Knows how to serialize and deserialize the object to a JSON-compatible NSDictionary.

* Serving as the app’s data model by providing a filterable list of locations via filteredObjects.
* Communicating with the server by loading and saving items via import, persist and query.

The *Categories* class contains the list of categories that a Location can belong to and provides the ability to filter the list of locations by category. *Categories* also does the following:

Daten laden:

* Houses allCategories which provides the master list of categories. You can also add additional categories to its array.
* Provides a list of all categories in the active set of locations.
* Filters the locations by categories.
* - (void)import { [NSURL](http://developer.apple.com/documentation/Cocoa/Reference/Foundation/Classes/NSURL_Class/)\* url = [[NSURL](http://developer.apple.com/documentation/Cocoa/Reference/Foundation/Classes/NSURL_Class/) URLWithString:[kBaseURL stringByAppendingPathComponent:kLocations]]; *//1*   [NSMutableURLRequest](http://developer.apple.com/documentation/Cocoa/Reference/Foundation/Classes/NSMutableURLRequest_Class/)\* request = [[NSMutableURLRequest](http://developer.apple.com/documentation/Cocoa/Reference/Foundation/Classes/NSMutableURLRequest_Class/) requestWithURL:url]; request.HTTPMethod = *@*"GET"; *//2* [request addValue:*@*"application/json" forHTTPHeaderField:*@*"Accept"]; *//3*   NSURLSessionConfiguration\* config = [NSURLSessionConfiguration defaultSessionConfiguration]; *//4* NSURLSession\* session = [NSURLSession sessionWithConfiguration:config];   NSURLSessionDataTask\* dataTask = [session dataTaskWithRequest:request completionHandler:^([NSData](http://developer.apple.com/documentation/Cocoa/Reference/Foundation/Classes/NSData_Class/) \*data, [NSURLResponse](http://developer.apple.com/documentation/Cocoa/Reference/Foundation/Classes/NSURLResponse_Class/) \*response, [NSError](http://developer.apple.com/documentation/Cocoa/Reference/Foundation/Classes/NSError_Class/) \*error) { *//5* if (error == nil) { [NSArray](http://developer.apple.com/documentation/Cocoa/Reference/Foundation/Classes/NSArray_Class/)\* responseArray = [NSJSONSerialization JSONObjectWithData:data options:0 error:NULL]; *//6* [self parseAndAddLocations:responseArray toArray:self.objects]; *//7* } }];   [dataTask resume]; *//8* }
* The most important bits of information are the URL and request headers. The URL is simply the result of concatenating the base URL with the “locations” collections.
* You’re using GET since you’re reading data from the server. GET is the default method so it’s not necessary to specify it here, but it’s nice to include it for completeness and clarity.
* The server code uses the contents of the Accept header as a hint to which type of response to send. By specifying that your request will accept JSON as a response, the returned bytes will be JSON instead of the default format of HTML.
* Here you create an instance of *NSURLSession* with a default configuration.
* A *data task* is your basic NSURLSession task for transferring data from a web service. There are also specialized upload and download tasks that have specialized behavior for long-running transfers and background operation. A data task runs asynchronously on a background thread, so you use a callback block to be notified when the operation completes or fails.
* The completion handler checks for any errors; if it finds none it tries to deserialize the data using a NSJSONSerialization class method.
* Assuming the return value is an array of locations, parseAndAddLocations: parses the objects and notifies the view controller with the updated data.
* Oddly enough, data tasks are started with the resume message. When you create an instance of NSURLSessionTask it starts in the “paused” state, so to start it you simply call resume.

Daten speichern

- (void) persist:(Location\*)location { if (!location || location.name == nil || location.name.length == 0) { return; *//input safety check* }     [NSString](http://developer.apple.com/documentation/Cocoa/Reference/Foundation/Classes/NSString_Class/)\* locations = [kBaseURL stringByAppendingPathComponent:kLocations];   BOOL isExistingLocation = location.\_id != nil; [NSURL](http://developer.apple.com/documentation/Cocoa/Reference/Foundation/Classes/NSURL_Class/)\* url = isExistingLocation ? [[NSURL](http://developer.apple.com/documentation/Cocoa/Reference/Foundation/Classes/NSURL_Class/) URLWithString:[locations stringByAppendingPathComponent:location.\_id]] : [[NSURL](http://developer.apple.com/documentation/Cocoa/Reference/Foundation/Classes/NSURL_Class/) URLWithString:locations]; *//1*   [NSMutableURLRequest](http://developer.apple.com/documentation/Cocoa/Reference/Foundation/Classes/NSMutableURLRequest_Class/)\* request = [[NSMutableURLRequest](http://developer.apple.com/documentation/Cocoa/Reference/Foundation/Classes/NSMutableURLRequest_Class/) requestWithURL:url]; request.HTTPMethod = isExistingLocation ? *@*"PUT" : *@*"POST"; *//2*   [NSData](http://developer.apple.com/documentation/Cocoa/Reference/Foundation/Classes/NSData_Class/)\* data = [NSJSONSerialization dataWithJSONObject:[location toDictionary] options:0 error:NULL]; *//3* request.HTTPBody = data;   [request addValue:*@*"application/json" forHTTPHeaderField:*@*"Content-Type"]; *//4*   NSURLSessionConfiguration\* config = [NSURLSessionConfiguration defaultSessionConfiguration]; NSURLSession\* session = [NSURLSession sessionWithConfiguration:config];   NSURLSessionDataTask\* dataTask = [session dataTaskWithRequest:request completionHandler:^([NSData](http://developer.apple.com/documentation/Cocoa/Reference/Foundation/Classes/NSData_Class/) \*data, [NSURLResponse](http://developer.apple.com/documentation/Cocoa/Reference/Foundation/Classes/NSURLResponse_Class/) \*response, [NSError](http://developer.apple.com/documentation/Cocoa/Reference/Foundation/Classes/NSError_Class/) \*error) { *//5* if (!error) { [NSArray](http://developer.apple.com/documentation/Cocoa/Reference/Foundation/Classes/NSArray_Class/)\* responseArray = @[[NSJSONSerialization JSONObjectWithData:data options:0 error:NULL]]; [self parseAndAddLocations:responseArray toArray:self.objects]; } }]; [dataTask resume]; }

1. There are two endpoints for saving an object: /locations when you’re adding a new location, and /locations/\_id when updating an existing location that already has an id.
2. The request uses either PUT for existing objects or POST for new objects. The server code calls the appropriate handler for the route rather than using the default GET handler.
3. Because you’re updating an entity, you provide an HTTPBody in your request which is an instance of NSData object created by the NSJSONSerialization class.
4. Instead of an Accept header, you’re providing a Content-Type. This tells the bodyParser on the server how to handle the bytes in the body.
5. The completion handler once again takes the modified entity returned from the server, parses it and adds it to the local collection of Location objects.