Api Documentation

Overview

Api Documentation

Version information

Version : 1.0

License information

License : Apache 2.0
License URL : http://www.apache.org/licenses/LICENSE-2.0
Terms of service : urn:tos

URI scheme

Host : localhost:8080
BasePath : /

Tags

- admin-controller : Admin Controller
- basic-error-controller : Basic Error Controller
- list-controller : List Controller
- movie-api-controller : Movie API Controller
- movie-log-controller : Movie Log Controller

Paths

addEntryToList

```
POST /addEntry/{listId}
```

Description

Adds a movie entry to the specified list

Parameters

Type	Name	Description	Schema
Path	**listId** *required*	listId	integer (int64)
Body	**newEntry** *required*	newEntry	ListEntry

Responses

HTTP Code	Description	Schema
200	OK	MovieList
201	Created	No Content
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

Consumes

- `application/json`

Produces

- `*/*`

Tags

- list-controller

deleteEntryFromList

```
DELETE /deleteEntry/{listId}/{listPosition}
```

Description

Deletes a specific entry from the list

Parameters

Type	Name	Description	Schema
Path	**listId** *required*	listId	integer (int64)

Type	Name	Description	Schema
Path	**listPosition** *required*	listPosition	integer (int32)

Responses

HTTP Code	Description	Schema
200	OK	MovieList
204	No Content	No Content
401	Unauthorized	No Content
403	Forbidden	No Content

Produces

- `*/*`

Tags

- list-controller

deleteList

```
DELETE /deleteList/{listId}
```

Description

Used to delete a list and all of its entries

Parameters

Type	Name	Description	Schema
Path	**listId** *required*	listId	integer (int64)

Responses

HTTP Code	Description	Schema
200	OK	string
204	No Content	No Content
401	Unauthorized	No Content

HTTP Code	Description	Schema
403	Forbidden	No Content

Produces

- `*/*`

Tags

- list-controller

errorHtml

```
POST /error
```

Responses

HTTP Code	Description	Schema
200	OK	ModelAndView
201	Created	No Content
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

Consumes

- `application/json`

Produces

- `text/html`

Tags

- basic-error-controller

errorHtml

```
GET /error
```

Responses

HTTP Code	Description	Schema
200	OK	ModelAndView
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

Produces

- `text/html`

Tags

- basic-error-controller

errorHtml

```
PUT /error
```

Responses

HTTP Code	Description	Schema
200	OK	ModelAndView
201	Created	No Content
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

Consumes

- `application/json`

Produces

- `text/html`

Tags

- basic-error-controller

errorHtml

```
DELETE /error
```

Responses

HTTP Code	Description	Schema
200	OK	ModelAndView
204	No Content	No Content
401	Unauthorized	No Content
403	Forbidden	No Content

Produces

- `text/html`

Tags

- basic-error-controller

errorHtml

```
PATCH /error
```

Responses

HTTP Code	Description	Schema
200	OK	ModelAndView
204	No Content	No Content
401	Unauthorized	No Content
403	Forbidden	No Content

Consumes

- `application/json`

Produces

- `text/html`

Tags

- basic-error-controller

errorHtml

```
HEAD /error
```

Responses

HTTP Code	Description	Schema
200	OK	ModelAndView
204	No Content	No Content
401	Unauthorized	No Content
403	Forbidden	No Content

Consumes

- `application/json`

Produces

- `text/html`

Tags

- basic-error-controller

errorHtml

```
OPTIONS /error
```

Responses

HTTP Code	Description	Schema
200	OK	ModelAndView
204	No Content	No Content
401	Unauthorized	No Content
403	Forbidden	No Content

Consumes

- `application/json`

Produces

- `text/html`

Tags

- basic-error-controller

getMovie

```
GET /getMovie/{id}
```

Description

Gets a specific movie specified by the id

Parameters

Type	Name	Description	Schema
Path	**id** *required*	id	integer (int32)

Responses

HTTP Code	Description	Schema
200	OK	MovieRoot
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

Produces

- `*/*`

Tags

- movie-api-controller

getPopular

```
GET /getPopular
```

Description

Gets a list of movies that are currently popular

Responses

HTTP Code	Description	Schema
200	OK	< PopularResult > array
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

Produces

- `*/*`

Tags

- movie-api-controller

getSimilarMovies

```
GET /getSimilar/{id}
```

Description

Returns a list of movies that are similar to the given movie

Parameters

Type	Name	Description	Schema
Path	**id** *required*	id	integer (int32)

