JMETER - REGULAR EXPRESSIONS

Regular expressions are used to search and manipulate text, based on patterns. JMeter interprets forms of regular expressions or patterns being used throughout a JMeter test plan, by including the pattern matching software Apache Jakarta ORO.

With the use of regular expressions, we can certainly save a lot of time and achieve greater flexibility as we create or enhance a Test Plan. Regular expressions provide a simple method to get information from pages when it is impossible or very hard to predict an outcome.

> *A standard usage example of using expressions is to get a session ID from the server response. If the server returns a unique session key we can easily get it using expressions in our load script.*

To use regular expressions in your test plan, you need to use the Regular Expression Extractor of JMeter. You can place regular expressions in any component in a Test Plan.

It is worth stressing the difference between **contains** and **matches**, as used on the Response Assertion test element −

- **contains** means that the regular expression matched at least some part of the target, so 'alphabet' "contains" 'ph.b.' because the regular expression matches the substring 'phabe'.
- **matches** means that the regular expression matched the whole target. Hence the 'alphabet' is "matched" by 'al.*t'.

Suppose you want to match the following portion of a web-page −

```
name="file" value="readme.txt"
```

And you want to extract readme.txt. A suitable regular expression would be −

```
name="file" value="(.+?)">
```

The special characters above are −

- **(** and **)** − these enclose the portion of the match string to be returned
- **.** − match any character
- **+** − one or more times
- **?** − stop when first match succeeds

Create JMeter Test Plan

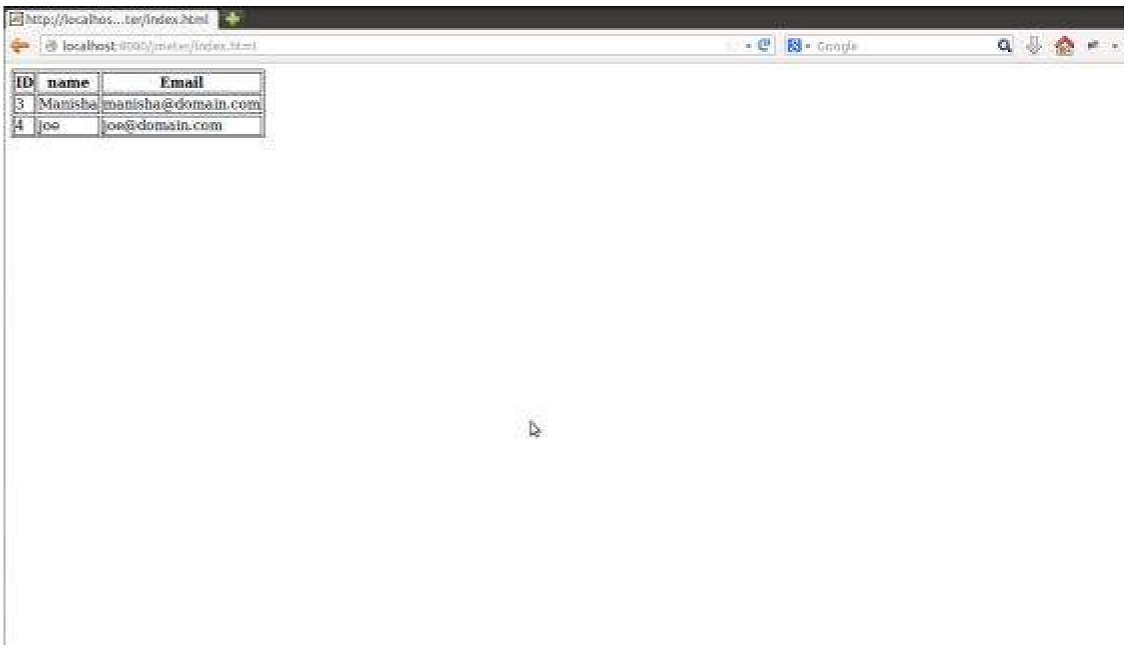
Let us understand the use of Regular expressions in the Regular Expression Extractor—a Post-Processor Element by writing a test plan. This element extracts text from the current page using a Regular Expression to identify the text pattern that a desired element conforms with.

