# Hemuppgift 4: Web-Based Applications And Application Servers

Nätverksprogrammering, ID1212

Evan Saboo
saboo@kth.se

2017-12-07

https://github.com/ZpeedX/Network-Programming/tree/master/Homework%204/CurrencyConverter

# 1 Introduktion

Målet med denna uppgift är att utveckla en trestegs webbaserad applikation med ramverk för varje lager.
Applikationen ska också ha en strukturerad arkitektur för att kunna dela upp logik och användargränssnitt.

Uppgiften gick ut på att utveckla en webbaserad applikation som kan konvertera valuta, dvs den ska kunna konvertera en viss mängd från en valuta till en annan, tex USD till GBP.

# 2 Litteraturstudie

Uppgiften utfördes med hjälp av dessa källor:

* Jag fick mest hjälp av Applikation Server föreläsningarna. Min applikation har fått inspiration från bank applikationsexemplaret.

* Jag använde StackOverflow hemsidan för att få hjälp med problem som uppstod och för att få kort och enkel information om några java syntaxer.[1]

* Java dokumentationshemsidan var också hjälpsam för att hitta Java syntaxförklaringar.[2]

* Det fanns Youtube tutorials som hjälpte mig förstå bättre om Java EE.

[1] https://stackoverflow.com
[2] https://javaee.github.io/javaee-spec/javadocs/

# 3 Metod

Inför hemuppgiften behövde jag gå igenom 3 olika ramverk, JavaServer Faces (JSF), Enterprise JavaBeans (EJB) och Java Persistence API (JPA).
Med dessa tre ramverk kan man skapa en trestegs applikation där JSF hanteraranvändargränssnitt, EJB hanterar transaktioner mellan användargränssnitt och logik, och JPA hanterar databas operationer.
Nedan förklaras lite om varje ramverk:

**JSF**
JSF tillåter webbutvecklare att bygga användargränssnitt för JavaServer-applikationer och den stöds av webbservrar som kör Java EE.
Med JSF förenklas skapandet av webbapplikationer genom att tillhandahålla en standard uppsättning av verktyg för att bygga användargränssnitt. Till exempel kan en webbutvecklare kalla på en enkel JSF-funktion som generar ett webbformulär istället för att koda formuläret i HTML. En annan JSF-funktion kan användas för att bearbeta de data som användaren har angett. Dessa funktioner behandlas på servern och den resulterande data matas ut till klientens webbläsare.

**EJB**
Ramverket används för att utveckla skalbara, robusta och säkrare företagsapplikationer i Java. EJB erbjuder middleware-tjänster såsom säkerhet, transaktionshantering och databasoperationer (med hjälp av JPA).

**JPA**
Java Persistence API är en samling av klasser och metoder för att kontinuerligt lagra stora mängder data i en databas. Med JPA kan man lagra, uppdatera och ta bort java objekt i databasen utan att behöva tänka SQL frågor och databas struktur.

För att kunna köra applikationen i webben behövs en applikationsserver. I denna hemuppgift användes Paraya Server, en förbättrad version of Oracles GlassFish Server.

# 4 Resultat

I figur 1 visas en demonstration när applikationen körs i webbläsaren.
Man börjar med att skriva ett belopp och väljer sedan valutan man vill konvertera beloppet från och till sist väljer valutan man vill konvertera till. Man kan välja mellan 20 olika valuta konversioner och då exkluderas konversioner som har samma valuta.
Efter att man har gjort sitt val och tryckt på ”Convert” knappen visas resultatet för valuta konverteringen (Figur 1).
Om man trycker på ”Convert” knappen utan att ha skrivit in någon belopp skickas man vidare till en sida vars innehåll är ett felmeddelande (Figur 2).