Responses

HTTP Code	Description	Schema
200	OK	< PopularResult > array
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

Produces

- `*/*`

Tags

- movie-api-controller

getUpcoming

```
GET /getUpcoming
```

Description

Gets a list of movies that release in the near future

Responses

HTTP Code	Description	Schema
200	OK	< PopularResult > array
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

Produces

- `*/*`

Tags

- movie-api-controller

getLogIfExists

```
GET /hasLogged/{movieID}/{userID}
```

Description

Gets a log for a specific movie if it exists for that user

Parameters

Type	Name	Description	Schema
Path	**movieID** *required*	movieID	integer (int32)
Path	**userID** *required*	userID	integer (int64)

Responses

HTTP Code	Description	Schema
200	OK	MovieLog
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

Produces

- `*/*`

Tags

- movie-log-controller

getListById

```
GET /list/{listId}
```

Description

Gets a specific list by id

Parameters

Type	Name	Description	Schema
Path	**listId** *required*	listId	integer (int64)

Responses

HTTP Code	Description	Schema
200	OK	ListRoot
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

Produces

- `*/*`

Tags

- list-controller

createList

```
POST /list/{userId}
```

Description

Creates a movie list for a specified user

Parameters

Type	Name	Description	Schema
Path	**userId** *required*	userId	integer (int64)
Body	**movieList** *required*	movieList	MovieList

Responses

HTTP Code	Description	Schema
200	OK	MovieList
201	Created	No Content
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

Consumes

- `application/json`

Produces

- `*/*`

Tags

- list-controller

getUsersLists

```
GET /lists/{userId}
```

Description

Returns a list of movie lists that the user has created

Parameters

Type	Name	Description	Schema
Path	**userId** *required*	userId	integer (int64)

Responses

HTTP Code	Description	Schema
200	OK	< MovieList > array
401	Unauthorized	No Content
403	Forbidden	No Content

HTTP Code	Description	Schema
404	Not Found	No Content

Produces

- `*/*`

Tags

- list-controller

deleteUser

```
DELETE /log/{id}
```

Description

Deletes a movie log with the specified id

Parameters

Type	Name	Description	Schema
Path	**id** *required*	id	integer (int64)

Responses

HTTP Code	Description	Schema
200	OK	string
204	No Content	No Content
401	Unauthorized	No Content
403	Forbidden	No Content

Produces

- `*/*`

Tags

- movie-log-controller

createLog

```
POST /log/{userId}
```

Description

Creates a log for a specified user

Parameters

Type	Name	Description	Schema
Path	**userId** *required*	userId	integer (int64)
Body	**movieLog** *required*	movieLog	MovieLog

Responses

HTTP Code	Description	Schema
200	OK	string
201	Created	No Content
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

Consumes

- `application/json`

Produces

- `*/*`

Tags

- movie-log-controller

loginUser

```
GET /login/{username}/{password}
```

Description

Used to login a user, returns the user object if the credentials are correct

Parameters

Type	Name	Description	Schema
Path	**password** *required*	password	string
Path	**username** *required*	username	string

Responses

HTTP Code	Description	Schema
200	OK	User
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

Produces

- `*/*`

Tags

- admin-controller

getUsersMovieLogs

```
GET /logs/{userId}
```

Description

Gets all the logs that a user has made

Parameters

Type	Name	Description	Schema
Path	**userId** *required*	userId	integer (int64)

Responses

HTTP Code	Description	Schema
200	OK	< MovieLogResult > array
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

Produces

- `*/*`

Tags

- movie-log-controller

getUsersMovieLogsByDate

```
GET /logsByDate/{userId}
```

Description

Gets all the logs that a user has made sorted by date

Parameters

Type	Name	Description	Schema
Path	**userId** *required*	userId	integer (int64)

Responses

HTTP Code	Description	Schema
200	OK	< MovieLogResult > array
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

Produces

- */*

Tags

- movie-log-controller

getUsersMovieLogsByRating

```
GET /logsByRating/{userId}
```

Description

Gets all the logs that a user has made sorted by rating

Parameters

Type	Name	Description	Schema
Path	**userId** *required*	userId	integer (int64)

Responses

HTTP Code	Description	Schema
200	OK	< MovieLogResult > array
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

Produces

- */*

Tags

- movie-log-controller

renameList

```
POST /renameList/{listId}/{newName}
```

Description

Used to rename a list

Parameters

Type	Name	Description	Schema
Path	**listId** *required*	listId	integer (int64)
Path	**newName** *required*	newName	string

