen som de lades till in i databasen.*

Lösningen är relativt enkel både att implementera och att använda, dessutom fungerar den logiskt – nyaste knappar hamnar sist in i sina grupper. Men om man vill ha möjligheten att sortera knappar på egen hand då blir det problematiskt: man får ta bort vissa knappar och lägga till dem på nytt. Om användaren vill lägga till en knapp i början av en grupp som redan har ett stort antal frasknappar då blir det väldigt frustrerande eftersom man får fylla gruppen från början– man behöver mer flexibilitet. Det är klart att förbättringar krävs.

### Vidareutveckling – kravanalys och lösningsval

Produktägaren ville ha möjlighet att friplacera knappar i gränssnittet på ett friare sätt (utvecklarteamet fick förslaget senare under utvecklingstiden), men det saknades tid för att implementera funktionen fullständigt – det fanns andra, mer



2017-05-22

Namn

viktiga funktioner som behövde uppmärksamhet just då, därför har frasknapparna inte förbättrats enligt produktägarens önskemål just under praktiktiden.

Produktägaren ville ha **friplacering** av knappar, ungefär som på skissen nedan:



Syftet med sådan lösning är att ge användaren möjlighet att skapa ett eget flexibelt knappsgränssnitt – man klickar vad en spelare gör (tacklar) och sedan klickar på en adjektivknapp som beskriver hur spelaren gjorde det. Nyttan av den här lösningen verkar vara självklar – relevanta knappar är grupperade nära varandra och det borde gå snabbare att beskriva händelser, men utan att provköra den här lösningen finns det inget sätt att utvärdera om det är verkligen märkbart bättre än gamla lösningen. Dessutom finns det en brist med den här lösningen: användare kommer behöva skapa fler kopior av adjektivknappar ifall de vill utnyttja friplacering på det föreslagna sättet. Därmed kommer det kräva mer tid att lägga ut knappar på gränssnitt och planera genom layout. Det ger mer frihet för användaren men också gör det något svårare att använda applikationen.

På tekniska sidan av frågan finns det ännu fler utmaningar. Implementering av friplacering kräver att hela frasknappsgränssnittet görs om från början. Nytt knappgränssnitt kräver någon sorters rutsystem för att kunna placera knappar på valfria platser samt med flera, komplicerade stödfunktioner (återkommer till dem senare i rapporten). Rutsystem och liknande lösningar är väldigt svåra att få fungera rätt på webbsidor eftersom olika enheter har olika upplösningar och grafiska element som knappar kommer skalas på olika sätt – något som webbutvecklare får problem med även om man använder statisk kod. Med dynamiska element som frasknappar blir det ännu svårare att producera felfritt gränssnitt.

Preliminär bedömning av friplacering visade att lösningen kräver alltför mycket tid för att implementeras på det sättet som produktägaren vill. Utveckling av en så komplicerad lösning står i strid med den tionde principen av "Agile Manifesto": "Simplicity - the art of maximizing the amount of work not done--is essential." <sup>1</sup>

Det sökes en annan lösning för problemet – användaren vill kunna sortera frasknappar på fler sätt än bara att placera dem i grupper men lösningen ska inte kräva ombyggnad av existerande funktioner. Grupper avgör redan vertikal positionering för knappar. Det som saknas är möjlighet att sätta knappar i valfri ordning inom grupperna, men det här kan man implementera som en utbyggnad på existerande funktioner – man behöver inte kasta undan gamla arbetet och börja om, utan man fortsätter med det som finns och fungerar. Det stämmer överens med den tionde principen – enkelhet är viktigast.

Denna lösning med knappsortering inom grupper kan implementeras på följande sätt:

1. Frasknappsobjekt behöver egenskap som beskriver deras position i respektive grupp. Den får heta "GroupPlace".
2. Vid skapelse, ska knappobjekt få genererad "GroupPlace" variabel som är unikt och sist i gruppen så att nya knappar hamnar sist i sin grupp.
3. Programmet ska sortera knapparna enligt "GroupPlace" så att de kommer enligt den ordningen.
4. Det behövs UI (gränssnitt) element som låter flytta på frasknapparna.
5. Det behövs funktion som tar emot information om vilken knapp som bör flyttas och åt vilket håll. Samma funktion ska hitta närmaste "granne" till knappen, antingen till vänster eller höger beroende på vilket håll knappen

---

<sup>1</sup> "Principles behind the Agile Manifesto"

2017-05-22

Namn

flyttas mot. Den här funktionen ska byta ut GroupPlace på de knapparna som behöver byta plats.

