Movie Reviews Nokia Asha Web App

Movie Reviews is a Series 40 Web app which features latest movies, shows details (release date, cast, synopsis) and reviews for the selected movies, and also allows users to search for other movies from the Rotten Tomatoes movie database. This article illustrates how the app is constructed.

10 Feb 2013

Introduction

The Rotten Tomatoes API gives access to wealth of movie information, allowing anyone to build applications and widgets enriched with Rotten Tomatoes data. The API allows to get the current box office movies, new releases, upcoming movies and search for movies and retrieve detailed movie information, like cast, directors, and movie posters.

The MovieReviews application, uses three of the Rotten Tomatoes APIs:

- **In Theaters**: /api/public/v1.0/lists/movies/in_theaters.json : Retrieves movies currently in theaters.
- **Movies Reviews**: /api/public/v1.0/movies/:id/reviews.json : Retrieves the reviews for a movie. Results are paginated if they go past the specified page limit.
- **Movies Search**: /api/public/v1.0/movies.json : The movies search endpoint for plain text queries. Let's you search for movies!

To access the Rotten Tomatoes API, you would need to register your web app at their developer portal: http://developer.rottentomatoes.com. Once registered, you would get an API key for your application which would be used in the HTTP requests to the API.

User Interface

The application has a Tab user interface with three tabs: Latest, Search and About.

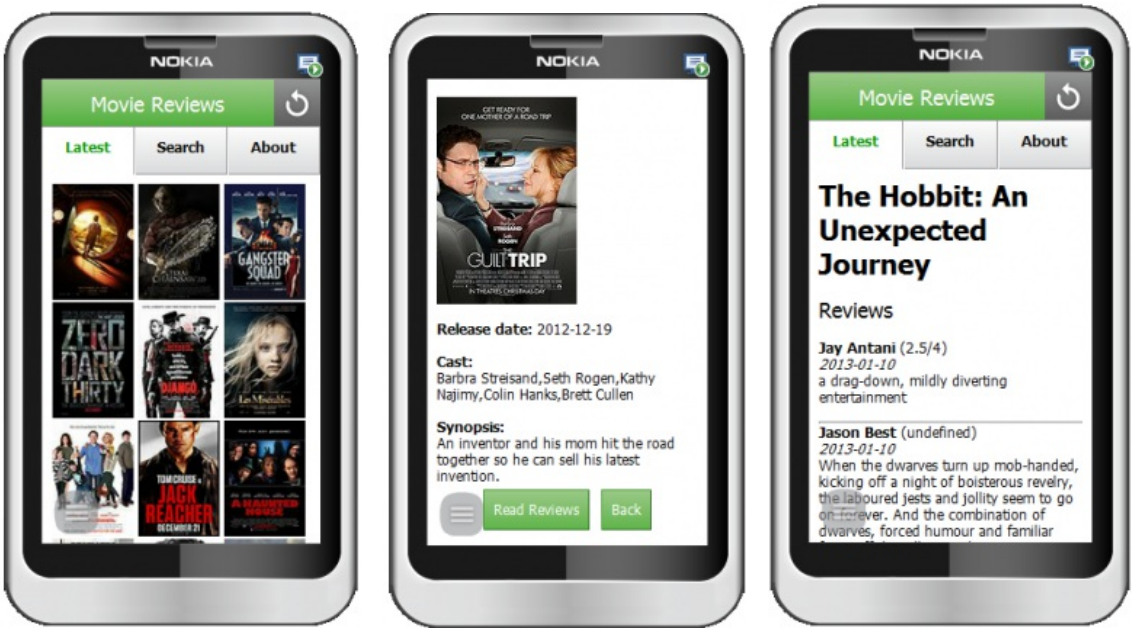
Each Tab control is a collection of tags. For example, `Tab1` contains three main tags shown below,

```
<div id="tab_1_content" class="ui-show">

 <div id="thumbnailPage" style="text-align: center" class="hide" ></div>
 <div id="detailsPage" class="hide"></div>
 <div id="reviewPage" class="hide"></div>

</div>
```

The first Tab displays a grid of latest movies' thumbnail posters. When the user selects a movie from the thumbnail grid, the app shows the movie details like release date, cast, movie posters and a short summary of the movie. On this screen the user gets an option to read the reviews for the movie as well.

The view switching within the tabs is done by using the 'hide' and 'show' CSS classes. For example, the mwl function `mwl.switchClass(targetNode, removeClass, addClass)` is used to show or hide a particular tag content. The functions provided by mwl are local and do not require a server round trip, hence the view is switched instantly. If the UI is implemented using a table, `mwl.setGroupTarget` can be used.