Responses

HTTP Code	Description	Schema
200	OK	MovieList
201	Created	No Content
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

Consumes

- `application/json`

Produces

- `*/*`

Tags

- list-controller

getAdmins

```
GET /role/admin
```

Description

Gets all users with the admin role

Responses

HTTP Code	Description	Schema
200	OK	< User > array
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

Produces

- `*/*`

Tags

- admin-controller

getModerators

```
GET /role/mod
```

Description

Gets all users with the mod role

Responses

HTTP Code	Description	Schema
200	OK	< User > array
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

Produces

- `*/*`

Tags

- admin-controller

getUserRole

```
GET /role/user
```

Description

Gets all users with the user role

Responses

HTTP Code	Description	Schema
200	OK	< User > array
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

Produces

- `*/*`

Tags

- admin-controller

searchMovie

```
GET /searchMovie/{term}
```

Description

Returns a list of movies that match the given query

Parameters

Type	Name	Description	Schema
Path	**term** *required*	term	string

Responses

HTTP Code	Description	Schema
200	OK	PopularRoot
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

Produces

- `*/*`

Tags

- movie-api-controller

searchMoviePage

```
GET /searchMovie/{term}/{page}
```

Description

Returns a list of movies that match the given query, with pagination

Parameters

Type	Name	Description	Schema
Path	**page** *required*	page	integer (int32)
Path	**term** *required*	term	string

Responses

HTTP Code	Description	Schema
200	OK	PopularRoot
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

Produces

- */*

Tags

- movie-api-controller

searchUser

```
GET /searchUser/{term}
```

Description

Used to search for a user by username

Parameters

Type	Name	Description	Schema
Path	**term** *required*	term	string

Responses

HTTP Code	Description	Schema
200	OK	< User > array
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

Produces

- */*

Tags

- admin-controller

setListEntreis

```
POST /setEntries/{listId}
```

Description

Sets the entries of a specified list

Parameters

Type	Name	Description	Schema
Path	**listId** *required*	listId	integer (int64)
Body	**newEntries** *required*	newEntries	< ListEntry > array

Responses

HTTP Code	Description	Schema
200	OK	MovieList
201	Created	No Content
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

Consumes

- `application/json`

Produces

- `*/*`

Tags

- list-controller

updateUserPass

```
PUT /userPassword/{id}
```

Description

Used to update a users password

Parameters

Type	Name	Description	Schema
Path	**id** *required*	id	integer (int64)
Body	**request** *required*	request	User

Responses

HTTP Code	Description	Schema
200	OK	User
201	Created	No Content
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

Consumes

- `application/json`

Produces

- `*/*`

Tags

- admin-controller

updateUserRole

```
PUT /userRole/{id}
```

Description

Update the role of a specified user

Parameters

Type	Name	Description	Schema
Path	**id** *required*	id	integer (int64)

Type	Name	Description	Schema
Body	**request** *required*	request	User

Responses

HTTP Code	Description	Schema
200	OK	User
201	Created	No Content
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

Consumes

- `application/json`

Produces

- `*/*`

Tags

- admin-controller

createUser

```
POST /users
```

Description

Creates a new user

Parameters

Type	Name	Description	Schema
Body	**user** *required*	user	User

Responses

HTTP Code	Description	Schema
200	OK	User
201	Created	No Content
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

Consumes

- `application/json`

Produces

- `*/*`

Tags

- admin-controller

getAllUsers

```
GET /users
```

Description

Returns a list of all the users

Responses

HTTP Code	Description	Schema
200	OK	< User > array
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

Produces

- `*/*`

Tags

- admin-controller

getUserById

```
GET /users/{id}
```

Description

Get a specific user by id

Parameters

Type	Name	Description	Schema
Path	**id** *required*	id	integer (int64)

Responses

HTTP Code	Description	Schema
200	OK	User
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

Produces

- `*/*`

Tags

- admin-controller

updateUser

```
PUT /users/{id}
```

Description

Updates a user with the specified id

Parameters

Type	Name	Description	Schema
Path	**id** *required*	id	integer (int64)
Body	**request** *required*	request	User

Responses

HTTP Code	Description	Schema
200	OK	User
201	Created	No Content
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

Consumes

- `application/json`

Produces

- `*/*`

Tags

- admin-controller

deleteUser

```
DELETE /users/{id}
```

Description

Deletes a user with the specified id

Parameters

Type	Name	Description	Schema
Path	**id** *required*	id	integer (int64)

