Dre Specifica Tecnica

Versione 0.5

24 marzo 2014

Versione	0.5
Uso	Esterno
Redazione	Ferlin Alessandro
Distribuzione	Nextep

Storico delle modifiche

Versione	***Data***	***Autore***	***Modifica Effettuata***
0.5	24-03-2014	Ferlin Alessandro	*Modifica capitolo 2 relativo all'email, ora sostituito con il token (email in base64)*
0.4	06-02-2014	Ferlin Alessandro	*Ampliato capitolo Risposta in caso di Errore*
0.3	09-01-2014	Ferlin Alessandro	*Aggiunti capitoli Risposta in caso di Errore e Casi base*
0.1	10-12-2013	Ferlin Alessandro	*Iniziato il documento con una prima stesura, descritti input e output servizio di raccomandazione*

Indice

1 Introduzione

1.1 Requisiti

- Includere i seguenti metatag, l'indirizzo `193.28.95.209` è di demo, in futuro sarà modificato

```
1  <script src="http://code.jquery.com/jquery-1.8.2.min.js" type="text/
       javascript" charset="utf-8"></script>
2   <script src="http://193.28.95.209/assets/javascripts/dre.js" type="
       text/javascript" charset="utf-8"></script>
```

2 API javascript

Di seguito vengono descritti i metodi per l'interfacciamento al servizio di raccomandazione (d'ora in poi `Dre`).

2.1 Input

Tale `JSON` viene utilizzato per richiedere una raccomandazione al servizio `Dre`

- `Parametri`:
  - `{user}` Campo opzionale, contiene l' `email` dell'utente codificata in base64, tale campo corrisponde al campo `token`, il campo `email` in chiaro è `deprecato`.
  - `{tags}` Campo opzionale, lista di tag.
  - `{feedback}` Campo opzionale, tale campo contiene l'id della raccomandazione andata a buon fine, ovvero l'identificativo di una raccomandazione precedentemente effettuata che ha attirato l'attenzione dell'utente, che ha portato l'utente a cliccare sul Tail.
- `Esempi`:
  - `Esempio con tutti i parametri`

```
1     {
2      "user" : {
3            "token": "cHJvdmFAZW1haWwuY29t"
4           },
5      "tags": [
6           {"tag": "categoria1:attr11"},
7           {"tag": "categoria1:attr12"},
8           {"tag": "categoria2:attr21"}
9          ],
10     "feedback": {
11           "idR": "1383039903274"
12          },
13    }
```

- Esempio senza utente

```
{
 "tags": [
     {"tag": "categoria1:attr11"},
     {"tag": "categoria1:attr12"},
     {"tag": "categoria2:attr21"}
    ],
 "feedback": {
       "idR": "1383039903274"
     },
}
```

- Esempio senza tags

```
{
 "user" : {
      "token": "cHJvdmFAZW1haWwuY29t",
     },
 "feedback": {
       "idR": "1383039903274"
     },
}
```

- Esempio senza nessun parametro

```
{
}
```

2.2 Output

A raccomandazione completata viene restituito un JSON formato come segue:

- Parametri:
  - {recommendation} Contiene la raccomandazione, composta da un id (idR) e da una lista di tags (tags).
  - {user} Contiene informazioni riguardo l'utente che ha richiesto la raccomandazione, è composto da un id (id) da un token (token, campo opzionale), il token rappresenta l'email codificata in base64.
- Esempi:
  - Esempio con tutti i parametri

```
{
 "recommendation": {
          "idR": "50931285-d736-4d58-ac94-6d4fca37733f",
          "tags": [
```

```
5                    { "tag": "categoria1:attr11" },
6                    { "tag": "categoria1:attr12" },
7                    { "tag": "categoria2:attr21" }
8                   ]
9               },
10       "user": {
11            "id": "d2898447-b2d9-4bc0-9744-78505f5caaa2",
12            "token": "cHJvdmFAZW1haWwuY29t"
13           }
14      }
```

2.3 Modalità d'utilizzo oggetto `Dre`

Nel javascript `dre.js` importato come descritto nel paragrafo 2.1 è presente un oggetto denominato `dre` con un metodo chiamato `advise`.

