Continuation Class

Use the `Continuation` class to make callouts asynchronously to a SOAP or REST Web service.

Namespace

System

Example

For a code example, see Make Long-Running Callouts from a Visualforce Page.

- **Continuation Constructors**
- **Continuation Properties**
- **Continuation Methods**

Continuation Constructors

The following are constructors for `Continuation`.

- **Continuation(timeout)**
  Creates an instance of the `Continuation` class by using the specified timeout in seconds. The timeout maximum is 120 seconds.

Continuation(timeout)

Creates an instance of the `Continuation` class by using the specified timeout in seconds. The timeout maximum is 120 seconds.

Signature

```
public Continuation(Integer timeout)
```

Parameters

timeout

Type: Integer

The timeout for this continuation in seconds.

Continuation Properties

The following are properties for `Continuation`.

- **continuationMethod**
  The name of the callback method that is called after the callout response returns.
- **timeout**
  The timeout of the continuation in seconds. Maximum: 120 seconds.
- **state**
  Data that is stored in this continuation and that can be retrieved after the callout is finished and the callback method is invoked.

```
public String continuationMethod {get; set;}
```

Property Value

Type: String

Usage

> **Note**
>
> If the `continuationMethod` property is not set for a Continuation, the same action method that made the asynchronous callout is called again when the callout response returns.

timeout

The timeout of the continuation in seconds. Maximum: 120 seconds.

Signature

```
public Integer timeout {get; set;}
```

Property Value

Type: Integer

state

Data that is stored in this continuation and that can be retrieved after the callout is finished and the callback method is invoked.

Signature

```
public Object state {get; set;}
```

Property Value

Type: Object

Example

This example shows how to save state information for a continuation in a controller.

```
// Declare inner class to hold state info
private class StateInfo {
    String msg { get; set; }
    List<String> urls  { get; set; }
    StateInfo(String msg, List<String> urls) {
        this.msg = msg;
        this.urls = urls;
    }
}

// Then in the action method, set state for the continuation
continuationInstance.state = new StateInfo('Some state data', urls);
```

Continuation Methods

The following are methods for `Continuation`.

pairs.

- getResponse(requestLabel)
  Returns the response for the request that corresponds to the specified label.

addHttpRequest(request)

Adds the HTTP request for the callout that is associated with this continuation.

Signature

```
public String addHttpRequest(System.HttpRequest request)
```

Parameters

request

Type: HttpRequest

The HTTP request to be sent to the external service by this continuation.

Return Value

Type: String

A unique label that identifies the HTTP request that is associated with this continuation. This label is used in the map that getRequests() returns to identify individual requests in a continuation.

Usage

You can add up tothree requests to a continuation.

> **Note**
>
> The timeout that is set in each passed-in request is ignored. Only the global timeout maximum of 120 seconds applies for a continuation.

getRequests()

Returns all labels and requests that are associated with this continuation as key-value pairs.

Signature

```
public Map<String,System.HttpRequest> getRequests()
```

Return Value

Type: Map<String,HttpRequest>

A map of all requests that are associated with this continuation. The map key is the request label, and the map value is the corresponding HTTP request.

getResponse(requestLabel)

Returns the response for the request that corresponds to the specified label.

Signature

```
public static HttpResponse getResponse(String requestLabel)
```

Parameters

requestLabel

Type: HttpResponse

Usage

The status code is returned in the HttpResponse object and can be obtained by calling `getStatusCode()` on the response. A status code of 200 indicates that the request was successful. Other status code values indicate the type of problem that was encountered.

Sample of Error Status Codes

When a problem occurs with the response, some possible status code values are:

- 2000: The timeout was reached, and the server didn't get a chance to respond.
- 2001: There was a connection failure.
- 2002: Exceptions occurred.
- 2003: The response hasn't arrived (which also means that the Apex asynchronous callout framework hasn't resumed).
- 2004: The response size is too large (greater than 1 MB).