### Implementering av lösningen:

*Man börjar med att lägga till en ny variabel in i datadefinitionen i SQL tabell som sparar frasknappar. Variabeln kallas för "GroupPlace".*

*När man lägger till en ny frasknapp då körs koden nedan: den hämtar in data som användaren matade in (knappnamn, vilka fraser knappen innehåller samt knappens grupp) och sedan genererar variabeln "highestPlaceInGroup" vilken är högsta numret för knappar i gruppen. Saknas det knappar i gruppen då blir variabeln lika med noll. Egenskapen "GroupPlace" för nya knappobjekten blir ett steg högre än "highestPlaceInGroup".*

```
List<Entities.Action> actionsInSameGroup = actions.Where(action => action.Group == group).ToList();
actionsInSameGroup = actionsInSameGroup.OrderByDescending(a => a.GroupPlace).ToList();

int highestPlaceInGroup = 0;
if (actionsInSameGroup.Count>0)
{
    highestPlaceInGroup = (int)actionsInSameGroup[0].GroupPlace;
}

Entities.Action receivedActionTemplate = new Entities.Action
{
    Text = newActionTemplateLabel,
    Group = group, ConfigID = rugbyConfig.Id,
    Phrases = phrases,
    GroupPlace = highestPlaceInGroup + 1
};

rugbyTwitterDB.Actions.Add(receivedActionTemplate);
rugbyTwitterDB.SaveChanges();

return RedirectToAction("ShowPlayerActions");
```

*Funktionen som förbereder vymodellen (knapplistan som ska visas upp) behöver sortera knapparna med avseende på "GroupPlace". Det görs enligt koden nedan. Med den här koden kommer knapparna visas i ordning efter "GroupPlace".*

2017-05-22

Namn

*OBS att det finns mindre relevant kod i funktionen, den är med i exemplet för att visa helheten.*

```
public ActionResult ShowPlayerActions()
{
    //hämta in användarID i ASP.NET
    string CurrentDotNetID = User.Identity.GetUserId();

    //hämta in användarobjekt i separata databasen
    User rugbyUser = rugbyTwitterDB.Users.First(i => i.DotNetID == CurrentDotNetID);

    //hitta Config som hör till användaren
    Config rugbyConfig = rugbyTwitterDB.Configs.First(config => config.UserID == rugbyUser.Id);

    //hitta frasknappar (Actions) som hör till användarens Config
    List<Entities.Action> Actions = rugbyTwitterDB.Actions.Where(action => action.ConfigID == rugbyConfig.Id).ToList();
    //frasknapparna (Actions) är nu sorterade efter ID

    //sortera frasknappar (Actions) efter GroupPlace
    Actions = Actions.OrderBy(a => a.GroupPlace).ToList();

    //skapa vymodell
    PlayerActionsViewModel actionsVM = new PlayerActionsViewModel();

    //placera sorterade frasknappar in i vymodellen
    actionsVM.PlayerActionsList = Actions;

    //skicka PartialView med vymodellen
    return PartialView(actionsVM);
}
```

#### Kodexempel A

*Nästa steg är att tillämpa UI (användargränssnitt) element som används för att flytta på frasknappar. De nya elementen kallas för pilknapparna. Se bilden nedan:*

```
if (item.Group == 1)
{
    <a class="action-mover btn pull-left btn-default btn-sm btn-sm-fh btn-action-input action action-left"
    onclick="moveActionJS('@item.Id', 'back')">
        <i class="fa fa-caret-left fa-2x fa-fw"></i>
    </a>

    <a class="btn btn-success btn-action-input action pull-left action-left"
    onclick="setActionToPlayer('@item.Phrases', '@item.Text')">@item.Text</a>
    <a class="btn btn-danger btn-action-remove action pull-left action-left"
    onclick="removeActionByID('@item.Id)"
    style="display: none;">@item.Text</a>

    <a class="action-mover btn pull-left btn-default btn-sm btn-sm-fh btn-action-input action action-left"
    onclick="moveActionJS('@item.Id', 'forward')">
        <i class="fa fa-caret-right fa-2x fa-fw"></i>
    </a>
}
```

*Den nya koden sitter inom blåa rektanglarna och behöver kopieras för varje knappgrupp (bilden visar endast kod för första gruppen). Koden inom första*

