**Summary: Issue and Tag helpers**

* This is used to help make the code more readable and help filter through the optional in core data
  + Optionals are converted into getters and setters to pass the data around easier
* Creates an example issue for the previews
* issueTags:
  + connects issues to tags through the relationship created in core data
* tagActiveIsssues:
  + connects tags to issues using relationship built in core data
  + filters for open tags

**Issue and Tag helpers**

* For the optional values and core data to remove the nil coalescing make an extension to set the values so the code easier to read
  + Strings
  + Dates
  + UUID
* Static example is used for previewing

Extension Issue

Var issueTags: [Tag]

Let result = tags?.allObjects as? [Tag] ?? []

Return result.sorted()

* [Tag] is the tag entity in core data
* tags? Is the relationship set up in core data
  + this is what maps issues to tags
  + this is where many to one is mapped there are many issues to one tag
  + many tags to one issue

Var issueTagList: String

* this is for the tags in IssueView and issueRow
  + so the tags can be read out individually
* check for tags else bail out
* if tag.count == 0
  + (NSSet must be count)
* Else
  + return the array of tags by their .tagName

var issueTitle: String

get { title ?? “”}

set{ title = newValue}

static example: Issue {

let controller = DataController(inMemory: True)

let viewContext = controller.container.viewContext

let issue = Issue(context: viewContext)

I issue.title = “example issue”

….

Return issue

Extension Tag

Var tagActiveIssues: [Issue]

Let result = issues?.allObjects as? [Issue] ?? []

Return result.filter { %0.completed == false }

* checks for the completed issues
* this is how each tag and issue inter connect