mapEqualNames

Syntax	<pre>set aTargetObject = anInputObject.mapEqualNames ([anotherInputObject]+);
	append mapEqualNames(anInputObject) to anArrayOfTargetObjects;</pre>	

Semantics	The macro generates set statements for all equal named, equally typed attributes found in the target object and at least one of the input objects. This applies to all **public** attributes including all inherited attributes. You can supply multiple input objects. This macro does not work recursively and thus do not perform a deep copy of attributes of complex types.	
Substitutables	`anInputObject`, `anotherInputObject`	Can be any **complex** object having attributes of any type.

Examples	<pre>set person = person1.mapEqualNames(person2, person3);
	set person.address = mapEqualNames(address);
	set person.alternativeAddress = detailedPerson.address.mapEqualNames();
	set person.adress = mapEqualNames(detailedPerson.addresses[0]);
	append mapEqualNames(detailedPerson) to people;</pre>	

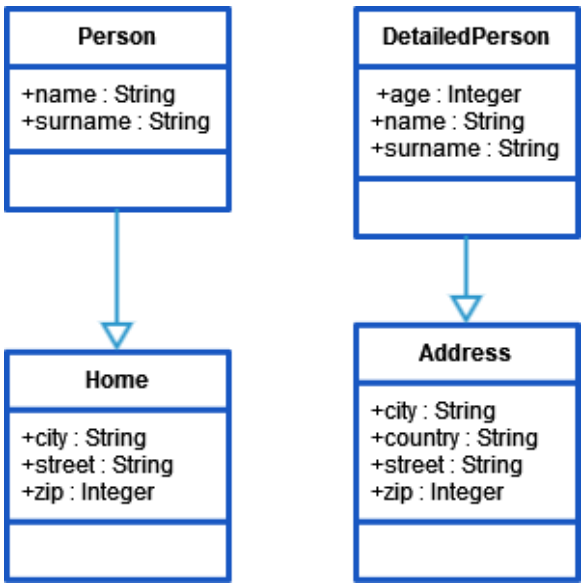
On this Page:

Related Pages:

- mapEqualNamesIfExists() Macro

Usage of mapEqualNames()

Given are the two unrelated classes **Person** and **DetailedPerson** .



Data / Script	Description / Result

personIn {
age: 45,
name: "Rose",
surname: "Bloom",
city: "San Francisco",
country: "USA",
street: "7, Waterfall Av.",
zip: 94016
 	
}
	The object **personIn** is of type **DetailedPerson** .

set personOut = mapEqualNames(personIn);	This statement assigns values to the matching attributes of object **personOut** which is of type **Person** :
	
	personOut {
	name: "Rose",
	surname: "Bloom",
	city: "San Francisco",
	street: "7, Waterfall Av.",
	zip: 94016
	}
	
	age and **country** remain unset.

Usage of "mapEqualNamesIfExists()"

Data / Script	Description / Result

personIn1 {
name: "John Robert Edward",
surname: "Snow",
city: "Anchorage",
street: "99, Malamute Street",
zip: 0
}
	The object **personIn1** of type **Person** already contains values.
	In our example, the zip code is unknown, so the attribute **zip** contains the value **0** .

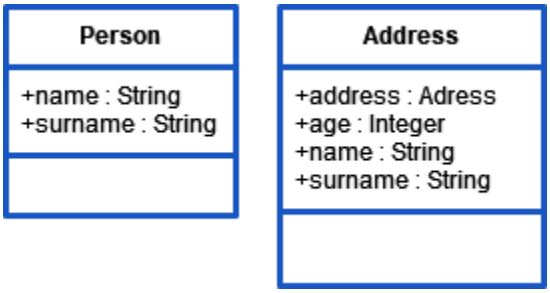
personIn2 {
age: 32,
name: "John",
surname: "Snow",
city: "Anchorage",
country: "USA",
zip: 12345
}
	The object **personIn2** is of type **DetailedPerson** .
	In this example object, the attribute **street** is not given.

set personIn1 = mapEqualNamesIfExists(personIn2);	This statement assigns values from **personIn2** to the matching attributes of **personIn1** :
	
	personIn1 {
	name: "John",
	surname: "Snow",
	city: "Anchorage",
	street: "99, Malamute Street",
	zip: 12345
	}
	
	The properties of **personIn1** get overwritten with existing values. The value of **street** remains unchanged, because it did not exist in the source object **personIn2** (was NULL).

Usage of mapEqualNames() with the "append" Statement

You can use `mapEqualNames()` along with the append statement to add a complex object with numerous attributes to an array of unrelated objects which needs only some of the information. The `mapEqualNames()`macro will create set statements for all equal named attributes found in the target object while the `append` statement will add the result to an array.

Given are the two unrelated classes **Person** and **DetailedPerson**.



Data / Script	Description / Result
`aDetailedPerson { adress: { city: "San Francisco", country: "USA", street: "7, Waterfall Av.", zip: 94016 } age: 45, name: "Rose", surname: "Bloom" }`	Assume that **aDetailedPerson** is provided from an external source. For further processing, we are interested in **name** and **surname** only.
`append mapEqualNames (aDetailedPerson) to customers;`	`mapEqualNames()` collects the data from **aDetailedPerson,** and `append` adds it to the array **customers** . **customers** is an array of type **Person** and contains already the data of " **John Snow** " and " **Liv Falls** ". Result array: `customers [ {name: "John", surname: "Snow";} {name: "Liv", surname: "Falls";} {name: "Rose", surname: "Bloom";} ]`