Clojure

Clojure (uttalas som “closure”) är en relativt nyutvecklad variant av LISP som kan köras i JVM och CLR. LISP är som bekant ett språk för funktionell programmering, något som gör det speciellt lämpat för multitrådade miljöer, ett område som bara kommer bli viktigare i framtiden i och med spridningen av proces­sorer med flera kärnor.

Anledningen till att funktionella språk är mer lämpade för multitrådad utveckling är att de i stor utsträckning saknar “tillstånd”, det vill säga att en funktion alltid returnerar samma värde om den anropas med samma parametrar. I objektorienterade språk kan resultatet av ett anrop ofta påverkas av det ägande objektets interna tillstånd, något som gör systemet mycket mer komplicerat att debugga. Detta eftersom ett objekts tillstånd i en multitrådad miljö kan påverkas av någon av de andra trådarna som körs parallellt. Den här typen av problem kan var mycket svåra att återskapa och är därför jobbiga att lösa. Dessutom är data i ett funktionellt program typiskt sett oföränder­lig, vilket innebär att flera trådar kan läsa denna data samtidigt, utan att man behöver låsa den. Kan data inte ändras kan den inte bli korrupt.

Clojure tar fördelarna med LISP och lägger till stöd för att kunna använda Javas och .NETs redan välutvecklade APIer.

**Syntax**

Liksom LISP är syntaxen i Clojure baserad runt “s-uttryck”. Varje uttryck i språket är antingen atomärt (i princip en sträng eller ett numeriskt värde) eller en kombination av två s-uttryck. I LISP är det första uttrycket i varje S-uttryck oftast en operator och de efterföljande uttrycken operander till denna operator. Ett exempel:

En jämförelse i C:

5 == (3 + 2)

En jämförelse i LISP

(= 5 (+ 3 2))

**Experimentera med Clojure**

Som med vilket språk som helst är det enklast att lära sig grunderna genom att helt enkelt testa själv. I Clojures programmeringsmiljö arbetar man i en “read-eval-print loop” (REPL). Detta är i motsats till traditionella språk som till exempel C där man arbetar i en “edit-compile-run loop” (ECRL). Detta betyder att individuella uttryck kan evalueras direkt i miljön, du behöver inte skriva hela koden, kompilera och köra programmet för att se att din kod gör det den ska.

Det enklaste sättet att komma igång med Clojure är att gå till sidan tryclj.com. Här har du en komplett Clojure-miljö med en REPL-kompatibel texteditor. Låt oss skriva ett väldigt enkelt program för att beräkna arean hos en rektangel.

Vi börjar med att definiera vår funktion. Vi kallar den area:

(defn area [x y] (\* x y))

För att anropa vår funktion kan vi nu skriva

(area 20 40); -> Ger resultatet 800

Detta är ett väldigt enkelt exempel. Vi vill nu vidareutveckla vår funktion så vi kan arbeta på en lista av rektanglar istället. Låt oss börja med att definiera en lista av rektan­glar som vi kan arbeta med.

(def rects [[34 5] [89 2] [32 45]])

Detta definierar upp en struktur åt oss som innehåller en lista av värdepar, där varje värdepar representerar en rektangel med höjd och bredd. Vi måste nu skriva om vår area-funktion så den kan hantera den här typen av struktur. Vi kallar vår nya funktion för rect-area.

(defn rect-area [r] (\* (first r) (second r)))

Vi vill nu applicera vår funktion rect-area på alla individer i rect. För att göra detta till­handahåller Clojure nyckelordet “map”. Map kommer att ta varje sak i listan och skicka det som argument till funktionen vi anger.

(map rect-area rects)

När vi kör detta genom vår REPL-miljö får vi svaret (170, 178, 1440), alltså en lista av alla areor av våra definierade rektanglar.

**Sammanfattning**

Clojure är en modern anpassning av LISP, som behåller LISPs styrkor och lägger till stöd för de APIer som tillhandahålls av Java och .NET. Utöver detta finns stöd för webutveckling genom projektet Noir.

Det kan ta tid att sätta sig in i det nya tank­esätt som funktionell programmering faktiskt innebär, men i framtiden kommer denna typ av programmering att spela en allt större roll på grund av dess styrkor inom multitrådning så det kommer vara värt ansträngningen. Tack vare Clojures rötter inom Java och .NET så är språket ett bra alternativ att titta på oavsett vilken plattform utvecklingen sker i.