OClock - Part Two

Let extend the clock to return JSON

Modify test

```
....

func TestOClock_Body(t *testing.T) {
    req := httptest.NewRequest("GET", "http://example.com", nil)
    w := httptest.NewRecorder()
    OClock(w, req)
    value := time.Now().Format("2006-01-02T15:04")
  expected := fmt.Sprintf(`{"time":"%s"}`,value)
    result := w.Body.String()
    if result != expected {
        t.Errorf("Expected %q Result %q", expected, result)
    }

}
```

```
=== RUN   TestOClock
--- PASS: TestOClock (0.00s)
=== RUN   TestOClock_Body
--- FAIL: TestOClock_Body (0.00s)
    main_test.go:29: Expected "{\"time\":\"2017-02-15T21:15\"}" Result "2017-02-15
T21:15"
FAIL
exit status 1
FAIL    oclock  0.009s
```

Code Change

```
type clock struct {
    Time time.Time
}

func OClock(w http.ResponseWriter, r *http.Request) {

    c := clock{Time: time.Now()}

    cJSON, err := json.Marshal(c)
    if err != nil {
        fmt.Printf("Can't marshal time - %v\n", err.Error())
        w.WriteHeader(http.StatusInternalServerError)
    }

    w.Write([]byte(cJSON))
}
```

Test

go test -v oclock

```
=== RUN   TestOClock
--- PASS: TestOClock (0.00s)
=== RUN   TestOClock_Body
--- FAIL: TestOClock_Body (0.00s)
    main_test.go:29: Expected "{\"time\":\"2017-02-15T21:35\"}" Result "{\"Time\":
\"2017-02-15T21:35:31.152756804Z\"}"
FAIL
exit status 1
FAIL    oclock  0.009s
```

Code Change

```
type clock struct {
    Time time.Time `json:"time"`
}
```

Test

go test -v oclock

```
=== RUN   TestOClock
--- PASS: TestOClock (0.00s)
=== RUN   TestOClock_Body
--- FAIL: TestOClock_Body (0.00s)
    main_test.go:29: Expected "{\"time\":\"2017-02-15T21:35\"}" Result "{\"time\":
\"2017-02-15T21:35:31.152756804Z\"}"
FAIL
exit status 1
FAIL    oclock  0.009s
```

Code Change

```
type clock struct {
    Time jsonTime `json:"time"`
}

type jsonTime time.Time

func (t jsonTime) MarshalJSON() ([]byte, error) {
    return []byte(time.Time(t).Format("2006-01-02T15:04")), nil
}
```

Test

go test -v oclock

```
=== RUN   TestOClock
--- PASS: TestOClock (0.00s)
=== RUN   TestOClock_Body
--- PASS: TestOClock_Body (0.00s)
PASS
ok      oclock  0.009s
```