WAY2LEARN

LAB- Using Try, fail and log

In this lab you will understand how to write defensive code using try , fail functions and also logging in dwl

Create a project with name **05-using-try-fail-log-start**

Create a new Mule configuration file with name “trydemo.xml”

Drag a “Transform Message” component into a new flow and rename the flow as “trydemo”

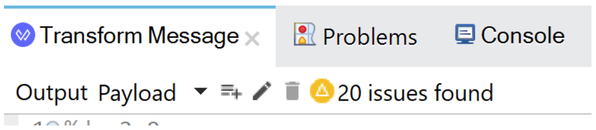
Copy states.xml given to you into src/main/resources/examples/states.xml and observe it.

Write a 2 functions like below to get state details by state name

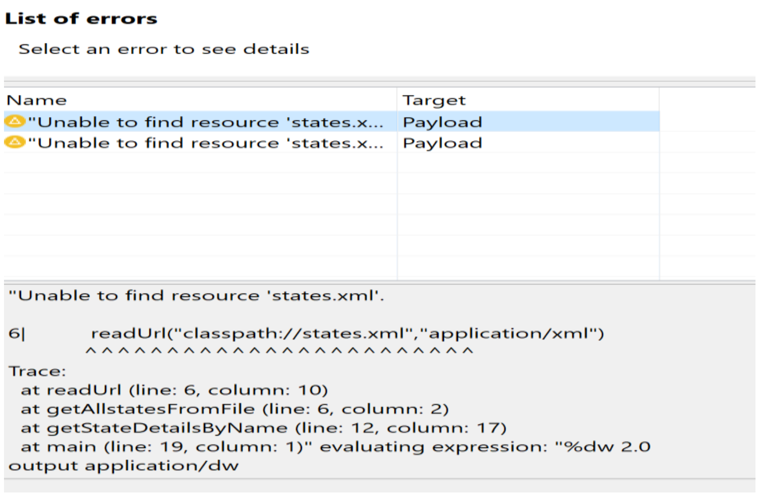
```
fun getAllstatesFromFile(inputFileName:String)=
      readUrl("classpath://" ++ inputFileName,"application/xml")


fun getStateDetailsByName(name) =

do{
      var statesXml= getAllstatesFromFile("states.xml")
      ---
      statesXml.states.*state[?($.@name ~= name)].@[0]
}
```

Now change the body expression to `getStateDetailsByName("California")`

Click on issues found


Transform Message × Problems Console
Output Payload 20 issues found

Observe that you will get an error like below:



List of errors

Select an error to see details

Name	Target
⚠"Unable to find resource 'states.x...	Payload
⚠"Unable to find resource 'states.x...	Payload

```
"Unable to find resource 'states.xml'.

6|        readUrl("classpath://states.xml","application/xml")
          ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
Trace:
  at readUrl (line: 6, column: 10)
  at getAllstatesFromFile (line: 6, column: 2)
  at getStateDetailsByName (line: 12, column: 17)
  at main (line: 19, column: 1)" evaluating expression: "%dw 2.0
output application/dw
```

This is because states.xml is in examples folder

Make a call to dw::Runtime::try function as shown below:

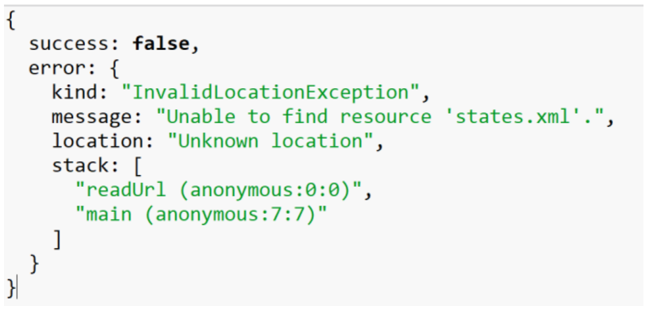
```
fun getAllstatesFromFile(inputFileName:String)=
        dw::Runtime::try(
        readUrl("classpath://"++ inputFileName,"application/xml")
)
```

Still you should see the same error

Now change to

```
fun getAllstatesFromFile(inputFileName:String)=
        dw::Runtime::try(
        ()-> readUrl("classpath://"++ inputFileName,"application/xml")
        )
```

Verify that error is resolved but appears in TryResult object's error element

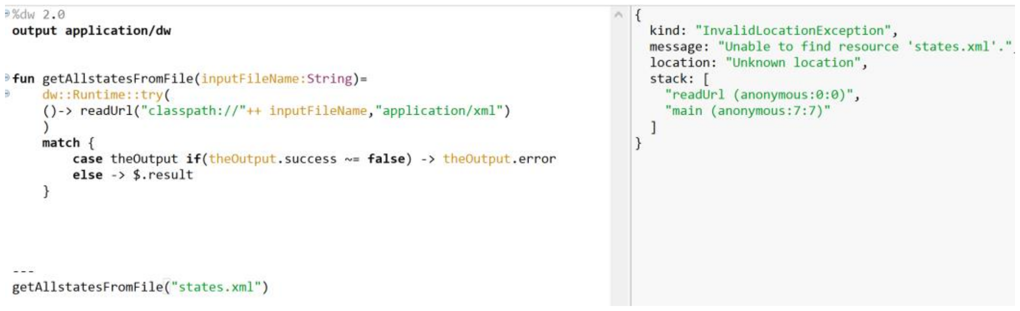


```
{
  success: false,
  error: {
    kind: "InvalidLocationException",
    message: "Unable to find resource 'states.xml'.",
    location: "Unknown location",
    stack: [
      "readUrl (anonymous:0:0)",
      "main (anonymous:7:7)"
    ]
  }
}
```

