Rails ActiveRecord Associations

belongs_to	has_one	has_many	has_and_belongs_to_many
association(force_reload = false)	*association*(force_reload = false)	*collection*(force_reload = false)	*collection*(force_reload = false)
		collection<<(object, …)	*collection*<<(object, …)
		collection.delete(object, …)	*collection*.delete(object, …)
association=(associate)	*association*=(associate)	*collection*=objects	*collection*=objects
		*collection_singular*_ids	*collection_singular*_ids
		*collection_singular*_ids=ids	*collection_singular*_ids=ids
		collection.clear	*collection*.clear
		collection.empty?	*collection*.empty?
		collection.size	*collection*.size
		collection.find(…)	*collection*.find(…)
		collection.exists?(…)	*collection*.exists?(…)
build_*association*(attributes = {})	build_*association*(attributes = {})	*collection*.build(attributes = {}, …)	*collection*.build(attributes = {})
create_*association*(attributes = {})	create_*association*(attributes = {})	*collection*.create(attributes = {})	*collection*.create(attributes = {})
	:as	:as	
			:association_foreign_key
:autosave	:autosave	:autosave	:autosave
:class_name	:class_name	:class_name	:class_name
:conditions	:conditions	:conditions	:conditions
:counter_cache			
		:counter_sql	:counter_sql
			:delete_sql
:dependent	:dependent	:dependent	
		:extend	:extend
		:finder_sql	:finder_sql
:foreign_key	:foreign_key	:foreign_key	:foreign_key
		:group	:group
:include	:include	:include	:include
			:insert_sql
			:join_table
		:limit	:limit
		:offset	:offset
	:order	:order	:order
:polymorphic			
	:primary_key	:primary_key	
:readonly	:readonly	:readonly	:readonly
:select	:select	:select	:select
	:source	:source	
	:source_type	:source_type	
	:through	:through	
:touch			
		:uniq	:uniq
:validate	:validate	:validate	:validate
To know whether there's and associated object just check *association*.nil?	To know whether there's and associated object just check *association*.nil?		
Assigning an object to a belongs_to association does *not* automatically save the object. It does not save the associated object either.	When you assign an object to a has_one association, that object is automatically saved (in order to update its foreign key). In addition, any object being replaced is also automatically saved, because its foreign key will change too. If either of these saves fails due to validation errors, then the assignment statement returns false and the assignment itself is cancelled. If the parent object (the one declaring the has_one association) is unsaved (that is, new_record? returns true) then the child objects are not saved. They will automatically when the parent object is saved. If you want to assign an object to a has_one association without saving the object, use the association.build method.	When you assign an object to a has_many association, that object is automatically saved (in order to update its foreign key). If you assign multiple objects in one statement, then they are all saved. If any of these saves fails due to validation errors, then the assignment statement returns false and the assignment itself is cancelled. If the parent object (the one declaring the has_many association) is unsaved (that is, new_record? returns true) then the child objects are not saved when they are added. All unsaved members of the association will automatically be saved when the parent is saved. If you want to assign an object to a has_many association without saving the object, use the collection.build method.	When you assign an object to a has_and_belongs_to_many association, that object is automatically saved (in order to update the join table). If you assign multiple objects in one statement, then they are all saved. If any of these saves fails due to validation errors, then the assignment statement returns false and the assignment itself is cancelled. If the parent object (the one declaring the has_and_belongs_to_many association) is unsaved (that is, new_record? returns true) then the child objects are not saved when they are added. All unsaved members of the association will automatically be saved when the parent is saved. If you want to assign an object to a has_and_belongs_to_many association without saving the object, use the collection.build method.