# 3 Implicit Parameters

### 3.1.2 About the :body\_json parameter

The :body\_json implicit parameter can be used in Resource Handlers to receive multipart/form-data POST requests where multiple files are expected to be processed *and* form data should be formatted as a JSON string (*as a CLOB data type*)[[1]](#footnote-1).

When the:body\_json implicit parameter is included for Resource Handlers that process multiple files, the :body\_json implicit parameter must be invoked. The :body\_json parameter can be invoked in various ways, such as:

* The DBMS\_OUTPUT package such as dbms\_output.put\_line(:body\_json);
* The hypertext procedures (htp) and functions (htf) packages, such as in htp.print(:body\_json);
* Assigning the :body\_json implicit parameter as variable, e.g. l\_body\_json := :body\_json;

**NOTE:** There is no specific requirement to assign :body\_json to a local variable. Similarly, there is no requirement to re-use the local variable, should you choose to assign one.

#### Example

A table (DEMO\_TABLE) has been created with the following attributes:

CREATE TABLE ORDSDEMO.DEMO\_TABLE   
 (   
 ID NUMBER (\*,0) GENERATED BY DEFAULT AS IDENTITY   
 ( START WITH 1 CACHE 20 ) NOT NULL ,   
 FILE\_NAME VARCHAR2 (200) ,   
 FILE\_BODY BLOB ,   
 CONTENT\_TYPE VARCHAR2 (200) ,   
 FILE\_VISIBILITY VARCHAR2 (10) ,   
 SUBMITTED\_BY VARCHAR2 (200) ,   
 SUBMITTED\_ON TIMESTAMP DEFAULT systimestamp   
 )   
 TABLESPACE DATA   
 LOGGING   
;

**NOTE:** Columns such as FILE\_VISIBILITY, SUBMITTED\_BY, and SUBMITTED\_ON are for *demonstration purposes only*. They are not required.

An ORDS Endpoint has been created (with the below Resource Handler code).

* The endpoint expects multiple files and form data *in a JSON format* (i.e., the use of the :body\_json implicit parameter).
* The ORDS.BODY\_FILE\_COUNT function will be used to count the total files in the POST request.
* The ORDS.GET\_BODY\_FILE procedure will be used to store, in session, the files.

The following code example then performs an INSERT on the DEMO\_TABLE and relies upon various HTP procedures to “print” the results to a user, client, or application.

DECLARE  
 L\_PARAMETER\_NAME VARCHAR2(4000);  
 L\_FILE\_NAME VARCHAR2(4000);  
 L\_CONTENT\_TYPE VARCHAR2(200);  
 L\_FILE\_BODY BLOB;  
 L\_FILE\_VISIBILITY CLOB;  
 L\_SUBMITTED\_BY CLOB;  
BEGIN  
 L\_SUBMITTED\_BY := JSON\_VALUE(:BODY\_JSON, '$.submitted\_by');  
 L\_FILE\_VISIBILITY := JSON\_VALUE(:BODY\_JSON, '$.file\_visibility');  
 HTP.PARAGRAPH;  
 HTP.PRINT('Submitted By: ' || L\_SUBMITTED\_BY);  
 HTP.BR;  
 HTP.PARAGRAPH;  
 HTP.PRINT('File visibility status: ' || L\_FILE\_VISIBILITY);  
 HTP.BR;  
 FOR I IN 1..ORDS.BODY\_FILE\_COUNT LOOP  
 ORDS.GET\_BODY\_FILE(  
 P\_FILE\_INDEX => I,  
 P\_PARAMETER\_NAME => L\_PARAMETER\_NAME,  
 P\_FILE\_NAME => L\_FILE\_NAME,  
 P\_CONTENT\_TYPE => L\_CONTENT\_TYPE,  
 P\_FILE\_BLOB => L\_FILE\_BODY  
 );  
 INSERT INTO DEMO\_TABLE (  
 FILE\_NAME,  
 FILE\_BODY,  
 CONTENT\_TYPE,  
 FILE\_VISIBILITY,  
 SUBMITTED\_BY  
 ) VALUES ( L\_FILE\_NAME,  
 L\_FILE\_BODY,  
 L\_CONTENT\_TYPE,  
 L\_FILE\_VISIBILITY,  
 L\_SUBMITTED\_BY );  
  
 HTP.PARAGRAPH;  
 HTP.PRINT('Inserted File: ' || L\_FILE\_NAME);  
 END LOOP;  
END;

To test this :body\_json implicit parameter a curl command such as the one below may be used:

curl --location 'https://localhost:8080/ords/ordsdemo/demo\_api/demo' \  
--form 'file\_one=@"/Users/file\_one.txt"' \  
--form 'file\_two=@"/Users/ile\_two.txt"' \  
--form 'submitted\_by="chris"' \  
--form 'file\_visibility="public"'

Accordingly, a client may respond with the following:

<p>Submitted By: chris  
<br />  
<p>File visibility status: public  
<br />  
<p>Inserted File: demo-3.sql  
<p>Inserted File: demo-2.sql

Along with an update to target DEMO\_TABLE:

| ID FILE\_NAME | FILE\_BODY | CONTENT\_TYPE | FILE\_VISIBILITY | SUBMITTED\_BY | SUBMITTED\_ON |
| --- | --- | --- | --- | --- | --- |
| 144 | demo-3.sql | REVDTEFS… | application/x-sql | public | chris |
| 145 | demo-2.sql | Q1JFQVRF… | application/x-sql | public | chris |

1. Although you can store JSON in the Oracle database as JSON, VARCHAR2, CLOB, and BLOB, ORDS uses the CLOB data type, to ensure backward compatibility with earlier releases of the Oracle database. [↑](#footnote-ref-1)