Software Development Kit for onPar System Documentation

The purpose of this software development kit (SDK) is to provide developers with an object oriented system that accesses data through the onPar application programming interface (API) without having to know how the API functions.

The SDK also takes away the need of handling API errors. The SDK provides its own alert messages and failure returns to the caller.

There are both private variables and methods that will be described and should not be used. The biggest one is every object has an ID. While this variable is not private, You should never manipulate any variable with ID in the name of any object.

The SDK also includes constants that should be used throughout the application.

First, the constants. All constants are stored in an object named 'defines.' The following table provides a name, whether private or public, way of accessing the constant, and the purpose of the constant.

Name	Private /Public	Accessing	Description
API_USERNAME	Private	defines.API_USERNAME	The username for accessing the API.
API_PASSWORD	Private	defines.API_PASSWORD	The password for accessing the API.
BASE_PATH	Public	defines.BASE_PATH	This is the base path of the application. When changing pages, such as href.windowlocation, etc., say defines.BASE_PATH + 'whateverPage';
AGGIES MAROONS COWBELLS BULLDOGS	Public	defines.AGGIES defines.MAROONS defines.COWBELLS defines.BULLDOGS	These are constants that represent the different tees that can be used for a round of golf.

Name	Private /Public	Accessing	Description
DRIVER THREE_WOOD FOUR_WOOD FIVE_WOOD SEVEN_WOOD NINE_WOOD TWO_HYBRID THREE_HYBRID FOUR_HYBRID FIVE_HYBRID SIX_HYBRID TWO_IRON THREE_IRON FOUR_IRON FIVE_IRON SIX_IRON SEVEN_IRON EIGHT_IRON NINE_IRON PW SW AW LW HLW	Public	defines.DRIVER defines. THREE_WOOD defines. FOUR_WOOD defines. FIVE_WOOD defines. SEVEN_WOOD defines. NINE_WOOD defines. TWO_HYBRID defines. THREE_HYBRID defines. FOUR_HYBRID defines. FIVE_HYBRID defines. SIX_HYBRID defines. TWO_IRON defines. THREE_IRON defines. FOUR_IRON defines. FIVE_IRON defines. SIX_IRON defines. SEVEN_IRON defines. EIGHT_IRON defines. NINE_IRON defines. PW defines. SW defines. AW defines. LW defines. HLW	These are constants representing the clubs used for golf shots. They are stored as an integer in the databases. The constants make it to where you never need to know what integer represents a club.

Next are a list of objects used in the SDK. Each object follows the same basic layout.

The first object is Course.

First, there is a way to create an empty course to be able to save a course to the database. This is how to do that.

Course
var c = new Course();

After creating an empty Course, there is a need to fill the variables. These examples take into account the Course that was just created - c. Variables are accessed by the method shown in the examples below. To change a variable, set it equal to its new

value. All the variables are listed in the following table. Variables that are not meant to be manipulated are red.

Variable	Javascript
Course ID	c.ID;
Course name	c.name;
Course location	c.location;

Then there has to be a way to save this new Course to the database. This function returns a boolean value to signal success or failure. To do this,

Javascript
var check = c.save();

If you create an existing Course, change something, and then need it updated in the database, you call the same function.

Now maybe you need to delete a Course. This function returns a boolean value to signal success or failure.

Javascript
var check = c.del();

That leads through creation of a Course. How to load a Course already in the database, you construct by the database ID of the Course. If you try to load a Course that does not exist in the database, Javascript will fill the variables with null. So I suggest checking for that in your code. The SDK will alert that the Course does not exist.

Javascript
var c = new Course(1);

But knowing all the IDs of the courses might not be possible. So there is a function called get all to be able to retrieve all the Courses in the database. Return data is an array of Course objects.

Javascript
var courses = CourseGetAll();

The next object is the User. It follows the same basic layout as Course.

To create a new User:

Javascript
var u = new User();

Variables nickname and memberID fields can be null. memberID is the only variable with ID in it that can be manipulated. Variables that are not meant to be manipulated are red.