- thumbnailPage: shows the thumbnail posters of the latest movies,
- detailsPage: shows the details of the movies and
- reviewPage: shows a list of reviews for the selected movie



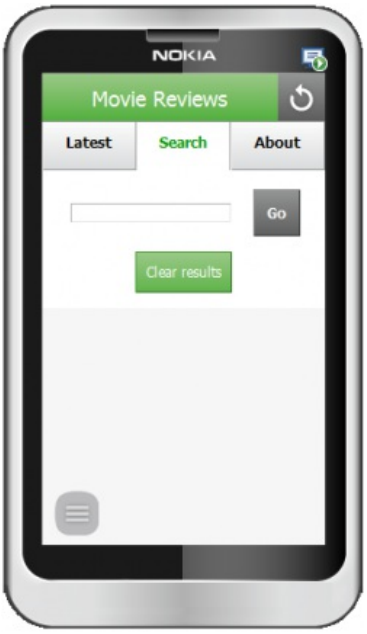
Similarly, the second Tab contains the tags shown in below,

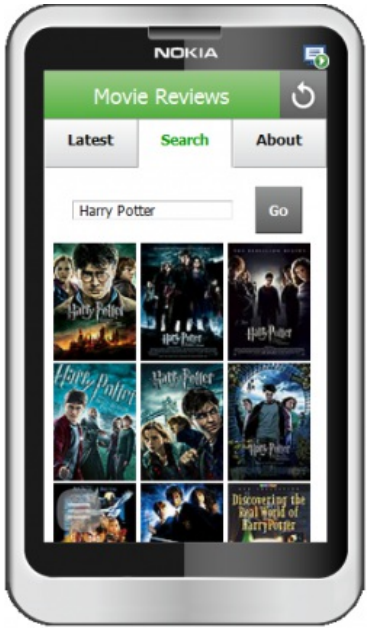
(C) Copyright Nokia 2013. All rights reserved.

```
<div id="tab_2_content" class="ui-hide">

<div id="searchBar" class="show">

 <!-- text-field-with-button -->
 <div class="ui-fieldset-text-button">
  <table class="header_table" border="0" cellspacing="0"
   cellpadding="0">
   <tr>
    <td><input type="text" id = "lineEdit"
     class="ui-text-input-with-adj-button" />
    </td>
    <td><input name="button" type="submit"
     class="ui-button ui-fixed-40px" id="button" value="Go"
     onclick="searchMovies();"/>
    </td>
   </tr>
  </table>

 </div>
</div>

<div id="searchResults" class="show"></div>

<div id="searchDetailsPage" class="hide"></div>

</div>
```

The Search Tab includes the following tags,

- searchBar: with a lineEdit and a Go button,
- searchResults: shows the search results for the movie queried,
- searchDetailsPage: shows the details of the movie, when selected from the search results.



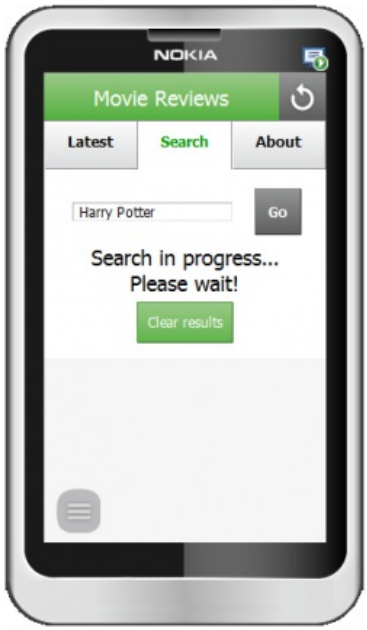


The search tab also includes a clear button, which clears the search results for a new query.

```
<div style="text-align: center;">
 <input type="button" name="clearButton" class="ui-button" value="Clear results" onclick="mwl.switchClass('#searchResults', 'show',
</div>
```

Both the tabs, Latest and Search, contain a tag which is used as a loading screen, while querying the API or rendering the data.

(C) Copyright Nokia 2013. All rights reserved.




```
<div id="loadingPage" style="vertical-align: center; text-align: center;" class="show">
     <h2>Loading data...<br>Please wait!</h2>
 </div>
```

Implementation

The implementation is contained in the `main.js` file. The `init_script()` function is called when the html `<body>` is loaded,

```
<body onload="init_script();">
```

In-Theater Movies

The `init_script()` function handles the API call for the first Tab i.e. in-theater movies, by calling the function `GetJSONData(url)`.