2017-05-22

Namn

rektangeln skapar ett UI element som ser ut som en pil åt vänster. Pilen sitter tillvänster om frasknappen och har egenskap

```
onclick="moveActionJS(@item.Id, 'back')"
```

vilket gör att JavaScript funktion "moveActionJS" kommer köras när användaren klickar på pilknappen. Funktionen tar emot id på frasknappen samt en sträng, i det här fallet "back". Dvs. frasknappen ska flyttas bakåt i listan. Det är likadant med knappen "pil åt höger" men den är anpassad för att flytta knappar åt höger och ser annorlunda ut.

Här är utseendet man får på UI:



Pilknapparna är dock inte funktionella, det behövs tre steg till.

Koden här nedan består av två delar:

En AJAX form som skickar iväg data till funktionen "MovePlayerAction" i kontroller "Config". Formen är osynlig på UI och består av tre element: två textinmatningar samt en "submit" knapp som skickar iväg formen. AJAX funktionen kommer uppdatera frasknappslistan och visa förändringarna på sidan så fort de tillämpas och sparas.

Andra delen koden är JavaScript funktion "moveActionJS" som tar emot knappinformation från pilknapparna (vilken frasknapp behöver flyttas samt åt vilket håll).

```
@using (Ajax.BeginForm("MovePlayerAction", "Config", new AjaxOptions
{ UpdateTargetId = "WholePlayerAction" }, new { @class = "hidden" }))
{
    <input id="inputActionToMoveID" name="stringId" type="text"
        class="hidden" value="" />
    <input id="inputDirectionToMoveID" name="direction" type="text"
        class="hidden" value="" />

    <input id="inputBtnMove" class="button btn hidden"
        type="submit" value="">
}
</div>

<script>
function moveActionJS(id, direction)
{
    document.getElementById('inputActionToMoveID').value = id;
    document.getElementById('inputDirectionToMoveID').value = direction;
    document.getElementById('inputBtnMove').click();
}
</script>
```

*JavaScript funktionen placerar mottagna variabler in i AJAX formen och "klickar" knappen "skicka" (submit). Det borde finnas en bättre lösning att skicka data via AJAX, men utvecklingsteamet fokuserade på att leverera fungerande prototyp och letade inte efter andra lösningar – den här lösningen är inte optimal men fungerar utan att skapa problem.*

*Informationen som skickas av AJAX formen går till kontroller. Kontroller är en kodstruktur, en del av applikationen, i det här fallet i C# programmeringsspråk, som körs på webbservern vilken behandlar data och omredigerar användaren till olika vyer ("sidor").*

*AJAX funktionen skickar iväg data till kontrollern som hanterar knappflytt på följande sätt:*

*Koden hämtar in alla knappar som ägs av användaren, hittar knappar som tillhör samma grupp som den flyttande knappen hör till, sorterar dessa knappar efter GroupPlace och sedan kör själva platsbytet mellan knappar med avseende på parameter "direction". Om knappen som användaren vill flytta på sitter längst ut*

2017-05-22

Namn

i grupplistan då sker inget platsbyte – funktionen gör inga ändringar i listor och visar knapparna i samma ordning som de var i. I slutet av funktionen sker det omredigering till funktion ”ShowPlayerActions” (se Kodexempel A) vilket gör att frasknapparna visas på sidan enligt den nya ordningen.

