# Django Tips #4 Automatic DateTime Fields

Both Django’s DateTimeField and DateField have two very useful arguments for automatically managing date and time. If you want keep track on when a specific instance was created or updated you don’t need to do it manually: just set the auto\_now and auto\_now\_add arguments to True like in the following example:

class Invoice(models.Model):

description = models.CharField(max\_length=255)

status = models.CharField(max\_length=10)

vendor = models.ForeignKey(Vendor)

created\_at = models.DateTimeField(auto\_now\_add=True)

updated\_at = models.DateTimeField(auto\_now=True)

The auto\_now\_add will set the timezone.now() only when the instance is created, and auto\_now will update the field everytime the save method is called.

It is important to note that both arguments will trigger the field update event with timezone.now(), meaning when you create an object, both created\_at and updated\_at will be filled.

This is a very simple trick that will make your codebase cleaner.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* **auto\_now** - updates the value of field to current time and date every time the Model.save() is called.
* **auto\_now\_add** - updates the value with the time and date of creation of record.

My question is what if a filed in model contains both the auto\_now and auto\_now\_add set to True? What happens in that case?

auto\_now takes precedence (obviously, because it updates field each time, while auto\_now\_add updates on creation only).