



# Performance Testing Course : JMeter - Post Processor

---

Author: Hien HOANG



# Agenda

---

- Introduction about several popular post extractor.
- Showcases.
- Exercises.

# Definition

---

JMeter has some of the elements which execute after receiving the response from the server.

=> Note: Post-Processors run before Assertions, so they do not have access to any Assertion Results, nor will the sample status reflect the results of any Assertions.

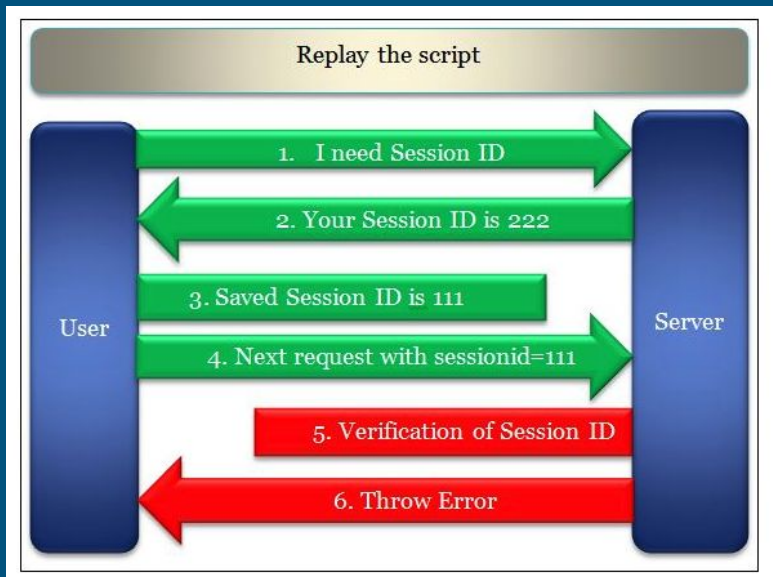
# What is Correlation?

---

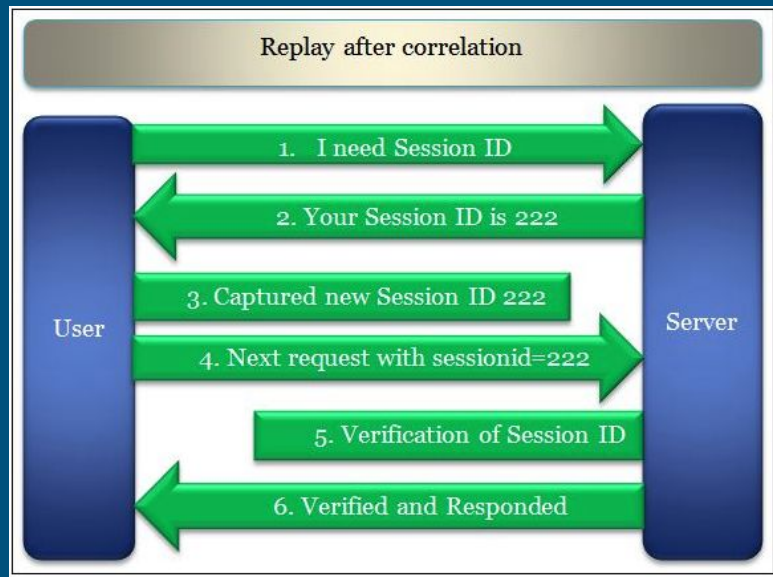
We sometimes need to parameterize dynamic value (data which changes) coming from the server that needs to be handled or captured and pass into the next request as a parameter. This process is called "Correlation" or "Handling of Dynamic Values".

# Why correlation is important?

- Without handling the dynamic value



- With correlation



# What dynamic value is important?

---

- Session ID
- Access Token
- Customer Name / ID
- Order Number
- Bill Number
- Number of records displayed on a page
- Current Date and Time

=> Ultimate goal must be to find out each and every dynamic value which causes failure of the script.

# Regular Expression- Definition

---

Regular Expression is a pattern which is used to specify a set of strings required for getting dynamic value.

-> It can work on XML, JSON or any structure and it has small processing time.