*Funktion ”MovePlayerAction” i kontroller som flyttar på frasknappar:*

```
public ActionResult MovePlayerAction(string stringId, string direction)
{
    int id = int.Parse(stringId);
    string CurrentDotNetID = User.Identity.GetUserId();
    User rugbyUser = rugbyTwitterDB.Users.First(i => i.DotNetID == CurrentDotNetID);
    Config rugbyConfig = rugbyTwitterDB.Configs.First(config => config.UserID == rugbyUser.Id);

    List<Entities.Action> actions = rugbyTwitterDB.Actions.Where(action => action.ConfigID == rugbyConfig.Id).ToList();

    Entities.Action actionToMove = rugbyTwitterDB.Actions.First(action => action.Id == id);

    List<Entities.Action> actionsInSameGroup = actions.Where(action => action.Group == actionToMove.Group).ToList();
    actionsInSameGroup = actionsInSameGroup.OrderBy(a => a.GroupPlace).ToList();

    int actionToMoveIndex = actionsInSameGroup.FindIndex(a => a.Id == actionToMove.Id);

    int tempActionPlace;
    int verifiableIndex;
    if (direction == "forward")
    {
        verifiableIndex = actionToMoveIndex + 1;
        if (verifiableIndex < actionsInSameGroup.Count)
        {
            tempActionPlace = (int)actionsInSameGroup[actionToMoveIndex + 1].GroupPlace;
            actionsInSameGroup[actionToMoveIndex + 1].GroupPlace = actionToMove.GroupPlace;
        }
        else
        {
            return RedirectToAction("ShowPlayerActions");
        }
    }
    else
    {
        verifiableIndex = actionToMoveIndex - 1;

        if (verifiableIndex > -1)
        {
            tempActionPlace = (int)actionsInSameGroup[actionToMoveIndex - 1].GroupPlace;
            actionsInSameGroup[actionToMoveIndex - 1].GroupPlace = actionToMove.GroupPlace;
        }
        else
        {
            return RedirectToAction("ShowPlayerActions");
        }
    }

    actionToMove.GroupPlace = tempActionPlace;

    rugbyTwitterDB.SaveChanges();

    return RedirectToAction("ShowPlayerActions");
}
```

Det är allt som krävs för att flytta på knapparna. Knapparna lades till i ordning 1-6 och A-C från början, men det gick att omvända ordningen manuellt tack vare nya funktionen.

2017-05-22

Namn



### Förbättringar till den implementerade lösningen:

Nu kan användare sortera knappar inom grupper, dvs. det går att placera frasknapparna nästan på alla möjliga sätt. En brist med lösningen är att pilknapparna tar en stor del av gränssnitt och det står i strid med produktägarens krav – gränssnitt ska vara så minimalistiskt och effektivt som möjligt.

Man kan skapa en funktion som låter de svarta pilknapparna vara osynliga tills användaren sätter på knappflyttningsfunktionen.

*Förbättringen kräver ändringar på front-end – endast HTML, CSS och JavaScript.*

*De svarta pilknapparna har redan CSS klass "action-mover", klassbeskrivning skapas i en CSS fil med egenskap "display: none" – pilknapparna blir osynliga. Sedan läggs det till en knapp i verktygsfält under frasknapparna. Knappens kod:*

```
<a id="btn-toggle-move" class="btn pull-left btn-default btn-sm btn-sm-fh action-left"
onclick="toggleMoveActionsButtons()">
  <i class="fa fa-caret-left fa-2x"></i>
  <i class="fa fa-caret-right fa-2x"></i>
</a>
```



2017-05-22

Namn

Resultat:



*Den nya knappen är markerad med ett blått streck.*

*Nu krävs det en JavaScript funktion som verktygsknappen kallar på:*

*"toggleMoveActionsButtons()"*

```
var enableMove = false;

function toggleMoveActionsButtons() {
    var moveButtons = document.getElementsByClassName('action-mover');

    enableMove = !enableMove;

    if (enableMove) {
        for (var i = 0; i < moveButtons.length; i++) {
            moveButtons[i].style.display = 'block';
        }
    }
    else {
        for (var i = 0; i < moveButtons.length; i++) {
            moveButtons[i].style.display = 'none';
        }
    }
}
```

*Bool variabel "enableMove" är inställd på "false" från början. Det vill säga, pilknapparna är avstängda från början. Funktionen hittar alla pilknappar via klassen "action-mover" och byter stilegenskap "display" med avseende på*

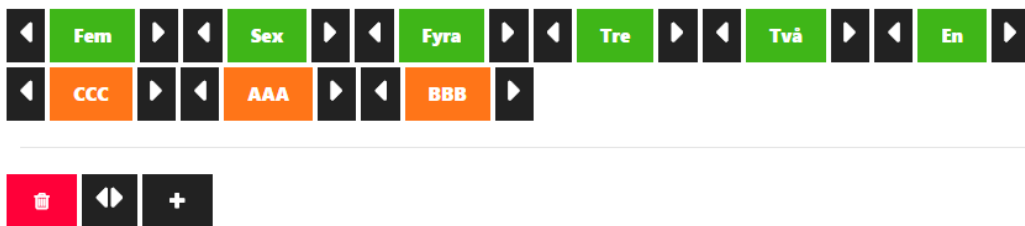
2017-05-22

Namn

*"enableMove". Om knappflyttfunktionen är på, då visas pilknapparna, annars göms dem.*

