Item 117
git_comments:

1. * * Licensed to the Apache Software Foundation (ASF) under one or more * contributor license agreements. See the NOTICE file distributed with * this work for additional information regarding copyright ownership. * The ASF licenses this file to You under the Apache License, Version 2.0 * (the "License"); you may not use this file except in compliance with * the License. You may obtain a copy of the License at * * http://www.apache.org/licenses/LICENSE-2.0 * * Unless required by applicable law or agreed to in writing, software * distributed under the License is distributed on an "AS IS" BASIS, * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. * See the License for the specific language governing permissions and * limitations under the License.
2. should inherit default values
3. should inherit default options
4. * * Licensed to the Apache Software Foundation (ASF) under one or more * contributor license agreements. See the NOTICE file distributed with * this work for additional information regarding copyright ownership. * The ASF licenses this file to You under the Apache License, Version 2.0 * (the "License"); you may not use this file except in compliance with * the License. You may obtain a copy of the License at * * http://www.apache.org/licenses/LICENSE-2.0 * * Unless required by applicable law or agreed to in writing, software * distributed under the License is distributed on an "AS IS" BASIS, * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. * See the License for the specific language governing permissions and * limitations under the License.
5. create a thread pool from big
6. create a thread pool from low
7. Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements. See the NOTICE file distributed with this work for additional information regarding copyright ownership. The ASF licenses this file to You under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at http://www.apache.org/licenses/LICENSE-2.0 Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.
8. Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements. See the NOTICE file distributed with this work for additional information regarding copyright ownership. The ASF licenses this file to You under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at http://www.apache.org/licenses/LICENSE-2.0 Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.
9. define a low profile
10. define a big profile with many threads
11. create and register the default profile
12. and mark the new profile as default
13. the default profile has the following values
14. create the pool
15. fallback to use values from default profile if not specified
16. validate that all options has been given as its mandatory for a default thread pool profile as it is used as fallback for other profiles if they do not have that particular value
17. no profile with that id
18. fallback and use old default values for new default profile if absent (convention over configuration)
19. * * Gets the thread pool profile by the given id * * @param id id of the thread pool profile to get * @return the found profile, or <tt>null</tt> if not found
20. * * Registers the given thread pool profile * * @param profile the profile
21. * * Creates a new thread pool using based on the given profile id. * * @param source the source object, usually it should be <tt>this</tt> passed in as parameter * @param name name which is appended to the thread name * @param threadPoolProfileId id of the thread pool profile to use for creating the thread pool * @return the created thread pool, or <tt>null</tt> if the was no thread pool profile with that given id.
22. * * Gets the id of this profile * * @return the id of this profile

23. should inherit the default values
24. just change the max pool as the default profile should then inherit the old default profile

git_commits:

1. **summary:** CAMEL-1588: Added support for multiple threadPoolProfile. And a custom thread pool profile will inherit parameters from default if not explict given.
   message: CAMEL-1588: Added support for multiple threadPoolProfile. And a custom thread pool profile will inherit parameters from default if not explict given. git-svn-id: https://svn.apache.org/repos/asf/camel/trunk@925588 13f79535-47bb-0310-9956-ffa450edef68

github_issues:

github_issues_comments:

github_pulls:

github_pulls_comments:

github_pulls_reviews:

jira_issues:

1. **summary:** ExecutorService configuration
   description: Camel uses some thread pools, for component, endpoint and processor. We should have some default configuration that is easy to configure with Camel to adjust pool sizes for individual/groups/type etc. By not they use 5 as the default pool size. And have their own executor unless configured otherwise.
2. **summary:** ExecutorService configuration
   description: Camel uses some thread pools, for component, endpoint and processor. We should have some default configuration that is easy to configure with Camel to adjust pool sizes for individual/groups/type etc. By not they use 5 as the default pool size. And have their own executor unless configured otherwise.
3. **summary:** ExecutorService configuration
   description: Camel uses some thread pools, for component, endpoint and processor. We should have some default configuration that is easy to configure with Camel to adjust pool sizes for individual/groups/type etc. By not they use 5 as the default pool size. And have their own executor unless configured otherwise.
4. **summary:** ExecutorService configuration
   description: Camel uses some thread pools, for component, endpoint and processor. We should have some default configuration that is easy to configure with Camel to adjust pool sizes for individual/groups/type etc. By not they use 5 as the default pool size. And have their own executor unless configured otherwise.
5. **summary:** ExecutorService configuration
   description: Camel uses some thread pools, for component, endpoint and processor. We should have some default configuration that is easy to configure with Camel to adjust pool sizes for individual/groups/type etc. By not they use 5 as the default pool size. And have their own executor unless configured otherwise.
6. **summary:** ExecutorService configuration
   description: Camel uses some thread pools, for component, endpoint and processor. We should have some default configuration that is easy to configure with Camel to adjust pool sizes for individual/groups/type etc. By not they use 5 as the default pool size. And have their own executor unless configured otherwise.
7. **summary:** ExecutorService configuration
   description: Camel uses some thread pools, for component, endpoint and processor. We should have some default configuration that is easy to configure with Camel to adjust pool sizes for individual/groups/type etc. By not they use 5 as the default pool size. And have their own executor unless configured otherwise.
8. **summary:** ExecutorService configuration

