Adding new resources to AnnotationHub.

Marc Carlson

October 13, 2014

1 Overview of the process

If you are reading this it is (hopefully) because you intend to write some code that will allow the processing of online resources into R objects that are to be made available via that the *AnnotationHub* package. In order to do this you will have to do three basic steps (outlined below). These steps will have you writing two functions and then calling a third function to do some automatic set up for you. The 1st function will contain instructions on how to process data that is stored online into metadata for describing your new R resources for the AnnotationHub. And the 2nd function is for describing how to take these online resources and transform them into an R object that is useful to end users.

2 Introducing `AnnotationHubMetadata` Objects

The *AnnotationHubData* package is a complementary package to the *AnnotationHub* package that provides a place where we can store code that processes online resources into R objects suitable for access through the *AnnotationHub* package. But before you can understand the requirements for this package it is important that you 1st learn about the objects that are used as intermediaries between the hub and its web based repository behind the scenes. That means that you need to know about `AnnotationHubMetadata` objects. These objects store the metadata that describes an online resource. And if you want to see a set of online resources added to the repository and maintained, then it will be necessary to become familiar with the `AnnotationHubMetadata` constructor. For each online resource that you want to process into the AnnotationHub, you will have to be able to construct an `AnnotationHubMetadata` object that describes it in detail and that specifies where the recipe function lives.

3 Step 1: Writing your `AnnotationHubMetadata` generating function

The 1st function you need to provide is one that processes some online resources into `AnnotationHubMetadata` objects. This function MUST return a list of `AnnotationHubMetadata` objects. It can rely on other helper functions that you define, but ultimately it (and it's helpers) need to contain all of the instructions needed to find resources and process those resources into `AnnotationHubMetadata` objects.

The following example function takes files from the latest release of inparanoid and processes them into `AnnotationHubMetadata` objects using Map. The calling of the Map function is really the important part of this function, as it shows the function creating a series of `AnnotationHubMetadata` objects. Prior to that, the function was just calling out to other helper functions in order to process the metadata so that it could be passed to the `AnnotationHubMetadata` constructor using Map. Notice how one of the fields specified by this function is the Recipe, which indicates both the name and location of the recipe function. We expect most people will want to submit their recipe to the same package as they are submitting their metadata processing function.

```
> makeinparanoid8ToAHMs <- function(currentMetadata){
+     baseUrl <- 'http://inparanoid.sbc.su.se/download/current/Orthologs'
+     ## Make list of metadata in a helper function
+     meta <- .inparanoidMetadataFromUrl(baseUrl)
+     ## then make AnnotationHubMetadata objects.
+     Map(AnnotationHubMetadata,
+         Description=meta$description,
+         Genome=meta$genome,
+         SourceFile=meta$sourceFile,
+         SourceUrl=meta$sourceUrl,
+         SourceVersion=meta$sourceVersion,
+         Species=meta$species,
+         TaxonomyId=meta$taxonomyId,
+         Title=meta$title,
+         RDataPath=meta$rDataPath,
+         MoreArgs=list(
+           Coordinate_1_based = TRUE,
+           DataProvider = baseUrl,
+           Maintainer = "Marc Carlson <mcarlson@fhcrc.org>",
```

```
+           RDataClass = "SQLiteFile",
+           RDataDateAdded = Sys.time(),
+           RDataVersion = "0.0.1",
+           Recipe = c("inparanoid8ToDbsRecipe",
+                      package="AnnotationHubData"),
+           Tags = c("Inparanoid", "Gene", "Homology", "Annotation")))
+ }
```

4 Step 2: Writing your recipe

The 2nd kind of function you need to write is called a recipe function. It always must take an single argument that must be an `AnnotationHubMetadata` object. The job of a recipe function is to use the metadata from an `AnnotationHubMetadata` object to produce an R object or data file that will be retrievable from the AnnotationHub service later on. Below is a recipe function that calls some helper functions to generate an inparanoid database object from the metadata stored in it's `AnnotationHubMetadata` object.

```
> inparanoid8ToDbsRecipe <- function(ahm){
+     require(AnnotationForge)
+     inputFiles <- metadata(ahm)$SourceFile
+     dbname <- makeInpDb(dir=file.path(inputFiles,""),
+                         dataDir=tempdir())
+     db <- loadDb(file=dbname)
+     outputPath <- file.path(metadata(ahm)$AnnotationHubRoot,
+                             metadata(ahm)$RDataPath)
+     saveDb(db, file=outputPath)
+     outputFile(ahm)
+ }
```

5 Step 3: Calling the `makeAnnotationHubResource` helper

Finally you will need to call the `makeAnnotationHubResource` function to do some setup. This function only has two required arguments. The 1st is basically the name of a class that describes the kind of resource you are writing code to import. It just needs to be a unique name and will be used internally to create a class for dispatch. The 2nd argument is the name of your metadata processing function from step one. Once you have finished this, the only step left is to export the class name into the NAMESPACE (rememeber that this is that string you are providing as your 1st argument), and

then add this code to the *AnnotationHubData* repository. We are planning to set up a bridge to github so that you can give us a pull request.

```
> makeAnnotationHubResource("Inparanoid8ImportPreparer",
+                           makeinparanoid8ToAHMs)
```

6 Session Information

```
R version 3.1.1 Patched (2014-09-25 r66681)
Platform: x86_64-apple-darwin13.1.0 (64-bit)

locale:
[1] C/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

attached base packages:
[1] parallel  stats4    stats     graphics  grDevices utils     datasets
[8] methods   base

other attached packages:
[1] GenomicRanges_1.18.0 GenomeInfoDb_1.2.0   AnnotationHub_1.6.0
[4] IRanges_2.0.0        S4Vectors_0.4.0      BiocGenerics_0.12.0

loaded via a namespace (and not attached):
 [1] AnnotationDbi_1.28.0         Biobase_2.26.0
 [3] BiocInstaller_1.16.0         Category_2.32.0
 [5] DBI_0.3.1                    GSEABase_1.28.0
 [7] MASS_7.3-35                  Matrix_1.1-4
 [9] R6_2.0                       RBGL_1.42.0
[11] RColorBrewer_1.0-5           RJSONIO_1.3-0
[13] RSQLite_0.11.4               Rcpp_0.11.3
[15] XML_3.98-1.1                 XVector_0.6.0
[17] annotate_1.44.0              colorspace_1.2-4
[19] digest_0.6.4                 genefilter_1.48.0
[21] ggplot2_1.0.0                graph_1.44.0
[23] grid_3.1.1                   gridSVG_1.4-0
[25] gtable_0.1.2                 htmltools_0.2.6
[27] httpuv_1.3.0                 httr_0.5
[29] interactiveDisplay_1.4.0     interactiveDisplayBase_1.4.0
[31] lattice_0.20-29              mime_0.2
[33] munsell_0.4.2                plyr_1.8.1
```

```
[35] proto_0.3-10                reshape2_1.4
[37] rjson_0.2.14                scales_0.2.4
[39] shiny_0.10.2.1              splines_3.1.1
[41] stringr_0.6.2               survival_2.37-7
[43] tools_3.1.1                 xtable_1.7-4
```