# Regular Expression syntax - Basics - Quantifier


Tokens	Meaning	Example
.	Matches any character other than newline.	Test string : <b>USD 950,00 Regex : <b>USD . Result : 9
?	Matches a specific character that occurs 0 or 1 time	Test string : <b>USD 950,00 Regex : <b>USD 9? Result : 9
+	Matches a specific character that occurs 1 or many times	Test string : <b>USD 950,000 Regex : <b>USD 9+ Result : 950,000
*	Matches a specific character that occurs 0 or many times	Test string : <b>USD 950,000 Regex : <b>USD 9* Result : 950,000
{X,Y}	Matches a specific character that occurs X to Y times	Test string : <b>USD 950,000 Regex : <b>USD 9{2,3} Result : no matched result



# Regular Expression syntax - Basics - Meta characters

Tokens	Meaning	Example
\d	Matches any digits from 0 -> 9	Test string : <b>USD 950,000 Regex : <b>USD \d Result : 9
\D	Matches any non-digits characters	Test string : <b>USD 950,00 Regex : <b>USD \D Result : no matched result
\w	Matches any word character	Test string : <b>USD 950,000 Regex : <b>\w Result : U
\W	Matches any other word character	Test string : <b>USD 950,00 Regex : <b>\W Result : no matched result

# Regular Expression syntax - Basics(continued) - Matching symbol

Tokens	Meaning	Example
<i>^</i> regex	Matches the strings from the beginning of line	Test string : <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" Regex : ^. Result : 
regex\$	Matches the strings in the end of line.	
[abc]	Matches any character a, b or c	
[abc][xz]	Matches any character a,b or c following by x or z	
(a b)	Matches character a or b	

# Regular Expression syntax - Basics(continued) - Flag

Flag	Meaning	Example
/g	Don't return after first match	Test string : Regular Expression is a pattern Regex : (.+)/g Result : Regular Expression is a pattern
/m	Multi line	Test string : abc def Regex : .+ Result : abc def

# Regular Expression Extractor

## Attribute:

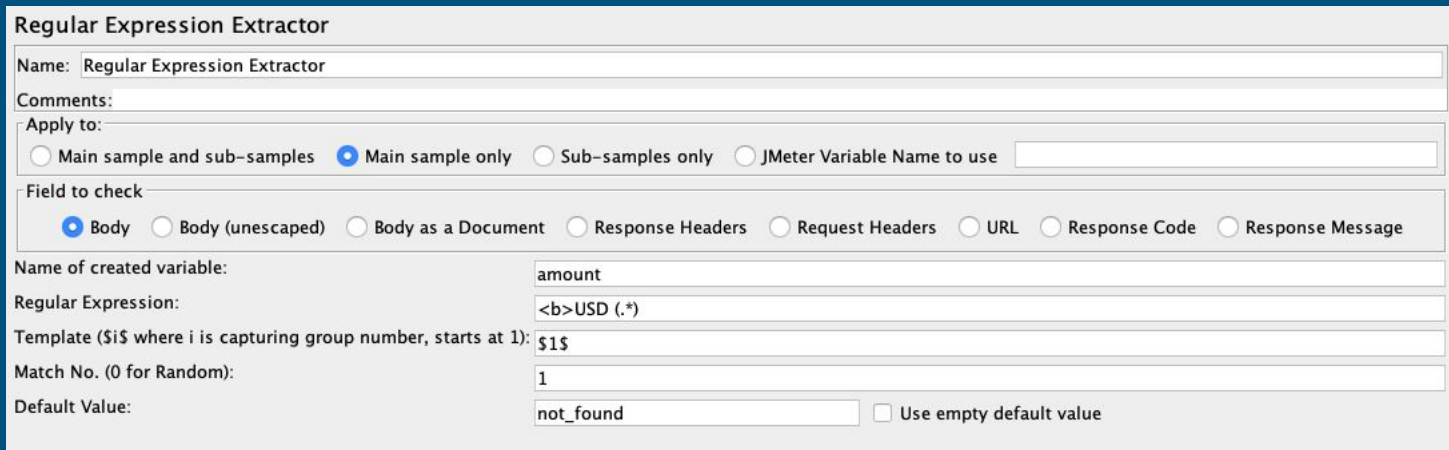
- Name:

What does the  
extractor serves.

