DETTA ÄR ETT FÖRSÄTTSBLAD

Projektnamn	**Kontinuerlig leverans av system under test**
Beställare	**Sigma**

Sammanfattning
Här görs en sammanfattning av dokumentet som ska rymmas på denna sida.

Revisionshistorik

Datum	Revision	Anmärkning	Signatur
12-04-02	1	Rapport skapad	ML

Innehållsförteckning

Inledning

Bakgrund

Kontinuerlig leverans
Att vid utvecklingen av en produkt regelbundet bygga hela systemet för verifiering av funktionalitet är ofta önskvärt. Utvecklare får snabb feedback på de delar som inte integrerats korrekt med övriga systemet och kan då rätta till detta direkt. Om systemet fungerar korrekt fortsätter ofta utvecklingen och med korta intervaller sätts hela systemet ihop för testning av funktionalitet.
Ett positivt resultat av testningen av hela systemet bör innebära att teamet har en fullt fungerande produkt. Ändå är det ofta med mycket långt intervall som produkten levereras till kund så att denne kan ta del av ny funktionalitet. Tanken med kontinuerlig leverans är att inte bara testa ny funktionalitet regelbundet utan även leverera ny funktionalitet regelbundet. [1]

Exekverbara specifikationer
Det språk som används i en kravspecifikation är talspråk och definierar ofta en produkt utan

konkreta exempel. Det finns inte heller något som säger att det som är angivet verkligen uppfylls av produkten. Med exekverbara specifikationer minskar gapet mellan beställarens och utvecklingsteamets språk och de specifikationer som är angivna består av konkreta exempel som kan förstås av beställaren och exekveras av utvecklaren.

Syfte

Syftet med det här projektet är att ta ett helhetsgrepp om utvecklingen av en webapplikation, från idé till slutanvändare. Vid projektets slut ska en modell finnas för kontinuerlig leverans av en automatiskt testad produkt till molntjänst.

Teori

Git
Ett versionshanteringssystem som använts i detta projekt på grund av den utbredda användningen samt kopplingen till communityt GitHub(se nedan).

GitHub

Ett community där utvecklare kan ladda upp projekt utvecklade med Git revisionskontrollsystem.

Selenium

Ett verktyg för analys av innehållet på en hemsida. Givet namn eller css-id kan värden på attribut erhållas vilka sedan kan användas för testning.

Cucumber

Cucumber erbjuder ett ramverk för högnivåtest genom reguljära uttryck. Uttrycken har formen:

1. Givet att - följt av angivna värden av användaren.¨
2. När – följt av operationen användaren utfört när testet ska utvärderas.
3. Då – följt av ett booleanskt uttryck som är det uttryck som ska utvärderas.

Java EE

JSF

Ett verktyg för generering av html-sidor via java. En xhtml-fil specificerar uppläget på sidan och hämtar värden från en ”manage bean”.

Junit

Ett testverktyg för java. Genom användandet av olika assert-metoder kan funktionalitet testas. Om ett test inte går igenom visas ett tydligt meddelande om det antagande som inte stämmer.

Heroku

Heroku är en molntjänst integrerad med git för kontinuerlig leverans. Genom en anslutning från ett git-arkiv kan en web-applikations källkod levereras till och byggas på en heroku-server. I en

Procfile anges vilket sätt Heroku ska använda för att starta applikationen.

WebappRunner

Ett program som kan starta en web-applikation från en war-fil och köra denna i en tomcat-container.

JettyRunner

Jetty runner fyller samma funktion som WebAppRunner. Skillnaden är att applikationen i detta fall levereras på en Jetty-container.

Mockito

Metod

Projektet har bedrivits agilt med veckoiterationer. I början av varje vecka har arbetet planerats och i slutet utvärderats. Vid arbete har fokus legat på en enskild uppgift åt gången, exempelvis ”Leverera Hello World till molntjänst”.
Applikationen har utvecklats i små steg där varje steg innebär utökad funktionalitet för användaren.

Genomförande

Första veckan
Till en början var lärokurvan brant. Första veckan ägnades främst åt informationsinhämtning i syfte att skapa större förståelse för projektet.
De områden som kunskap inhmtädes inom var:

- Git för versionshantering.
- Maven för specificering av hur en applikation ska byggas.
- Grundläggande kunskap i jsf för utveckling av webapplikation.

En prototyp för applikationen skapades även första veckan.

