Subject: Re: NEW: Dispatcher (templateable dispatcher helper for MVC pattern and more)
Posted by kohait00 on Thu, 15 Jul 2010 12:33:12 GMT
View Forum Message <> Reply to Message

Quote:elliminate the overly complex PTEABLE branch.
Ok, that should be no problem..

Quote:
make DispatcherGen use Dispatcher objects internally (as I did) or even derive it from Dispatcher

from which Dispatcher<T> should it derive? the goal is to have a generic Dispatcher, that can register arbitrary types. it wouldnt work becase DispatcherGen only would support one T again. the options is only to have a List / Vector / Array / Map of Dispatcher<T> inside to check then and forward to, or to use 'Container'<Any> to do it. i cant imagine that it would be that much of a big performance strike..look at the Any implementation, its a O(1) funtion call. but in huge amounts of different types, ofcorse yes.. so thats the 2 options.

Quote:
do away with the NOMAP , separate them completly

thats probably the best solution, dont worry, i hardly can read it myself i am not that much a fan of #ifdef branches

i will provide a clean package.

on more usecases: i am currently dot.net Reflector'ing the Microsoft.CCR.Core.dll to get a grisp on how they have done this whole Ports and Arbiters thing, believe it or not, 50% of that is already provided in Upp, named CoWork and Callbacks . this is actually the direction i want to go for..

dont know if you ever used Microsoft Robotics Development Studio (current RC3 is for free again). but the design patterns are really coool. so i'd like to have some of the benefits in upp as well.