Con tale metodo è possibile richiedere una raccomandazione al servizio `Dre`, di seguito ne viene descritto l'utilizzo:

2.3.1 Metodo advise

- `Parametri`:
  - `{jsonData}` un oggetto javascript rappresentante l'input della richiesta.
  - `{callbackSuccess}` Una funzione chiamata quando la raccomandazione è pronta, tale callback deve avere come parametri di ingresso un json, tale json conterrà la raccomandazione
- `Esempi`:
  - `Esempio di utilizzo`

```
1       // genero il json di input per la raccomandazione
2       var json = $.parseJSON('{"tags": [{"tag":"categoria1:attr11
            "}, {"tag":"categoria1:attr12"}]}');
3
4       // genero la funzione di callback
5       // che fara' qualcosa con la raccomandazione
6       var callback = function processData(data) {
7        // faccio qualcosa con il risultato
8        console.log(JSON.stringify(data));
9       };
10
11      // richiedo una raccomandazione al servizio
12      // al completamento della raccomandazione
13      // verra' eseguita la callback
14      dre.advise(json, callback);
```

3 Risposta in caso di Errore

Di seguito viene descritto il comportamento del sistema `Dre` in caso di `Json` malformato o non completo.

3.1 Errori con `Json` valido

Il `Json` risulta valido (ovvero privo di errori di sintassi) ma il contenuto non è accettato dal sistema `Dre` perché non completo.

Il questi casi gli errori vengono descritti tramite un `Json` con la seguente struttura: `Struttura Json errori`

```
{
  "obj.xxxx.yyy": [{
    "msg": "...",
    "args": ["..."]
  }],
  ...
}
```

`Esempio errore`:

- `Input con feedback e tag malformati`

```
{
 "tags":[{"tag":"ab::"}],
 "feedback": {}
}
```

- `Risposta del sistema`

```
{
  "obj.tags[0].tag": [{
    "msg": "validate.error.unexpected.value",
    "args": ["tag format isn't correct, maybe missing colon?"]
  }],
  "obj.feedback.idR": [{
    "msg": "error.path.missing",
    "args": []
  }]
}
```

3.2 Errori con `Json` non valido

Il `Json` risulta invalido ovvero contiene almeno un errore di sintassi (es.: una graffa non chiusa).

In questo caso il sistema risponde con un `Json` avente la seguente struttura: `Struttura Json errori Json non valido`

```
{
    "status": ...,
    "errorCode": ...,
    "field": "...",
    "message": "...",
    "developerMessage": "messaggio per lo sviluppatore, sara' omesso
        nelle versioni successive"
}
```

Esempio errore:

- Input con feedback e tag malformati

```
    {
     "tags:[{"tag":"ab::ab"}]
    }
```

- Risposta del sistema

```
    {
        "status": 500,
        "errorCode": 10000,
        "field": "",
        "message": "Internal server error",
        "developerMessage": "Execution exception[[JsonParseException:
            Unexpected character ('t' (code 116)): was expecting a
            colon to separate field name and value\n at [Source: java.
            io.StringReader@4e245d3e; line: 2, column: 16]]]"
    }
```

3.3 Errori con Json non valido o valido ma url non riconosciuta

Il Json può essere valido o meno ma la url risulta non corretta.

In questo caso il sistema risponde con un Json avente la seguente struttura: Struttura Json errori Json non valido

```
{
    "status": ...,
    "errorCode": ...,
    "field": "...",
    "message": "...",
    "developerMessage": "messaggio per lo sviluppatore, sara' omesso
        nelle versioni successive"
}
```

Esempio errore:

- Input con feedback e tag malformati

```
In questo caso il sistema non valuta il contenuto della richiesta,
     quindi non e' necessario un esempio
Per tale errore e' sufficiente effettuare una richiesta ad una url
     non riconosciuta
```

- `Risposta del sistema`

```
{
    "status": 404,
    "errorCode": 10000,
    "field": "",
    "message": "Not Found",
    "developerMessage": "Error performing operation"
}
```

4 Casi base

Di seguito vengono descritti i casi nei quali il sistema `Dre` assumerà un comportamento differente dal caso comune.

4.1 Prima richiesta di un utente

In questo caso l'utente è alla sua prima richiesta di raccomandazione, si ha quindi come precondizione l'assenza di tag nel profilo dell'utente.

A tal punto sono possibili due varianti:

- `{Input senza tags:}` Non avendo nessun dato per l'utente che effettua la richiesta, il sistema restituirà i 5 tag con rating assoluto più alto.
- `{Input con almeno un tag:}` Il sistema effettua la raccomandazione sui tag in input. Il numero di tag contenuti nella raccomandazione variano in base allo stato del sistema, almeno un tag sarà comunque restituito.

4.2 Considerazione

Le `home page` non dovrebbero contenere tag perché la pagina non deriva dalla scelta dell'utente. All'interno della home page converrebbe inserire dei tail preimpostati da lago e assieme a questi dei tail alimentati dalla raccomandazione

Nel caso di un utente loggato le considerazioni sulla home page devono essere rivalutate in base alle strategie di profilazione

- `{es:}` un architetto potrebbe avere una home page completamente diversa da quella di un cliente.