Andra veckan
Kunskap om molntjänsten Heroku hämtades och ett exempelprojekt levererades. Efter detta levererades även den tidigare framtagna prototypen till molntjänsten.
Grundläggande funktionalitet lades till i applikationen.

Informationsinsamling

Informationsinhämtningen bedrevs genom sökningar på de verktyg som presenterats av handledare.

Lösningsförslagen

Om det fanns alternativa lösningar, beskriv dessa och hur man valde.

Historik
Vecka 2 – molntjänsten Heroku används för leverans av prototyp.

Problem
Till en början var det svårt med förståelsen för hela konceptet. Frågan är då om problemet ska brytas ner i mindre delar eller om fokus ska läggas på helheten. I mitt fall fick jag helheten förklarad av handledare innan själva implementationen delades upp i mindre delar.

Resultat

Molntjänster

Vid användandet av molntjänster är det viktigt att i byggspecifikationen(pomen) specificera versionen för beroenden och plugins. Detta för att applikationen ska byggas på samma sätt lokalt som på molntjänsten. Det är också att föredra att de specificerade beroendena och plugins finns i ett allmänt arkiv. I detta projekt har endast bibliotek från maven central använts.

Heroku

Heroku är relativt lätt att komma igång med och kombinerat med verktyget webapprunner eller jetty-runner kan web-applikationer sättas upp lokalt eller via heroku med några enkla kommandon. Ett problem med Heroku är att ett projekt som ska levereras till en Heroku-appcontainer måste ligga i en git-root. Ett projekt som består av Maven parent-module med undermoduler kan således inte levereras till Heroku direkt från projektets ordinarie git-arkiv. En lösning till detta är att hålla applikationsmodulen i ett separat projekt som placeras i en git-root. Andra moduler får sedan specificera ett beroende till den byggda artefakten.
Det är osäkert om web-app runner och jetty-runner passar vid driftssättande av en web-applikation. Dock är det till dessa applikationer instruktionerna för att leverera en java ee applikation pekar.[3]

Förutsatt att ett projekt versionshanteras med Git och byggs via maven utförs några enkla steg för att leverera projektet till Heroku.
Först installeras Heroku Toolbelt. Path-variabeln i Windows sätts så att Heroku-kommandon blir tillgängliga via commando-prompten. Därefter används ett kommando för inloggning på Heroku:
```
Heroku login
```
För att projektet ska kunna köras på Heroku-servern specificeras hur applikationen ska köras. Detta anges i en fil som måste heta Procfile och ligga i git-rooten.
Sedan skapas en remote-koppling från ägarens git-arkiv till Heroku.
```
Heroku create –stack cedar
```
Det som återstår är att pusha applikationen till heroku.
```
Git push Heroku master
```
Nu levereras en web-adress där applikationen presenteras.

Applikationen
Applikationen är ett lastbokningssystem.

Uppfyllnad

Måluppfyllnad och förväntningar

Funktion/kvalitet mot kravspecifikation

Acceptans av projektet

Beskriv acceptanstestets utfall, tillvägagångssätt, när det genomfördes och vem som var kundens representant.

Rekommendationer

Förslag på förbättringar och ärenden som kan vara intressanta i framtiden.

Lärdomar, erfarenheter och slutsatser

Detta är ett mycket viktigt avsnitt. Vad har vi lärt oss under projektet – positivt och negativt.

Personliga reflektioner

Att arbeta med java ee stacken skiljer sig mycket mot att utveckla en swing-applikation i en lokal miljö. Fokus ligger mycket på de verktyg som används vid byggandet och testandet av applikationen. I detta projekt har en trivial applikation utvecklats mest för att testa tillvägagångssättet att utveckla en applikation varför inga tyngre algoritmer förekommer. Det svåraste var att initialt greppa syftet med olika verktyg och tillvägagångssätt.

Slutsatser

Ett sammanfattande avsnitt som bygger på problemformuleringen, resultaten och erfarenheter. Glöm inte bort att komma fram till en slutsats om era lärandemål – har de uppfyllts?

Referenser

[1] Conitnues Delivery Jez Humble and David Farley.
[2] http://www.jamesward.com/2012/02/15/webapp-runner-apache-tomcat-as-a-dependency
[3] https://devcenter.heroku.com/articles/spring-mvc-hibernate

Bilagor

Dokument, diagram, etc. som inte ”platsar” i den löpande texten i rapporten. T.ex. kravspecifikationen, avtal, o dylikt.