First we write an HTML page which a list of people and their email IDs. We deploy it to our tomcat server. The contents of html *index.html* are as follows −

```
<html>
   <head>
   </head>

   <body>

      <table style="border: 1px solid #000000;">
```

```
         <th style="border: 1px solid #000000;">ID</th>
            <th style="border: 1px solid #000000;">name</th>
            <th style="border: 1px solid #000000;">Email</th>

         <tr>
            <td >3</td>
            <td >Manisha</td>
            <td >manisha@domain.com</td>
         </tr>

         <tr>
            <td >4</td>
            <td >joe</td>
            <td >joe@domain.com</td>
         </tr>

      </table>

   </body>

</html>
```

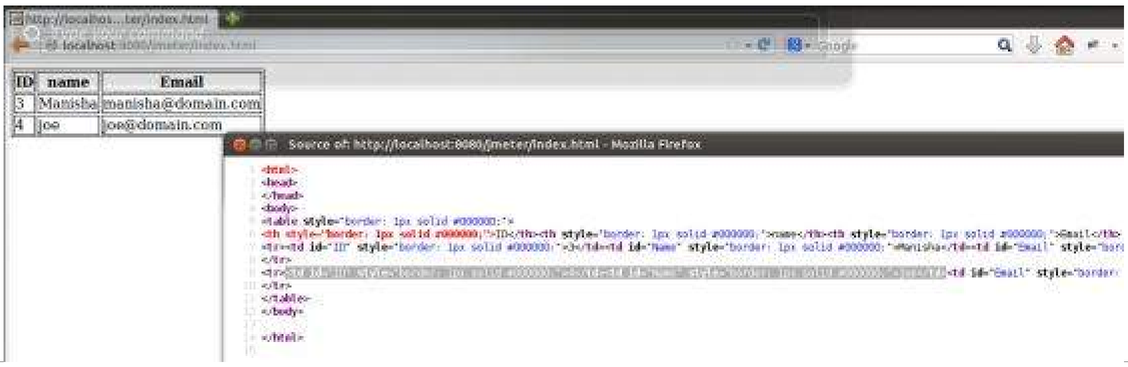
On deploying it on the tomcat server, this page would look like as shown in the following screenshot −



ID	name	Email
3	Manisha	manisha@domain.com
4	joe	joe@domain.com

In our test plan, we will select the person in the first row of the person table seen in the person list page above. To capture the ID of this person, let us first determine the pattern where we will find the person in the second row.

As can be seen in the following snapshot, the ID of the second person is surrounded by <td > and </td >, and it is the second row of data having this pattern. We can use this to match the exact pattern that we want to extract information from. As we want to extract two pieces of information from this page, the person ID and the person name, the fields are defined as follows −



ID	name	Email
3	Manisha	manisha@domain.com
4	joe	joe@domain.com

Source of: http://localhost:8080/jmeter/index.html - Mozilla Firefox

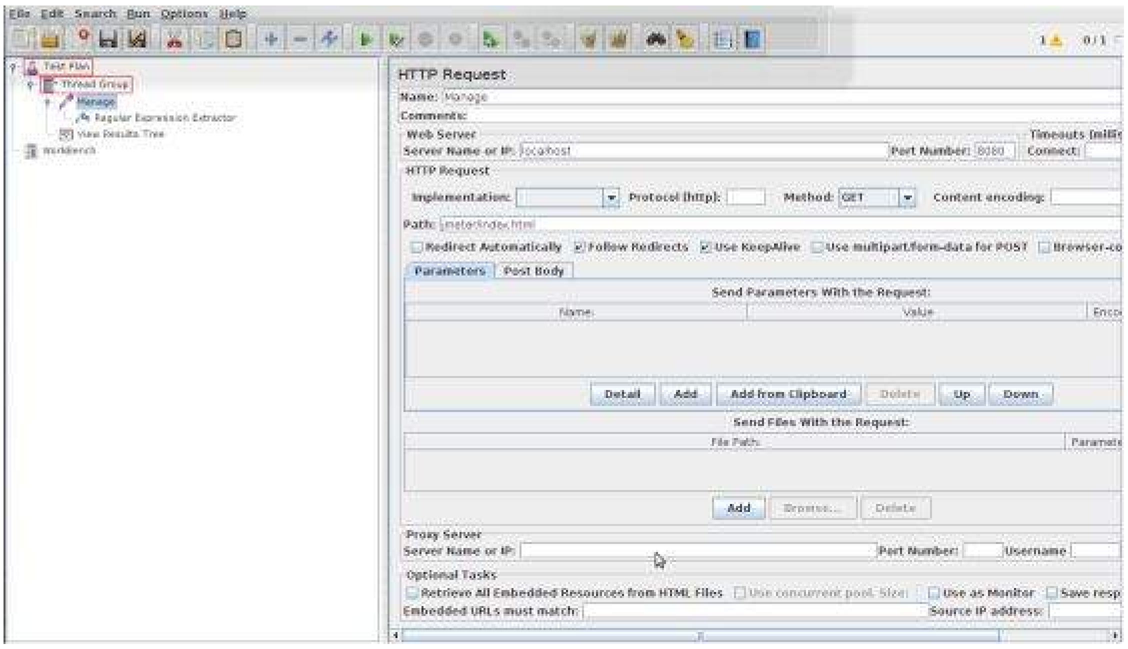
```
<html>
<head>
</head>
<body>
<table style="border: 1px solid #000000;">
<th style="border: 1px solid #000000;">ID</th><th style="border: 1px solid #000000;">name</th><th style="border: 1px solid #000000;">Email</th>
<tr><td id="ID" style="border: 1px solid #000000;">3</td><td id="Name" style="border: 1px solid #000000;">Manisha</td><td id="Email" style="bor
</tr>
<tr>[illegible]<td id="Email" style="border:
</tr>
</table>
</body>

</html>
```

Line 9, Col 1:15

Start JMeter, add a Thread group **Test Plan > Add> Threads*Users*> Thread Group**.