*OBS (för utvecklare): JavaScript med variabler ska inte vara beroende av "partial views" som uppdateras av AJAX eller andra funktioner, annars nollställs dem. I det här fallet leder det till att pilknapparna kommer bli osynliga igen så fort man flyttar på en frasknapp. Därför ligger JavaScript funktionen i en fil som inte laddas om av partial views. Man kan koppla JavaScript filen till huvudvyn som innehåller alla partial views.*

Resultat:



Bildan ovan visar pilknapparna efter användaren satt på dem med hjälp av verktygsfältet. Det går att stänga av pilknapparna genom att trycka på den nya knappen verktygsfältet igen (knappen mellan "sopkorgen" och "pluset")

## Diskussion/slutsats

Syftet med att använda den enkla lösningen är att spara utvecklingsresurser och ändå uppnå samma slutmål – skapa en produkt där användare har tillräckligt med flexibilitet för att anpassa frasknappar efter sina krav. Ändå skapade den enkla lösningen ett nytt problem som krävde en lösning därpå – möjlighet att sätta på och stänga av pilknappar. Det blev en mer komplex lösning än jag uppskattade från början, men det var ändå mycket mindre komplicerat än att implementera den lösningen som uppdragsgivaren föreslog.

Implementering av friplacering (koncept som produktägaren föreslog) skulle kräva mer tid att utvecklas även som basfunktion, dessutom skulle det kräva ännu fler förbättringar och stödfunktioner än den valda lösningen ”placering inom grupper”, eftersom mer komplexa funktioner i regel kräver mer plats på gränssnitt. Att ha så effektivt gränssnitt som möjligt var ett av huvudkrav från början.

2017-05-22

Namn

Här nedan finns det en jämförelsetabell som visar antal nödvändiga funktioner för dessa två olika lösningar att fungera:

Friplacering (preliminärt)	Placering inom grupper (slutversion)
<ol style="list-style-type: none"><li>1. Två nya parameter för knappobjekt (X och Y koordinater)</li><li>2. Nytt rutsystem med celler</li><li>3. Funktion som lägger ut knappar i rutsystemet enligt deras positioner</li><li>4. Nytt gränssnitt att lägga till knappar (cellplacering)</li><li>5. Funktion som kontrollerar om det finns tillräckligt med plats till närliggande knappar (så att knapparna inte läggs över varandra)</li><li>6. Funktion som tar emot och sparar knapparnas positioner</li><li>7. Funktion att flytta på befintliga knappar (både UI i HTML och back-end i kontroller)</li></ol>	<ol style="list-style-type: none"><li>1. En ny parameter för frasknappobjekt (GroupPlace)</li><li>2. Uppdatera frasknappsortering i "ShowPlacersAction"</li><li>3. Uppdatera funktionen som lägger till nya frasknappar</li><li>4. Lägga till enkla pilknappar till gränssnitt</li><li>5. Ny funktion för att flytta på knappar</li><li>6. Ny funktion för att sätta på och stänga av pilknapparna från verktygsfält</li></ol>

Tabellen visar att lösningen "placering inom grupper" är mycket enklare att implementera – det krävdes att uppdatera två funktioner, och lägga till tre nya funktioner medan friplacering av frasknappar kräver minst sex helt nya funktioner

2017-05-22

Namn

som kan brytas i ännu fler (till exempel nummer 7 i tabellen), dessutom är det bara en preliminär uppskattning vilket innebär att mängden arbete kunde öka under implementeringen.

Lösningen ”placering inom grupper” är tillräcklig för att applikationen ska vara flexibel och enkel medan ”friplacering” är både mycket mer komplicerad att utveckla och använda.

Det var värt att ta kompromiss och köra ”placering inom grupper”.

Det här exemplet visar att det är betydligt mer effektivt att arbeta agilt och skapa egna lösningar än att följa förbestämda planen, speciellt när man har begränsad tid och behöver utveckla andra delar av applikationen. Man behöver inte ta lättaste lösningen vid alla tillfällen – man gör olika val med avseende på enskilda situationer och specifika krav.

2017-05-22

Namn

## Referenser

<http://agilemanifesto.org/principles.html> - Principles behind the Agile Manifesto

2017-05-22

Namn

Bilagor

Bilaga 1 – bild på applikationens huvudsida vid slutet av praktiktiden