- Comments:

Any description about this extractor.

- Apply to: Where do you want to extract data from.
- Field to check: The field where extractor applies on is in which component.



The screenshot shows the 'Regular Expression Extractor' configuration window. The 'Name' field is set to 'Regular Expression Extractor'. The 'Comments' field is empty. Under 'Apply to:', the 'Main sample only' radio button is selected. Under 'Field to check:', the 'Body' radio button is selected. The 'Name of created variable:' field is set to 'amount'. The 'Regular Expression:' field contains the pattern '<b>USD (.\*)'. The 'Template (\$i\$ where i is capturing group number, starts at 1):' field contains '\$1\$'. The 'Match No. (0 for Random):' field is set to '1'. The 'Default Value:' field is set to 'not\_found', and the 'Use empty default value' checkbox is unchecked.

Regular Expression Extractor	
Name:	Regular Expression Extractor
Comments:	
Apply to:	<input type="radio"/> Main sample and sub-samples <input checked="" type="radio"/> Main sample only <input type="radio"/> Sub-samples only <input type="radio"/> JMeter Variable Name to use
Field to check	<input checked="" type="radio"/> Body <input type="radio"/> Body (unescaped) <input type="radio"/> Body as a Document <input type="radio"/> Response Headers <input type="radio"/> Request Headers <input type="radio"/> URL <input type="radio"/> Response Code <input type="radio"/> Response Message
Name of created variable:	amount
Regular Expression:	<b>USD (.*)
Template (\$i\$ where i is capturing group number, starts at 1):	\$1\$
Match No. (0 for Random):	1
Default Value:	not_found <input type="checkbox"/> Use empty default value

# Regular Expression Extractor

---

## Attribute:

- Name of created variable: Reference variable, where we could call in any place.
- Regular Expression: The syntax of regular expression to use.
- Template: The capturing group number.
- Match no: 0 for random, -1 to retrieve as list, any others value for the position.
- Default value: where no result is returned, Jmeter will show this value.

# Regular Expression Extractor(continued)

---

## Showcase:

- Use Regular Extractor for extract the Id of character “Tyrion Lannister”.

## Exercise:

- Get the user Id and password after requesting access for <http://demo.guru99.com/> bank application.
- In <https://restool-sample-app.herokuapp.com/#!/characters>, create one character and try to find your added character.

# JSON Extractor

---

## Description:

JSON (JavaScript Object Notation) Extractor is used to extract the values from JSON response. Actually, JSON is a simple text which is used to exchange the information between the client and the server. It is written with the JavaScript object. Due to increase, the use of the REST APIs, the JSON is used as a primary data exchange format.

-> Same as XML which have XPath, JSON got its own browsing mechanism called JSON Path

# Json Path syntax - Basics - Notation

Tokens	Meaning	Example
\$	The root object or array	
. <i>property</i>	Select specific property in parent object	\$.name-> return property called 'name'
[ <i>index</i> ]	Select the n-element in the array	\$.[0] -> return first element in array
[ <i>'property'</i> ]	Select specific property in parent object(used this notation when property contains special characters)	\$.[0].['id'] -> return value of property called id of 1st element in array
.. <i>property</i>	Searches for the specified property name recursively and returns an array of all values with this property name.	\$.id -> return all values of property called id in array
*	Selects all elements in an object or an array, regardless of their names or indexes	\$.[*] -> return all values of all properties in all elements of array



