Definition: deftemplate

```
(deftemplate prospect     ; name of deftemplate
   "Vital Information"    ; optional comment
   (slot name             ; name of field
      (type STRING)       ; type of field
      (default ?DERIVE))  ; default value of field
   (slot assets
      (type SYMBOL)
      (default rich))
   (slot age
      (type NUMBER)
      (default 80)))

(assert (prospect))
(facts)
(assert (prospect (age 99) (name "Dopey")))
(facts)
```

Note that the default for *?DERIVE* is the null string. Every data type has a standard default value. Note that order of how the fields are given is not important.

Another Example of deftemplate

```
CLIPS> (deftemplate ratings
                    (slot grade
                         (allowed-symbols a b c d f))
                    (slot scale
                          (range 1 10))
                    )
CLIPS> (assert (ratings))
<Fact-0>
CLIPS> (facts)
f-0     (ratings (grade a) (scale nil)) CF 1.00
For a total of 1 fact.
CLIPS> (assert (ratings (grade a) (scale 5)))
<Fact-1>
CLIPS> (facts)
f-0     (ratings (grade a) (scale nil)) CF 1.00
f-1     (ratings (grade a) (scale 5)) CF 1.00
For a total of 2 facts.
CLIPS> (assert (ratings (scale 11)))

[CSTRNCHK1] A literal slot value found in the assert command
does not fall in the allowed range 1 to 10 for slot scale.
CLIPS> CLIPS> (assert (ratings (grade e)))

[CSTRNCHK1] A literal slot value found in the assert command
does not match the allowed values for slot grade.
CLIPS>
```

Accessing and using Templates

```
CLIPS> (deftemplate prospect     ; name of deftemplate
   "Vital Information"    ; optional comment
   (slot name             ; name of field
      (type STRING)       ; type of field
      (default ?DERIVE))  ; default value of field
   (slot assets
      (type SYMBOL)
      (default rich))
   (slot age
      (type NUMBER)
      (default 80)))

CLIPS> (defrule matrimonial-candidate
          (prospect (name ?name) (assets ?net-worth) (age ?months))
        =>
          (printout t "Prospect: " ?name crlf
                      "Worth:      " ?net-worth crlf
                      "Age           " ?months crlf crlf))
CLIPS> (assert (prospect (name "Dopey Wonderful") (age 99)))
<Fact-0>
CLIPS> (run)
Prospect: Dopey Wonderful
Worth:      rich
Age           99

CLIPS>
```

Manipulation of Templates

```
CLIPS> (defrule bye-bye
           ?prospect <- (prospect (assets poor) (name ?name))
         =>
           (retract ?prospect)
           (printout t "Bye Bye " ?name crlf))
CLIPS> (reset)
CLIPS> (assert  (prospect  (name "Dopey Wonderful") (assets rich)))
<Fact-1>
CLIPS>  (assert  (prospect  (name "Dopey  NotWonderful")  (assets poor)))
<Fact-2>
CLIPS> (facts)
f-0     (initial-fact) CF 1.00
f-1     (prospect (name "Dopey Wonderful") (assets rich) (age 80)) CF 1.00
f-2     (prospect (name "Dopey  NotWonderful") (assets poor) (age 80)) CF 1.00
For a total of 3 facts.
CLIPS> (run)
Bye Bye Dopey  NotWonderful
CLIPS> (facts)
f-0     (initial-fact) CF 1.00
f-1     (prospect (name "Dopey Wonderful") (assets rich) (age 80)) CF 1.00
For a total of 2 facts.
CLIPS>
```

Multiple Slot Entries

```
CLIPS> (deftemplate prospect
(multislot name
    (type  SYMBOL)
    (default ?DERIVE))
(slot assets
   (type SYMBOL)
    (allowed-symbols poor rich wealthy loaded)
    (default rich))
(slot age
    (type INTEGER)
    (range 80 ?VARIABLE)
    (default 80)))
CLIPS> (deffacts duck-bachelor
(prospect (name Dopey Wonderfu) (assets rich) (age 99)))
CLIPS>  (defrule happy_relationship
              (prospect (name $?name)  (assets ?net_worth) (age ?months))
          =>
              (printout t "Prospect:         " ?name crlf
                          "Worth:              " ?net_worth crlf
                          "Age (months)  " ?months crlf crlf))
CLIPS> (reset)
CLIPS> (run)
Prospect:         (Dopey Wonderfu)
Worth:              rich
Age (months)  99

CLIPS>
```

Modifying specific entries

```
CLIPS> (undefrule *)
CLIPS>  (defrule make-bad-guys
           ?prospect <- (prospect (name $?name) (assets rich) (age ?months))
         =>
           (printout t "ExProspect:      " ?name crlf
                       "Rich guy" crlf
                       "Age                    " ?months crlf crlf)
           (modify ?prospect (assets poor)))
CLIPS> (deffacts duck-bachelor
          (prospect (name Dopey Wonderful) (assets rich) (age 99)))
CLIPS> (reset)
CLIPS> (run)
ExProspect:      (Dopey Wonderful)
Rich guy
Age                   99

ExProspect:      (Dopey Wonderful)
Poor GuyAge                   99

CLIPS> (facts)
f-0     (initial-fact) CF 1.00
f-2     (prospect (name Dopey Wonderful) (assets poor) (age 99)) CF 1.00
For a total of 2 facts.
CLIPS>
```