ActiveResourceKit

Generated by Doxygen 1.7.5.1

Mon Oct 3 2011 12:15:23

Contents

Chapter 1

Class Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

Class Documentation

3.1 ARBase Class Reference

```
#import <ARBase.h>
```

Public Member Functions

- (id) - **initWithSite:**
- (id) - **initWithSite:elementName:**
- (NSString ∗) - prefixWithOptions:
- (NSString ∗) - **elementPathForID:prefixOptions:queryOptions:**
- (NSString ∗) - **newElementPathWithPrefixOptions:**
- (NSString ∗) - **collectionPathWithPrefixOptions:queryOptions:**
- (void) - buildWithAttributes:completionHandler:
- (void) - **findAllWithOptions:completionHandler:**
- (void) - findFirstWithOptions:completionHandler:
- (void) - **findLastWithOptions:completionHandler:**
- (id< ARFormat >) - **defaultFormat**
- (NSString ∗) - **defaultElementName**
- (NSString ∗) - **defaultCollectionName**
- (NSString ∗) - **defaultPrefixSource**
- (void) - **findEveryWithOptions:completionHandler:**
- (NSArray ∗) - instantiateCollection:prefixOptions:
- (AResource ∗) - **instantiateRecordWithAttributes:prefixOptions:**
- (NSSet ∗) - prefixParameters
- (void) - splitOptions:prefixOptions:queryOptions:
- (void) - get:completionHandler:

Properties

- NSDictionary ∗ **schema**
- NSArray ∗ knownAttributes
- NSURL ∗ site
- id< ARFormat > **format**
- NSTimeInterval **timeout**
- NSString ∗ **elementName**
- NSString ∗ **collectionName**
- NSString ∗ **prefixSource**

3.1.1 Detailed Description

Under Rails ActiveResource, the ActiveResource::Base singleton class carries the following state. See list below. This might help to define what ARBase does, its purpose. ARBase implements the anonymous singleton class behaviours belonging to ActiveResource::Base. AResource defines the class for Active Resource instances, but ARBase defines the class for Active Resource classes, singleton classes that is. Objective-C 2.0 does not provide anything like singleton classes. The Rails singleton class becomes the Objective-C "base" class.

auth_type collection_name connection element_name headers known_attributes nospam password prefix_parameters primary_key proxy schema site ssl_options test timeout user

Singleton methods for ActiveResource::Base include the following.

all auth_type build check_prefix_options collection_name collection_path connection create create_proxy_uri_from create_site_uri_from delete element_name element_path exists find find_every find_one find_single first format headers instantiate_collection instantiate_record known_attributes last new_element_path password prefix prefix_parameters prefix_source primary_key proxy query_string schema site split_options ssl_options timeout user

3.1.2 Member Function Documentation

3.1.2.1 - (void) buildWithAttributes: dummy(NSDictionary ∗) *attributes* completionHandler:(AResource ∗resource, NSError ∗error) *completionHandler*

Asynchronously builds an Active Resource.

Executes the completion handler on success or upon error. Completion handler arguments signal the outcome: non-nil attributes indicate successful completion. In such case, error always equals nil. There is no error.

3.1.2.2 - (void) findFirstWithOptions: dummy(NSDictionary ∗) *options* completionHandler:(AResource ∗resource, NSError ∗error) *completionHandler*

Answers just the first resource in a collection of resources. Finds all the resources first, then extracts the first element. Acts as a convenience wrapper.

3.1.2.3 - (void) get: dummy(NSString ∗) *path* completionHandler:(id object, NSError ∗error) *completionHandler*

Sends an asynchronous GET request. When the response successfully arrives, the format decodes the data. If the response body decodes successfully, finally sends the decoded object (or objects) to your given completion handler. Objects may be hashes (dictionaries) or arrays, or even primitives.

