

Kauri Routing Requests to Controllers

One thought hanging around my head is to use something like a static file tree listing of a directory to get a hold of the template-paths that should build up the restlet router.

e.g. for a directory with stuff like this:

```
$ tree
.
|-- src
|   |-- main
|   |   |-- java
|   |   |-- kauri
|   |   |   |-- classloader.xml
|   |   |   |-- local
|   |   |   |-- filter
|   |   |   |-- public
|   |   |       |-- contact
|   |   |           |-- {code}
|   |   |           |   |-- struct.js
|   |   |           |   |-- {code}.xml
|   |   |           |-- contact-search.xml
|   |   |           |-- index.js
|   |   |           |-- model
|   |   |           |-- contact.js
|   |   |       |-- ui
|   |   |           |-- contacts-manager.xhtml
```

you get a fitting list of paths with

```
$ find -type f
./src/main/kauri/public/model/contact.js
./src/main/kauri/public/contact-search.xml
./src/main/kauri/public/ui/contacts-manager.xhtml
./src/main/kauri/public/contact/{code}.xml
./src/main/kauri/public/contact/{code}/struct.js
./src/main/kauri/public/index.js
```

This kinda automatically lists template-uri's that could be 'attached' to a restlet router, uhuh.

Together with this there should be some mechanism to allow 'activiating' some of these resources: ie. recognise them as resource-producers (teemplates, groovy controller, ...) rather than static content to be served up.

Obviously there are quite some remarks to make

- these { } marks in directories and files give a weird feeling (although they seem to be supported on common operating systems, not sure about the various editors on those platforms (although those should behave well by using system-api file open/save stuff) (we should test network-mounted filesystems as well) (and not make broad statements about what tools should do, but make sure to double-check ourselves)
- the order/precedence isn't really controlled (and might defer on windows/linux and/or when read from actual paths versus from inside module-jars - this would call for some kind of 'best' routing match like supported in restlet (although probably not the most performant for large sets)
- there is some stress to keep 2 sides of these 'paths' happy
 - since this assembles a uri, the extension should be editable to make that most useful (specially for 'dynamic' resources)
 - while the file on disk probably wants to just hold the extension that ensures launching the correct editor (bummer)