[Jar info 1](#_Toc455697623)

[Servlet/Controller view 2](#_Toc455697624)

[Servlet/webSession view 3](#_Toc455697625)

[Controller - configurable timer and thread pools added 3](#_Toc455697626)

[Asynchronous Crest Client added 4](#_Toc455697627)

[The Data Cache 5](#_Toc455697628)

# Jar info



# Servlet/Controller view



# Servlet/webSession view



# Controller - configurable timer and thread pools added



# Asynchronous Crest Client added



Note:

Some "ContactsWebPage calls the static get on the Contacts (CrestData), which uses the CrestClient to instantiate the needed CrestGetTask and submit it to the BlockingExecutor. The ContactsWebPage call returns at this point, non-blocked. The remainder executes in its own thread obtained from the Executor.

The CrestGetTask will first waitAsNeeded on the request url's cache time (url specific RequestThrottle), if this url has been seen before. It then waitAsNeeded on the generic system wide RequestThrottle and then executes the request. Note that the url specific wait types are in the minutes. When the request returns to the CrestGetTask object it uses Gson to convert the json string to the requested CrestData object. The given CrestResponseCallback object (initially from ContactsWebPage call) is then called to deliver it's requested CrestData object (Contacts in this example).

The Data Cache

****

Note:

The cache is going to be bitching. For those not in the know, when I was a teen bitching meant cool. [https://www.youtube.com/watch?v=05AeeEIbnsM](https://www.youtube.com/watch?v=05AeeEIbnsM%20)

The DataCache exposes two getters for each unique EveData type (xml and/or crest api). Contracts is shown as one of the many.

* DataCache.getContracts.get(CrestClientInfo cinfo)
* DataCache.getContracts.get(CrestClientInfo cinfo, CrestClientInfo friendInfo)

CrestClientInfo contains a User object which will be provided by the database. I won't go into it here, but that user will also be tied into AccessGroup's and Rights objects that will be used by the second getter above to determine if the caller (cinfo) should be able to access the friendInfo's Contracts.

An Event firing mechanism will also be provided where SomeWebPage can register a callback listener on some watched EveData (i.e. Contracts). I'll implement equals methods on all of the EveData implementations so these events can be fired on change only seen.