

Issue #694: API policy on shared or global data

Description

We currently have the ability to create any number of basis managers and element shape lists, in fact `Cmiss_region_create()` creates a new region with its own such lists.

But bases and shapes are eminently sharable objects & some of our region merge code depends on these objects being from a unified list (this could be changed but it would be less efficient).

`Cmiss_region_create()` takes no arguments and I am reluctant to add an argument to borrow 'global' data from another region (as I have temporarily done with the internal `Cmiss_region_create_share_globals` which takes a source region as an argument). In any case this information is not needed until finite element fields are defined...

In the future, bases and shapes may well be just special types of fields read in from library regions on some server. To be efficient in cases where the same shapes and bases were used in multiple regions, we would want to cache data read in from distant file systems so expensive requests are avoided. This cache either needs to be passed in to appropriate functions, or become global data. Currently the only way to define finite element fields via the API is with:
`int Cmiss_region_read_file(struct Cmiss_region *region, char *file_name);`
This also provides no avenue for passing in shared data.

I was originally thinking we could make some of this data global, e.g. with global variables or singletons, but I'm not so sure now.

Instead I am thinking we should pass an additional argument into the `Cmiss_region_read_file` - a "Field_package" with enough global information to efficiently create all the known field types from, or even making this function part of the `Command_data` API...

We must also consider the future of:

1. API to load standard file formats vs.
 2. API as an interface to a large external "FieldML" data set e.g. using HDF5, vs.
 3. Script use of the API
- Any comments welcome.

Status: pending, Assigned to:

Issuedata

Classification: Idea , Project: cmgui , Categories: API , Importance: Medium

Progress

Deadline: 2008/08/02 11:55:36.402 GMT+12 , Percent done: 0

Contact

Name: Richard Christie , E-Mail: r.christie@auckland.ac.nz

Transcript

#1 Comment 2008-06-03 14:05 (rchristie)

Changed: categories: "(" -> "["API", "]"

Changed: OS: "Linux" -> "[""]

Changed: version: "Development" -> "[""]

Changed: title: "" -> "API policy on shared or global data"

Changed: classification: "Feature" -> "Idea"

Changed: description: "" -> "We currently have the ability to create any number of basis managers and element shape lists, in fact `Cmiss_region_create()` creates a new region with its own such lists.

But bases and shapes are eminently sharable objects & some of our region merge code depends on these objects being from a unified list (this could be changed but it would be less efficient).

`Cmiss_region_create()` takes no arguments and I am reluctant to add an argument to borrow 'global' data from another region (as I have temporarily done with the internal `Cmiss_region_create_share_globals` which takes a source region as an argument). In any case this information is not needed until finite element fields are defined...

In the future, bases and shapes may well be just special types of fields read in

Collector: Old tracker, do not use this anymore

from library regions on some server. To be efficient in cases where the same shapes and bases were used in multiple regions, we would want to cache data read in from distant file systems so expensive requests are avoided. This cache either needs to be passed in to appropriate functions, or become global data. Currently the only way to define finite element fields via the API is with:
int Cmiss_region_read_file(struct Cmiss_region *region, char *file_name);
This also provides no avenue for passing in shared data.

I was originally thinking we could make some of this data global, e.g. with global variables or singletons, but I'm not so sure now. Instead I am thinking we should pass an additional argument into the Cmiss_region_read_file - a "Field_package" with enough global information to efficiently create all the known field types from, or even making this function part of the Command_data API...

We must also consider the future of:

1. API to load standard file formats vs.
 2. API as an interface to a large external "FieldML" data set e.g. using HDF5, vs.
 3. Script use of the API
- Any comments welcome."

#2 Comment 2008-06-04 11:37 (blackett)

Comment

I prefer the idea of a Field and/or Region package for creating shared data.[]
I would rather not use the Command_data for this, as I do imagine projects may want to use part of the API and would benefit from not having to link in the whole cmgui project which the Command_data model tends to proscribe.[]
I also like the explicit package rather than hidden globals as it is clear that two regions may share things if based on the same package. You can then choose to not share stuff in certain cases, and within zinc the browser may start many instances of cmgui in the same memory space but these should not be sharing anything I think.[]

#3 Comment 2008-06-04 17:15 (andre)

Comment

Created by blackett on 2008-06-04 11:37[]

[]

:I prefer the idea of a Field and/or Region package for creating shared data.[]

[]

seconded.