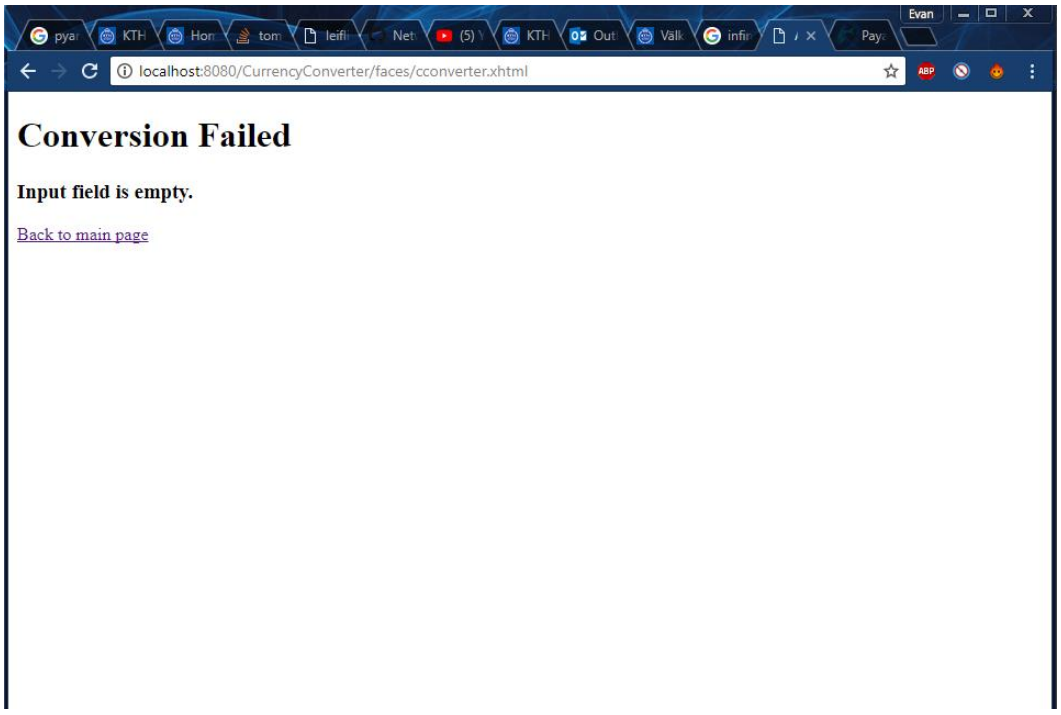
Applikationen följer MVC arkitekturen där view lagret implementerar JSF, controller implementerar EJB och model & integration implementerar JPA.

Först visas en layout för Valuta omvandlaren till användaren via webbläsaren. Denna layout finns i en xhtml fil som använder JSF för att hämta och sätta data från/i View lagret. I lagret finns det ”setter” metoder som sparar data från användaren och ”getter” metoder för att skicka data till xhtml filen som sedan visas upp för användaren.
I View finns det också metoden ”convertCurrency” som används för att konvertera beloppet användaren har matat in, vilket exekveras när användaren trycker på ”Convert” knappen som finns i xhtml filen.
Metoden börjar med att starta en ”session” eftersom kontroller lagret kan då börja utföra EJB transaktioner. Sessionen stoppas när användaren utför en valuta konvertering men skriver in fel inmatning, tex. tom inmatning.
Efter att ha startat sessionen kallar view på metoden ”convertCurrency” som finns i controller lagret.
Metoden hämtar sedan omräkningskurser från databasen för att sedan låta en metod i modell räkna ut svaret.
Svaret läggs sedan i ett DTO objekt för att returnera tillbaka till View lagret. Xhtml filen hämtar objektet från view och visar resultatet för användaren genom att ta ut värden från objektet.

Alla omräkningskurser i databasen följer valutan GBP, tex. i databasen finns valuta omvandlingen från 1 GBP till USD.
Då kan man följa formeln nedan för att konvertera en valuta till en annan:

$$Konvertade\ beloppet = \frac{Rate\ (Valuta\ man\ vill\ konverta\ till)}{Rate\ (Valuta\ man\ vill\ konverta\ från)} * Belopp$$


Figur 1: Användargränssnittet.




Figur 2: Sida för felmeddelande.

# 5 Diskussion

Kraven nedan uppfylldes:

- Programmet följer MVC arkitekturen för att ha bättre struktur på logiken och användargränssnittet och kunna kontrollera alla operationer som används mellan dem.
- Applikationen kan konvertera mellan minst 4 olika valutor.
- Varje lager i serven använder sitt eget ramverk, JSF för view, EJB för controller och JPA för model och integration.
- Server använder EJB för ”container-managed” transaktioner.
- Valutakurser finns i databasen.
- Användaren måste använda en webbläsare för att komma åt applikationen.

Svårigheten med hemuppgiften var inte applikationens funktionalitet, det var trestegs lager implementeringen.
Svårigheten uppkom när jag skulle implementera JFS, EJB och JPA. Jag behövde mer tid åt att förstå varje ramverk än att implementera dem eftersom hemuppgiften var inte så stor som de förgående hemuppgifterna.
Jag hade mest problem när jag skulle installera och konfigurera Paraya servern då jag fick flera problem när jag skulle starta upp den. Jag var tvungen ladda ner server installationen igen från deras hemsida eftersom några konfigurationsfiler var korrupta.

# 6 Kommentar om kursen

Jag har spenderad 6–7 timmar åt att gå igenom föreläsningarna innan jag började med hemuppgiften. Tiden det tog att utföra hemuppgiften var 5–6 timmar. Rapporten tog ungefär 3 timmar att skriva.