Responses

HTTP Code	Description	Schema
200	OK	string
204	No Content	No Content
401	Unauthorized	No Content
403	Forbidden	No Content

Produces

- `*/*`

Tags

- admin-controller

approveFriendsRequest

```
POST /users/{id}/approveRequest/{req}
```

Description

Approves/Accepts a friend request

Parameters

Type	Name	Description	Schema
Path	**id** *required*	id	integer (int32)
Path	**req** *required*	req	integer (int32)
Body	**user** *required*	user	User

Responses

HTTP Code	Description	Schema
200	OK	string
201	Created	No Content
401	Unauthorized	No Content

HTTP Code	Description	Schema
403	Forbidden	No Content
404	Not Found	No Content

Consumes

- `application/json`

Produces

- `*/*`

Tags

- admin-controller

disconnect

```
POST /users/{id}/disconnect
```

Description

Removes a friend from your friend list

Parameters

Type	Name	Description	Schema
Path	**id** *required*	id	integer (int32)
Body	**req** *required*	req	User

Responses

HTTP Code	Description	Schema
200	OK	string
201	Created	No Content
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

Consumes

- `application/json`

Produces

- `*/*`

Tags

- admin-controller

showFriendsRequests

```
GET /users/{id}/friend_requests
```

Description

Shows the friend requests that a user has received

Parameters

Type	Name	Description	Schema
Path	**id** *required*	id	integer (int32)

Responses

HTTP Code	Description	Schema
200	OK	< User > array
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

Produces

- `*/*`

Tags

- admin-controller

showFriends

```
GET /users/{id}/friends
```

Description

Shows a users friend list

Parameters

Type	Name	Description	Schema
Path	**id** *required*	id	integer (int32)

Responses

HTTP Code	Description	Schema
200	OK	< User > array
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

Produces

- `*/*`

Tags

- admin-controller

showPotentialFriends

```
GET /users/{id}/potential_friends
```

Description

Shows the friend requests that a user has sent

Parameters

Type	Name	Description	Schema
Path	**id** *required*	id	integer (int32)

Responses

HTTP Code	Description	Schema
200	OK	< User > array
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

Produces

- `*/*`

Tags

- admin-controller

sendFriendsRequest

```
POST /users/{id}/request
```

Description

Sends a friend request to a specific user

Parameters

Type	Name	Description	Schema
Path	**id** *required*	id	integer (int32)
Body	**user** *required*	user	User

Responses

HTTP Code	Description	Schema
200	OK	string

HTTP Code	Description	Schema
201	Created	No Content
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

Consumes

- `application/json`

Produces

- `*/*`

Tags

- admin-controller

Definitions

Cast

Name	Schema
adult *optional*	boolean
cast_id *optional*	integer (int32)
character *optional*	string
credit_id *optional*	string
gender *optional*	integer (int32)
id *optional*	integer (int32)
known_for_department *optional*	string
name *optional*	string

Name	Schema
order *optional*	integer (int32)
original_name *optional*	string
popularity *optional*	number (double)
profile_path *optional*	string

Credits

Name	Schema
cast *optional*	< Cast > array
crew *optional*	< Crew > array

Crew

Name	Schema
adult *optional*	boolean
credit_id *optional*	string
department *optional*	string
gender *optional*	integer (int32)
id *optional*	integer (int32)
job *optional*	string
known_for_department *optional*	string
name *optional*	string
original_name *optional*	string

Name	Schema
popularity *optional*	number (double)
profile_path *optional*	string

Genre

Name	Schema
id *optional*	integer (int32)
name *optional*	string

ListEntry

Name	Schema
listPosition *optional*	integer (int32)
movieID *optional*	integer (int32)

ListRoot

Name	Schema
list *optional*	MovieList
moviesInList *optional*	< MovieListResult > array

ModelAndView

Name	Schema
empty *optional*	boolean
model *optional*	object
modelMap *optional*	< string, object > map