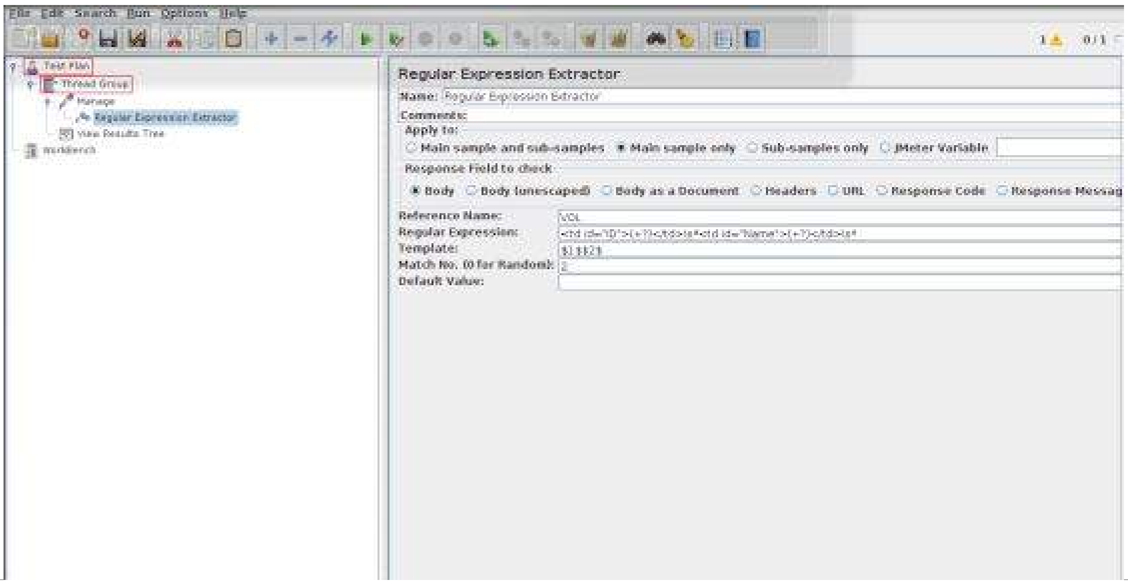
Next add a sampler HTTP Request, select the test plan, right click **Add > Sampler > HTTP Request** and enter the details as shown below −

- **Name** − Manage
- **Server Name or IP** − localhost
- **Port Number** − 8080
- **Protocol** − We will keep this blank, which means we want HTTP as the protocol.
- **Path** − jmeter/index.html

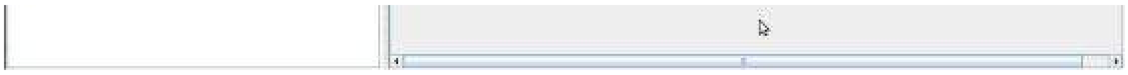


HTTP Request
Name: Manage
Comments:
Web Server
Server Name or IP: localhost
Port Number: 8080
HTTP Request
Implementation:
Protocol [http]:
Method: GET
Content encoding:
Path: jmeter/index.html
Redirect Automatically [✓ Follow Redirects] [✓ Use KeepAlive] Use multipart/form-data for POST
Parameters | Post Body
Send Parameters With the Request:
Name: | Value
Send Files With the Request:
File Path:
Proxy Server
Server Name or IP:
Port Number:
Username
Optional Tasks
Retrieve All Embedded Resources from HTML Files
Use concurrent pool. Size:
Use as Monitor
Embedded URLs must match:
Source IP address:

Next, add a Regular Expression Extractor. Select the HTTP Request Sampler *Manage*, right click **Add > Post Processor > Regular Expression Extractor**.