User's name is in the format - Last, First - including the comma
User's memberID is in the format 1111M-222222
User's hand is a string either 'right' or 'left'

Variable	Javascript
ID	u.ID;
memberID	u.memberID;
nickname	u.nuckname;
name	u.name;
email	u.email;
age	u.age;
gender	u.gender;
hand	u.hand;
birthdate	u.birthdate;
rightHanded	u.rightHanded;
DBgender	u.DBgender;
stats	u.stats;

The save, delete, and getAll methods are the same as Course with the exception is Javascript in UserGetAll(). When constructing a User already in the database, it can be constructed by ID, email, or memberID. Both SDKs will alert accordingly if any are invalid.

The last object that connects to the API is round.

Round uses two objects that do not connect to the API - Hole and Shot.
These are used by Round as every Round contains x Holes and every Hole contains y Shots. You can create empty Shots to add to a Hole and Holes to add to the Round.

Find the class layout at the end of this document for variable tables.

The final object is Round.

Everything is the same as Course and User. Rounds can only be constructed by ID.

There is an added object called roundGetAll that holds an array of Round objects either belonging to a certain User or all the rounds in the database. This object is used to support pagination since the Round objects can become large. From this array, certain holes can be pulled out, certain shots with a club can be pulled out, etc.

Here is how to use this object:

Description	Javascript
To get all rounds in the database	var roundsObject = new RoundGetAll();
To get all rounds in the database for a certain User	var roundsObject = new RoundGetAll(user.ID);
The variable rounds is an array of rounds objects.	roundsObject.rounds;
The next function is called once when the object is initialized. This function calls the API to get the next page of results.	roundsObject.next();

To support pagination, there is a nextPage variable for this object. If the variable is not null, there are more objects in the database. The code here demonstrates a button press to get more rounds out of the database.	if (roundsObject.nextPage) {
	  roundsObject.next();
	}

Class Layout for Javascript

Course

Attributes

type	variable
int	ID
String	name
String	location

Methods

return type	declaration	Comment
Object	new Course(data);	constructor; can take nothing or int ID
BOOL	Course.save();	save method; inserts or updates
BOOL	Course.del();	delete method
Array	CourseGetAll();	gets all Courses

User

Attributes

type	name
int	ID
String	memberID
String	nickname
String	name
String	email
int	age
String	gender
String	hand
String	birthdate
BOOL	rightHanded
String	DBgender
Object	stats

Methods

return type	declaration	Comment
Object	new User(data);	constructor; can take nothing or int ID, or string email or memberID
BOOL	User.save();	save method; inserts or updates
BOOL	User.del();	delete method
Array	UserGetAll();	gets all Users

Shot

Attributes

type	name
int	ID
int	holeID
int	club
int	shotNumber
double	aimLatitude
double	aimLongitude
double	startLatitude
double	startLongitude
double	endLatitude
double	endLongitude

Methods

return type	declaration	Comment
Object	new Shot();	constructor

Hole
Attributes

int	ID
int	roundID
int	distance
int	holeScore
int	holeDistance
bool	FIR
bool	GIR
int	putts
double	firstRefLat
double	firstRefLong
double	secondRefLat
double	secondRefLong
double	thirdRefLat
double	thirdRefLong
int	firstRefX
int	firstRefY
int	secondRefX
int	secondRefY
int	thirdRefX
int	thirdRefY
Array	shots

Methods

return type	declaration	Comment
Object	new Hole();	constructor

Round

Attributes

type	name
int	ID
Course	course
int	totalScore
int	teeID
String	startTime
Array	holes

Methods

return type	declaration	Comment
Object	new Round(data);	constructor; can take nothing or int ID
BOOL	Round.save();	save method; inserts or updates
BOOL	Round.del();	delete method
Array	RoundGetAll();	gets all Rounds
Array	RoundGetAllByUser(userID);	gets all Rounds of a certain user