Name	Schema
reference *optional*	boolean
status *optional*	enum (100 CONTINUE, 101 SWITCHING_PROTOCOLS, 102 PROCESSING, 103 CHECKPOINT, 200 OK, 201 CREATED, 202 ACCEPTED, 203 NON_AUTHORITATIVE_INFORMATION, 204 NO_CONTENT, 205 RESET_CONTENT, 206 PARTIAL_CONTENT, 207 MULTI_STATUS, 208 ALREADY_REPORTED, 226 IM_USED, 300 MULTIPLE_CHOICES, 301 MOVED_PERMANENTLY, 302 FOUND, 302 MOVED_TEMPORARILY, 303 SEE_OTHER, 304 NOT_MODIFIED, 305 USE_PROXY, 307 TEMPORARY_REDIRECT, 308 PERMANENT_REDIRECT, 400 BAD_REQUEST, 401 UNAUTHORIZED, 402 PAYMENT_REQUIRED, 403 FORBIDDEN, 404 NOT_FOUND, 405 METHOD_NOT_ALLOWED, 406 NOT_ACCEPTABLE, 407 PROXY_AUTHENTICATION_REQUIRED, 408 REQUEST_TIMEOUT, 409 CONFLICT, 410 GONE, 411 LENGTH_REQUIRED, 412 PRECONDITION_FAILED, 413 PAYLOAD_TOO_LARGE, 413 REQUEST_ENTITY_TOO_LARGE, 414 URI_TOO_LONG, 414 REQUEST_URI_TOO_LONG, 415 UNSUPPORTED_MEDIA_TYPE, 416 REQUESTED_RANGE_NOT_SATISFIABLE, 417 EXPECTATION_FAILED, 418 I_AM_A_TEAPOT, 419 INSUFFICIENT_SPACE_ON_RESOURCE, 420 METHOD_FAILURE, 421 DESTINATION_LOCKED, 422 UNPROCESSABLE_ENTITY, 423 LOCKED, 424 FAILED_DEPENDENCY, 425 TOO_EARLY, 426 UPGRADE_REQUIRED, 428 PRECONDITION_REQUIRED, 429 TOO_MANY_REQUESTS, 431 REQUEST_HEADER_FIELDS_TOO_LARGE, 451 UNAVAILABLE_FOR_LEGAL_REASONS, 500 INTERNAL_SERVER_ERROR, 501 NOT_IMPLEMENTED, 502 BAD_GATEWAY, 503 SERVICE_UNAVAILABLE, 504 GATEWAY_TIMEOUT, 505 HTTP_VERSION_NOT_SUPPORTED, 506 VARIANT_ALSO_NEGOTIATES, 507 INSUFFICIENT_STORAGE, 508 LOOP_DETECTED, 509 BANDWIDTH_LIMIT_EXCEEDED, 510 NOT_EXTENDED, 511 NETWORK_AUTHENTICATION_REQUIRED)
view *optional*	View

Name	Schema
viewName *optional*	string

MovieList

Name	Schema
id *optional*	integer (int64)
listEntries *optional*	< ListEntry > array
listName *optional*	string

MovieListResult

Name	Schema
listEntry *optional*	ListEntry
movie *optional*	MovieRoot

MovieLog

Name	Schema
date *optional*	string (date-time)
id *optional*	integer (int64)
movie *optional*	integer (int32)
rating *optional*	number (double)

MovieLogResult

Name	Schema
log *optional*	MovieLog

Name	Schema
movie *optional*	MovieRoot

MovieRoot

Name	Schema
adult *optional*	boolean
backdrop_path *optional*	string
belongs_to_collection *optional*	object
budget *optional*	integer (int32)
credits *optional*	Credits
genres *optional*	< Genre > array
homepage *optional*	string
id *optional*	integer (int32)
imdb_id *optional*	string
original_language *optional*	string
original_title *optional*	string
overview *optional*	string
popularity *optional*	number (double)
poster_path *optional*	string
release_date *optional*	string
revenue *optional*	integer (int32)

Name	Schema
runtime *optional*	integer (int32)
status *optional*	string
tagline *optional*	string
title *optional*	string
video *optional*	boolean
vote_average *optional*	number (double)
vote_count *optional*	integer (int32)

PopularResult

Name	Schema
adult *optional*	boolean
backdrop_path *optional*	string
genre_ids *optional*	< integer (int32) > array
id *optional*	integer (int32)
original_language *optional*	string
original_title *optional*	string
overview *optional*	string
popularity *optional*	number (double)
poster_path *optional*	string
release_date *optional*	string

Name	Schema
title *optional*	string
video *optional*	boolean
vote_average *optional*	number (double)
vote_count *optional*	integer (int32)

PopularRoot

Name	Schema
page *optional*	integer (int32)
results *optional*	< PopularResult > array
total_pages *optional*	integer (int32)
total_results *optional*	integer (int32)

User

Name	Schema
firstName *optional*	string
id *optional*	integer (int32)
lastName *optional*	string
password *optional*	string
role *optional*	integer (int32)
username *optional*	string

View

Name	Schema
contentType *optional*	string