# Json Path syntax - Basics - Notation(continued)

Tokens	Meaning	Example
<code>[start:end]</code> <code>[start:]</code>	Selects array elements from the <i>start</i> index and up to, but not including, <i>end</i> index. If <i>end</i> is omitted, selects all elements from <i>start</i> until the end of the array	<code>\$.[0:2].['id']</code> -> return element at position 0 and 1 of the array.
<code>[:end]</code>	Selects first n elements of the array	<code>\$.[:3]&gt;</code> return first 3 elements of array.
<code>[-end:]</code>	Select last n elements of the array	<code>\$.[-3:]</code> -> return last 3 element in array
@	Used in filter expressions to refer to the current node being processed.	
<code>[?(expression)]</code>	Selects all elements in an object or array that match the specified filter.	<code>\$.*[?(@.realName == 'John Bradley'    @.realName == 'Lena Headey')]</code> -> return element that have property realName to be equal to John Bradley or Lena Headey

# JSON Extractor

## Attribute:

- Name: What does

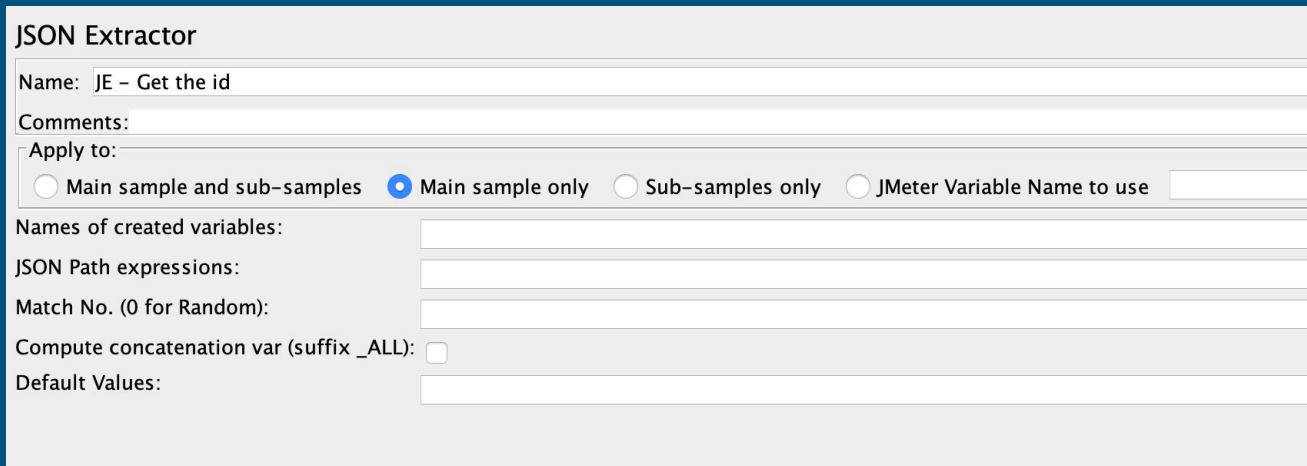
the extractor serves.

- Comments:

Any description about this

extractor.

- Apply to: Where do you want to extract data from.
- Field to check: The field where extractor applies on is in which component.



The screenshot shows the 'JSON Extractor' configuration window. It has a title bar 'JSON Extractor'. Below it, there are several fields: 'Name:' with the value 'JE - Get the id', 'Comments:' (empty), 'Apply to:' with four radio buttons: 'Main sample and sub-samples' (unselected), 'Main sample only' (selected), 'Sub-samples only' (unselected), and 'JMeter Variable Name to use' (unselected). Below these are four text fields: 'Names of created variables:', 'JSON Path expressions:', 'Match No. (0 for Random):', and 'Compute concatenation var (suffix \_ALL):' (with a checkbox). At the bottom is a 'Default Values:' field.

# JSON Extractor(continued)

---

## Attribute:

- Name of created variable: Reference variable, where we could call in any place.
- JSON Path Expression: The syntax of JSON Path to use.
- Match no: 0 for random, -1 to retrieve as list, any others value for the position.
- Compute concatenation var(suffix\_ALL): This option is marked when many results are found. Then JMeter concatenates all the values using the ',' separator and stores that value in a variable named <variable name>\_ALL
- Default value: where no result is returned, Jmeter will show this value.

# JSON Extractor(continued)

---

## Showcase:

- Reuse the expression of Json Path for demo.