Chain the TryResult to match as shown below:

```
fun getAllstatesFromFile(inputFileName:String)=
        dw::Runtime::try(
        ()-> readUrl("classpath://"++ inputFileName,"application/xml")
        )
        match {
                case theOutput if(theOutput.success ~= false) -> theOutput.error
                else -> $.result
        }
```

Now, you should see the error object in preview as shown below:



```
%dw 2.0
output application/dw

fun getAllstatesFromFile(inputFileName:String)=
    dw::Runtime::try(
    ()-> readUrl("classpath://"++ inputFileName,"application/xml")
    )
    match {
        case theOutput if(theOutput.success ~= false) -> theOutput.error
        else -> $.result
    }



---
getAllstatesFromFile("states.xml")
```

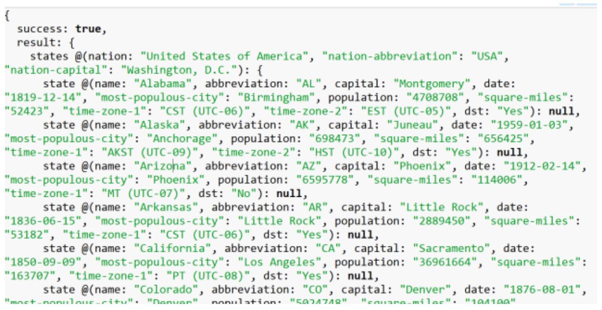
```
{
  kind: "InvalidLocationException",
  message: "Unable to find resource 'states.xml'."
  location: "Unknown location",
  stack: [
    "readUrl (anonymous:0:0)",
    "main (anonymous:7:7)"
  ]
}
```

Let us add logic in the first case to look for file in classpath://examples folder

Modify the function as shown below:

```
fun getAllstatesFromFile(inputFileName:String)=
        try(
         ()-> readUrl("classpath://"++ inputFileName,"application/xml")
        )
        match {
                case theOutput if(theOutput.success ~= false) ->
                        try(
                          ()->readUrl("classpath://examples/"++
inputFileName,"application/xml")
                        )
                else -> $.result
        }
```

Now, as the file is available in examples folder, try function returns TryResult . Observe that preview output looks like below:



```
{
  success: true,
  result: {
    states @(nation: "United States of America", "nation-abbreviation": "USA",
"nation-capital": "Washington, D.C."): {
      state @(name: "Alabama", abbreviation: "AL", capital: "Montgomery", date:
"1819-12-14", "most-populous-city": "Birmingham", population: "4708708", "square-miles":
"52423", "time-zone-1": "CST (UTC-06)", "time-zone-2": "EST (UTC-05)", dst: "Yes"): null,
      state @(name: "Alaska", abbreviation: "AK", capital: "Juneau", date: "1959-01-03",
"most-populous-city": "Anchorage", population: "698473", "square-miles": "656425",
"time-zone-1": "AKST (UTC-09)", "time-zone-2": "HST (UTC-10)", dst: "Yes"): null,
      state @(name: "Arizona", abbreviation: "AZ", capital: "Phoenix", date: "1912-02-14",
"most-populous-city": "Phoenix", population: "6595778", "square-miles": "114006",
"time-zone-1": "MT (UTC-07)", dst: "No"): null,
      state @(name: "Arkansas", abbreviation: "AR", capital: "Little Rock", date:
"1836-06-15", "most-populous-city": "Little Rock", population: "2889450", "square-miles":
"53182", "time-zone-1": "CST (UTC-06)", dst: "Yes"): null,
      state @(name: "California", abbreviation: "CA", capital: "Sacramento", date:
"1850-09-09", "most-populous-city": "Los Angeles", population: "36961664", "square-miles":
"163707", "time-zone-1": "PT (UTC-08)", dst: "Yes"): null,
      state @(name: "Colorado", abbreviation: "CO", capital: "Denver", date: "1876-08-01",
"most-populous-city": "Denver", population: "5024748", "square-miles": "104100"
```

Let us chain the result of second try with match to extract result of TryResult .