The `GetJSONData(url)` function contains a HTTP request and response parser and handler. The function is used for all the other API calls from the application. The `responseHandler` function distinguishes the type of API calls based on the global enum variable `'queryType'`.

```
var queryTypeEnum = {"latestMovie" : 0, "movieReview" : 1, "searchMovies" : 2};
var queryType = queryTypeEnum.latestMovie;
```

```
var obj = eval('(' + json + ')');
// check if queryType is latestMovie
if (queryType == queryTypeEnum.latestMovie)
{
 var movies = obj.movies;
 // store the fetched moviews
 fetchedMovies = movies;

 // generate html markup for the thumbnail page
 var markup = "";

 for(i=0; i < movies.length; i++){
  markup += this.generateHTMLMarkup(i, movies[i]);
 }

 document.getElementById("thumbnailPage").innerHTML = markup;

 // show thumbnailPage, hide loadingPage
 mwl.switchClass('#thumbnailPage', 'hide', 'show');
 mwl.switchClass('#loadingPage', 'show', 'hide');
}
```

`fetchedMovies` holds all the instances of the movie items from the JSON response. A JSON object of a movie looks like below,


```
{
 "id": "770675766",
 "title": "The Hobbit: An Unexpected Journey",
 "year": 2012,
 "mpaa_rating": "PG-13",
 "runtime": 170,
 "critics_consensus": "Peter Jackson's return to Middle-earth is an earnest, visually resplendent trip, but the film's deliberate pa
 "release_dates": {
  "theater": "2012-12-14"
 },
 "ratings": {
  "critics_rating": "Fresh",
  "critics_score": 65,
  "audience_rating": "Upright",
  "audience_score": 81
 },
 "synopsis": "The Hobbit: An Unexpected Journey follows title character Bilbo Baggins, who is swept into an epic quest to reclaim th
 "posters": {
  "thumbnail": "http://content9.flixster.com/movie/11/16/28/11162899_mob.jpg",
```


(C) Copyright Nokia 2013. All rights reserved.

```
   "profile": "http://content9.flixster.com/movie/11/16/28/11162899_pro.jpg",
   "detailed": "http://content9.flixster.com/movie/11/16/28/11162899_det.jpg",
   "original": "http://content9.flixster.com/movie/11/16/28/11162899_ori.jpg"
  },
  "abridged_cast": [
   {
    "name": "Ian McKellen",
    "id": "162653241",
    "characters": [
     "Gandalf"
    ]
   },
   {
    "name": "Martin Freeman",
    "id": "162652296",
    "characters": [
     "Bilbo",
     "Bilbo Baggins"
    ]
   }
  ],
  "alternate_ids": {
   "imdb": "0903624"
  },
  "links": {
   "self": "http://api.rottentomatoes.com/api/public/v1.0/movies/770675766.json",
   "alternate": "http://www.rottentomatoes.com/m/the_hobbit_an_unexpected_journey/",
   "cast": "http://api.rottentomatoes.com/api/public/v1.0/movies/770675766/cast.json",
   "clips": "http://api.rottentomatoes.com/api/public/v1.0/movies/770675766/clips.json",
   "reviews": "http://api.rottentomatoes.com/api/public/v1.0/movies/770675766/reviews.json",
   "similar": "http://api.rottentomatoes.com/api/public/v1.0/movies/770675766/similar.json"
  }
 }
```


In our application we use some of attributes of the above object. For example, when a movie thumbnail is clicked on,

```
function showMovie(id) {

 // Hides the thumbnailPage div tag
 mwl.switchClass('#thumbnailPage', 'show', 'hide');

 // get the json object for the selected movie
 selected_movie = getMovieObject(id);

 // create an html markup for details page
 var details_page_text = "";

 details_page_text += "<h2>" + selected_movie['title']+ "</h2>";
 details_page_text += "<br/>";
 details_page_text += "<img src=\""+ selected_movie['posters']['profile']+ "\"/>";
 details_page_text += "<br/><br/>";
 details_page_text +="<b>Release date:</b> " + selected_movie['release_dates']['theater'];
 details_page_text += "<br/>";
 details_page_text += "<br/>";

 var cast = selected_movie['abridged_cast'];
 var cast_string = "";
 for(i=0; i < cast.length; i++){
  if (i===cast.length-1)
  {
   cast_string += cast[i]['name'];
  }
  else{
   cast_string += cast[i]['name'] + ",";
  }

 }
 details_page_text += "<b>Cast: </b><br>" + cast_string;
 details_page_text += "<br/>";
 details_page_text += "<br/>";
 details_page_text += "<b>Synopsis: </b><br>" + selected_movie['synopsis'];
 details_page_text += "<br/>";

 currentMovieId = selected_movie['id'];
 currentMovieTitle =selected_movie['title'];

 // add a button for Read Reviews
 details_page_text += "<div style=\"text-align:center\"> <input type=\"button\" name=\"reviewButton\" class=\"ui-button\" value=\"Re

 details_page_text += "<input type=\"button\" name=\"backButton\" class=\"ui-button\" value=\"Back\" onclick=\"mwl.switchClass('#thu
 document.getElementById("detailsPage").innerHTML = details_page_text;

 // Makes the detailsPage div tag visible
 mwl.switchClass('#detailsPage', 'hide', 'show');
}
```