File Edit Search Run Options Help
Test Plan
Thread Group
Manage
Regular Expression Extractor
View Results Tree
WorkBench

Regular Expression Extractor
Name: Regular Expression Extractor
Comments:
Apply to:
Main sample and sub-samples [✓ Main sample only] Sub-samples only JMeter Variable
Response Field to check
[✓ Body] Body (unescaped) Body as a Document Headers URL Response Code Response Message
Reference Name: VOL
Regular Expression: <td id="ID">(+?)</td>\s*<td id="Name">(+?)</td>\s*
Template: $1$$2$
Match No. (0 for Random): 2
Default Value:

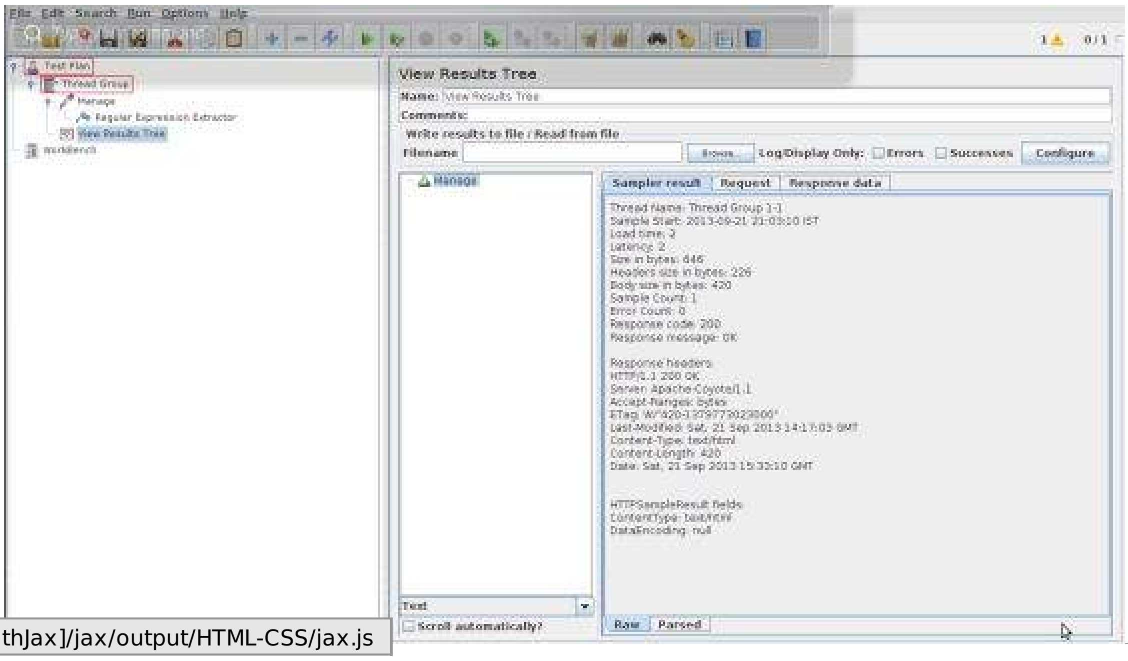


The following table provides a description of the fields used in the above screenshot −

Field	**Description**
Reference Name	The name of the variable in which the extracted test will be stored *refname*.
Regular Expression	The pattern against which the text to be extracted will be matched. The text groups that will extracted are enclosed by the characters '$'and'$'. We use '.+?' to indicate a single instance of the text enclosed by the <td..>..</td> tags. In our example the expression is − <td >$+?$ </td>\s*
Template	Each group of extracted text placed as a member of the variable Person, following the order of each group of pattern enclosed by '$'and'$'. Each group is stored as refname_g#, where refname is the string you entered as the reference name, and # is the group number. 1 to refers to group 1, 2 to refers to group 2, etc. 0 refers to whatever the entire expression matches. In this example, the ID we extract is maintained in Person_g1, while the Name value is stored in Person_g2.
Match No.	Since we plan to extract only the second occurrence of this pattern, matching the second volunteer, we use value 2. Value 0 would make a random matching, while a negative value needs to be used with the ForEach Controller.
Default	If the item is not found, this will be the default value. This is an optional field. You may leave it blank.

Add a listener to capture the result of this Test Plan. Right-click the Thread Group and select Add > Listener > View Results Tree option to add the listener.

Save the test plan as *reg_express_test.jmx* and run the test. The output would be a success as shown in the following screenshot −



```
Thread Name: Thread Group 1-1
Sample Start: 2013-09-21 21:03:10 IST
Load time: 2
Latency: 2
Size in bytes: 646
Headers size in bytes: 226
Body size in bytes: 420
Sample Count: 1
Error Count: 0
Response code: 200
Response message: OK

Response headers:
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Accept-Ranges: bytes
ETag: W/"420-1379772023000"
Last-Modified: Sat, 21 Sep 2013 14:17:03 GMT
Content-Type: text/html
Content-Length: 420
Date: Sat, 21 Sep 2013 15:33:10 GMT


HTTPSampleResult fields:
ContentType: text/html
DataEncoding: null
```

Loading [MathJax]/jax/output/HTML-CSS/jax.js