Modify the function as shown below:

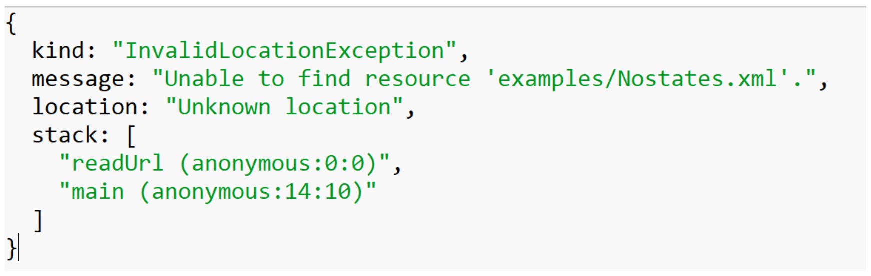
```
fun getAllstatesFromFile(inputFileName:String)=
       try(
        ()-> readUrl("classpath://"++ inputFileName,"application/xml")
       )
       match {
              case theOutput if(theOutput.success ~= false) ->
                     try(
                       ()->readUrl("classpath://examples/"++
inputFileName,"application/xml")
                     )      match {
                              case secondTryResult
                              if(secondTryResult.success ~=false) ->
secondTryResult.error

                              else -> $.result
                     }
              else -> $.result
       }
```

Now, you should see the contents for file in preview.

Change the body expression to call the function with a file which doesn't exist.

```
getAllstatesFromFile("Nostates.xml")
```

you should see the error object in the preview as shown below:



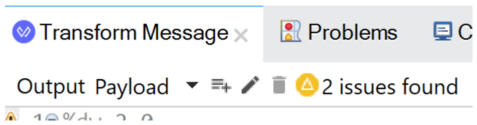
```
{
  kind: "InvalidLocationException",
  message: "Unable to find resource 'examples/Nostates.xml'.",
  location: "Unknown location",
  stack: [
    "readUrl (anonymous:0:0)",
    "main (anonymous:14:10)"
  ]
}
```

Now let us rethrow the error message if the file is not fount in examples folder also.

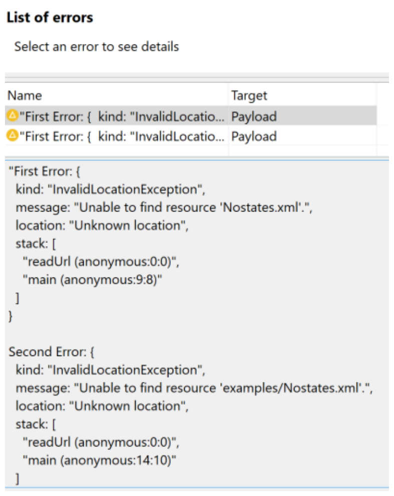
Change the function as shown :

```
fun getAllstatesFromFile(inputFileName:String)=
       try(
        ()-> readUrl("classpath://"++ inputFileName,"application/xml")
       )
       match {
              case theOutput if(theOutput.success ~= false) ->
                     try(
                       ()->readUrl("classpath://examples/"++
inputFileName,"application/xml")
                     )      match {
                                   case secondTryResult
                                   if(secondTryResult.success ~=false) ->
                                      fail("First Error: " ++ write(theOutput.error)
                                                   ++ "\n\nSecond Error: " ++
write(secondTryResult.error))

                                   else -> $.result
                     }
              else -> $.result
       }
```

Change Verify that this error message appears if you click on issues button



Error message looks like below:


List of errors

Select an error to see details

Name	Target
"First Error: { kind: "InvalidLocatio...	Payload
"First Error: { kind: "InvalidLocatio...	Payload

```
"First Error: {
  kind: "InvalidLocationException",
  message: "Unable to find resource 'Nostates.xml'.",
  location: "Unknown location",
  stack: [
    "readUrl (anonymous:0:0)",
    "main (anonymous:9:8)"
  ]
}

Second Error: {
  kind: "InvalidLocationException",
  message: "Unable to find resource 'examples/Nostates.xml'.",
  location: "Unknown location",
  stack: [
    "readUrl (anonymous:0:0)",
    "main (anonymous:14:10)"
  ]
```

We want to verify if this error is propagated into flow.

Drag "on Error Continue" in to Error Handling portion of flow

Drag "Set Payload" into "on Error Continue" and set its value as

```
output application/json --- error
```

Drag Http Listener into source part of the flow and configure it with path /try

Run the application and give a request to http://localhost:8081/try.

You should see the error message on your browser

We want to add logging incase of failure.

Modify the function as shown below:

```
fun getAllstatesFromFile(inputFileName:String)=
       try(
        ()-> readUrl("classpath://"++ inputFileName,"application/xml")
       )
       match {
              case theOutput if(theOutput.success ~= false) ->
                     try(
                             log( "Could not find classpath:// "
                                   ++ inputFileName ++"trying in examples folder",
                             ()->readUrl("classpath://examples/" ++
inputFileName,"application/xml")
                             )
                     //  ()->readUrl("classpath://examples/"++
inputFileName,"application/xml")
                     )      match {
                                   case secondTryResult
                                   if(secondTryResult.success ~=false) ->
                                      fail("First Error: " ++ write(theOutput.error)
                                             ++ "\n\nSecond Error: " ++
write(secondTryResult.error))

                                   else -> $.result
                     }
              else -> $.result
       }
```

Now run the application and give a request to http://localhost:8081/try

Observe the log in the console

This is the end of the Exercise