## Exercise:

- Get the name of character who have current location = Kings Landing.
- In <https://restool-sample-app.herokuapp.com/#/employees>, get the person who is fired.
- In <https://restool-sample-app.herokuapp.com/#/deads>, find the person who is dead by get poisoned.

# Boundary Extractor

## Attribute:

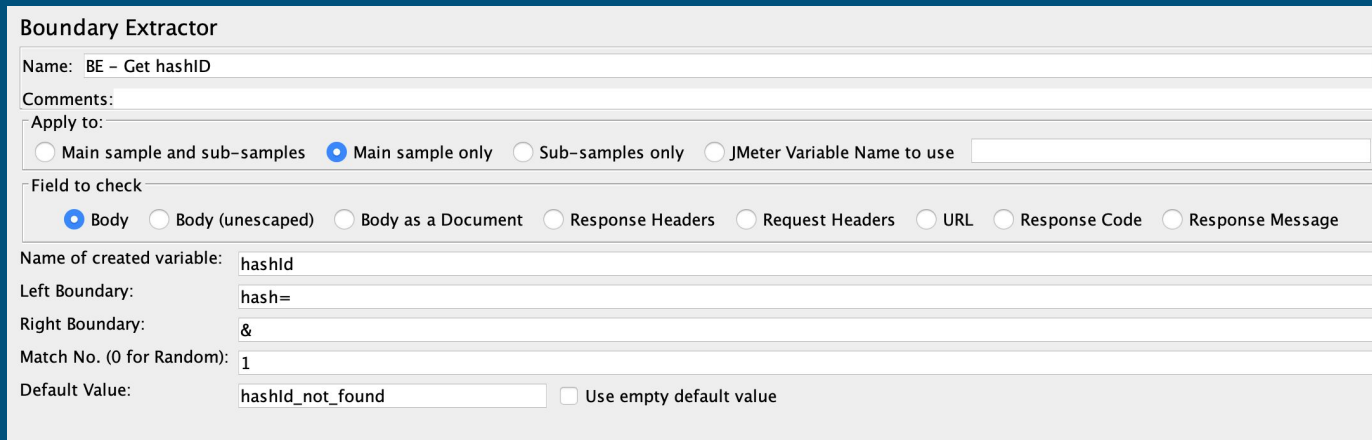
- Name: What does the extractor serves.

- Comments:

Any description about this

extractor.

- Apply to: Where do you want to extract data from.
- Field to check: The field where extractor applies on is in which component.



The screenshot shows the 'Boundary Extractor' configuration window. It includes fields for 'Name' (BE - Get hashID), 'Comments', 'Apply to' (radio buttons for 'Main sample and sub-samples', 'Main sample only' (selected), 'Sub-samples only', and 'JMeter Variable Name to use'), 'Field to check' (radio buttons for 'Body' (selected), 'Body (unescaped)', 'Body as a Document', 'Response Headers', 'Request Headers', 'URL', 'Response Code', and 'Response Message'), 'Name of created variable' (hashId), 'Left Boundary' (hash=), 'Right Boundary' (&), 'Match No. (0 for Random)' (1), and 'Default Value' (hashId\_not\_found) with a checkbox for 'Use empty default value'.

# Boundary Extractor(continued)

---

## Attribute:

- Name of created variable: Reference variable, where we could call in any place.
- Left boundary: String or text before the value to be extracted.
- Right boundary: String or text after the value to be extracted.
- Match no: 0 for random, -1 to retrieve as list, any others value for the position.
- Default value: where no result is returned, Jmeter will show this value.

# Boundary Extractor(continued)

---

## Showcase:

- Use Boundary Extractor to extract hashID.

## Exercise:

- Use Boundary Extractor to extract character name of <https://restool-sample-app.herokuapp.com/#/characters>.
- Apply Boundary Extractor to all previous exercises.

# JSR223 Extractor

---

## Description:

- Can apply to extract value as well.
- Since it is written by JSR223 scripting language so the performance is the best.



# JSR223 Extractor

---

## Demo:

- Show later

## Exercise:

- Use JSR223 Extractor to get id of character called Tyron Lanister.
- Use it in other exercise.