Inlämning 2 Clojure Reflection

Av Markus Bowie och Noël hennings

Skillnaden mellan funktioner och macron är när evaluation sker. Funktioner evaluerar innan koden exekveras och skickar tillbaka ett värde som inte evaluerats. Macron jobbar inte med värden utan snarare kod, eftersom den inte är evaluerad innan kan den omvandlas eller förändras. Macron skickar sedan tillbaka koden eller forms som det kallas. Macron evalueras först vid exekvering och det är då det form som macrot skickar tillbaka som kommer evalueras, inte själva macrot.

När det gäller try-catch så är det möjligt att definiera i funktioner istället för i macron. Eftersom man i det här fallet vill kontrollera att det som ska evalueras är säkert att köra, kan det däremot vara smart att implementera som ett macro eftersom värdena då inte evalueras förens macrot explicit anropas. Evalueringen sker med andra ord när och om man vill, är någonting fel i koden kan den ändras eller hoppas över. Den output man får av ett macro är säkrare eftersom den evaluerats och passar därför bra här. Eftersom man använder sig av data som inte evaluerats i form av filer utifrån och liknande blir det också ett krav att använda macron. I det här fallet skulle det inte gå att använda endast funktioner eftersom man behöver tillgång till element som inte evaluerats, däremot kan man försöka bryta ut delar ur macrot och göra till funktioner.