The `getMovieObject(id)` function is used to get the movie object for the given id based on the query type.

```
function getMovieObject(id) {

 var movie_sets;
 if (queryType == queryTypeEnum.latestMovie)
  {
  // if the query type is latestMovie, use global variable fetechedMovies
  movie_sets = fetchedMovies;
  }
 else if (queryType == queryTypeEnum.searchMovies)
  {
  // if the query type is searchMovies, use global variable searchResultMovies
  movie_sets = searchResultMovies;
  }

 for (var i=0; i<movie_sets.length; i++)
 {
  if (id === movie_sets[i]['id']){

   return movie_sets[i];
  }
 }
 // return -1 if id not found
 return -1;
}
```


(C) Copyright Nokia 2013. All rights reserved.

Movie Reviews

When using the API call for movie reviews, the `queryType` variable is set to `queryTypeEnum.movieReview`. In this case the implementation of the response handler function, would be the following.

```
var obj = eval('(' + json + ')');

// check if the query is a movieReview query
else if (queryType== queryTypeEnum.movieReview)
{

 var reviews = obj.reviews;
 // store the fetched reviewed
 fetchedReviews = reviews;

 // generate html markup for the review page
 var markup = "";
 markup = "<h1>" + currentMovieTitle + "</h1><br>";
 markup += "<h2>Reviews</h2><br>"
 for(i=0; i < reviews.length; i++){
  markup += "<b>" + reviews[i]['critic'] + "</b> ("  + reviews[i]['original_score'] +")<br>"
      +"<i>" + reviews[i]['date']+ "</i><br>"
      + reviews[i]['quote']+ "<br><br><hr>";
 }
 markup += "<input type=\"button\" class=\"ui-button\" value=\"Back to Movie\" onclick=\"mwl.switchClass('#reviewPage', 'show', 'hid

 document.getElementById("reviewPage").innerHTML = markup;

 // show the reviewPage, hide the thumbnailPage & detailsPage
 mwl.switchClass('#thumbnailPage', 'show', 'hide');
 mwl.switchClass('#detailsPage', 'show', 'hide');
 mwl.switchClass('#reviewPage', 'hide', 'show');
}
```

Here is a sample movie review JSON object,

```
 {
 "critic": "Joe Baltake",
 "date": "2010-07-27",
 "freshness": "fresh",
 "publication": "Passionate Moviegoer",
 "quote": "'Toy Story 3': Alternately affecting, hilarious and heartbreaking and the most original prison-escape movie ever made",
 "links": {
  "review": "http://thepassionatemoviegoer.blogspot.com/2010/07/perfectimperfect.html"
 }
  }
```

Search Movies

The Movie search API is called when the Go button is clicked in the Search Tab. The `queryType` is set to `queryTypeEnum.searchMovies`, the `responseHandler` implementation would be,

```
// check if query type is searchMovies
else if (queryType== queryTypeEnum.searchMovies)
{
 var movies = obj.movies;
 // store search results
 searchResultMovies = movies;

 // generate html markup for the search results page
 var markup = "";

 for(i=0; i < movies.length; i++){
  markup += this.generateHTMLMarkup(i, movies[i]);
 }

 document.getElementById("searchResults").innerHTML = markup;

 // hide searchLoadingPage, show searchResults div
 mwl.switchClass('#searchLoadingPage', 'show', 'hide');
 mwl.switchClass('#searchResults', 'hide', 'show');

}
```

Summary

As of now, the Movie Reviews application uses several attributes provided by the Rotten Tomatoes API. The plan is to add more attributes and make the user interface richer. To track the progress of the project please check the project at Movie Reviews.

Source code

Movie Reviews application is available here.

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.