3.1.2.4 - (NSArray ∗) instantiateCollection: dummy(NSArray ∗) *collection* prefixOptions:(NSDictionary ∗) *prefixOptions*

Instantiates a collection of active resources given a collection of attributes; collection here meaning an array. The collection argument specifies an array of dictionaries. Each dictionary specifies attributes for a new active resource. Answers an array of newly instantiated active resources.

3.1.2.5 - (NSSet ∗) prefixParameters

Answers a set of prefix parameters based on the current prefix source. These constitute the current set of prefix parameters: an array of strings without the leading colon. Colon immediately followed by a word marks each parameter in the prefix source.

3.1.2.6 - (NSString ∗) prefixWithOptions: dummy(NSDictionary ∗) *options*

Answers the prefix after translating the prefix parameters according to the given prefix-options dictionary. The options dictionary may be nil. In that case -prefixWithOptions: answers the prefix unmodified. This assumes that the prefix contains no untranslated prefix parameters. The method quietly fails if you do not provide mappings for all parameters. The prefix result will contain parameter placeholders.

3.1.2.7 - (void) splitOptions: dummy(NSDictionary ∗) *options* prefixOptions:(NSDictionary ∗∗) *outPrefixOptions* queryOptions:(NSDictionary ∗∗) *outQueryOptions*

Splits an options dictionary into two dictionaries, one containing the prefix options, the other containing the leftovers, i.e. any query options.

3.1.3 Property Documentation

3.1.3.1 - (NSArray ∗) knownAttributes `[read, assign]`

Answers the known attributes, known because the resource server publishes them; the server does not necessarily publish everything. Known attributes depend on schema: they amount to the schema's keys. The schema is a dictionary of attribute name-type pairs.

3.1.3.2 - (NSURL∗) site `[read, write, copy]`

Always set up the site first. Other things depend on this essential property. You cannot fully operate the resource without the site URL. The site's path becomes the default prefix.

The documentation for this class was generated from the following files:

- ARBase.h
- ARBase.m

3.2 AResource Class Reference

```
#import <AResource.h>
```

Public Member Functions

- (id) - **initWithBase:**
- (id) - **initWithBase:attributes:**
- (void) - **loadAttributes:**
- (id) - **initWithBase:attributes:persisted:**

Properties

- ARBase ∗ base
- NSDictionary ∗ **attributes**
- NSDictionary ∗ **prefixOptions**
- BOOL **persisted**

3.2.1 Detailed Description

AResource is the core class mirroring Rails' ActiveResource::Base class. An active resource mimics an active record. Resources behave as records. Only their connection fundamentally differs. Active Records connect to a database whereas Active Resources connect to a RESTful API accessed via HTTP transport.

This class uses AResource as the class name rather than ARResource. A is for - Active, R for Resource. But the namespace convention for Objective-C lends itself to AResource if you eliminate the repeated term. Hence, ARResource becomes A-Resource!

3.2.2 Property Documentation

3.2.2.1 - (ARBase∗) base [read, write, retain]

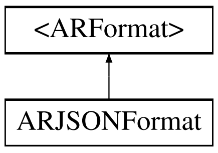
Retains the active-resource base. Does not copy the base. This has important implications. If you alter base properties, the changes affect all the resources which depend upon it.

The documentation for this class was generated from the following files:

- AResource.h
- AResource.m

3.3 <ARFormat> Protocol Reference

Inheritance diagram for <ARFormat>:



Public Member Functions

- (id) - **decode:error:**

Properties

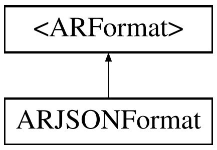
- NSString ∗ **extension**
- NSString ∗ **MIMEType**

The documentation for this protocol was generated from the following file:

- ARFormat.h

3.4 ARJSONFormat Class Reference

Inheritance diagram for ARJSONFormat:



Public Member Functions

- (id) - **decode:error:**

Static Public Member Functions

- (ARJSONFormat ∗) + **JSONFormat**

Properties

- NSString ∗ **extension**
- NSString ∗ **MIMEType**

The documentation for this class was generated from the following files:

- ARJSONFormat.h
- ARJSONFormat.m