description: Camel uses some thread pools, for component, endpoint and processor. We should have some default configuration that is easy to configure with Camel to adjust pool sizes for individual/groups/type etc. By not they use 5 as the default pool size. And have their own executor unless configured otherwise.

9. **summary:** ExecutorService configuration
   description: Camel uses some thread pools, for component, endpoint and processor. We should have some default configuration that is easy to configure with Camel to adjust pool sizes for individual/groups/type etc. By not they use 5 as the default pool size. And have their own executor unless configured otherwise.
10. **summary:** ExecutorService configuration
   description: Camel uses some thread pools, for component, endpoint and processor. We should have some default configuration that is easy to configure with Camel to adjust pool sizes for individual/groups/type etc. By not they use 5 as the default pool size. And have their own executor unless configured otherwise.
11. **summary:** ExecutorService configuration
   description: Camel uses some thread pools, for component, endpoint and processor. We should have some default configuration that is easy to configure with Camel to adjust pool sizes for individual/groups/type etc. By not they use 5 as the default pool size. And have their own executor unless configured otherwise.
12. **summary:** ExecutorService configuration
   description: Camel uses some thread pools, for component, endpoint and processor. We should have some default configuration that is easy to configure with Camel to adjust pool sizes for individual/groups/type etc. By not they use 5 as the default pool size. And have their own executor unless configured otherwise.
13. **summary:** ExecutorService configuration
   description: Camel uses some thread pools, for component, endpoint and processor. We should have some default configuration that is easy to configure with Camel to adjust pool sizes for individual/groups/type etc. By not they use 5 as the default pool size. And have their own executor unless configured otherwise.
14. **summary:** ExecutorService configuration
   description: Camel uses some thread pools, for component, endpoint and processor. We should have some default configuration that is easy to configure with Camel to adjust pool sizes for individual/groups/type etc. By not they use 5 as the default pool size. And have their own executor unless configured otherwise.
   label: code-design
15. **summary:** ExecutorService configuration
   description: Camel uses some thread pools, for component, endpoint and processor. We should have some default configuration that is easy to configure with Camel to adjust pool sizes for individual/groups/type etc. By not they use 5 as the default pool size. And have their own executor unless configured otherwise.
16. **summary:** ExecutorService configuration
   description: Camel uses some thread pools, for component, endpoint and processor. We should have some default configuration that is easy to configure with Camel to adjust pool sizes for individual/groups/type etc. By not they use 5 as the default pool size. And have their own executor unless configured otherwise.
17. **summary:** ExecutorService configuration
   description: Camel uses some thread pools, for component, endpoint and processor. We should have some default configuration that is easy to configure with Camel to adjust pool sizes for individual/groups/type etc. By not they use 5 as the default pool size. And have their own executor unless configured otherwise.
18. **summary:** ExecutorService configuration
   description: Camel uses some thread pools, for component, endpoint and processor. We should have some default configuration that is easy to configure with Camel to adjust pool sizes for individual/groups/type etc. By not they use 5 as the default pool size. And have their own executor unless configured otherwise.
19. **summary:** ExecutorService configuration
   description: Camel uses some thread pools, for component, endpoint and processor. We should have some default configuration that is easy to configure with Camel to adjust pool sizes for

individual/groups/type etc. By not they use 5 as the default pool size. And have their own executor unless configured otherwise.

jira_issues_comments:

1. Supports thread pools using core pool size, max pool size, timeout etc. e.g. we gotta do this ourself instead of the pre existing on the Executors. {code} ExecutorService executor = new ThreadPoolExecutor(numberOfThreads, numberOfThreads, 60, TimeUnit.SECONDS, new LinkedBlockingQueue<Runnable>()); {code} This can be retrofitted in the ExecutorServiceHelper which camel uses to obtain a default pool.
2. See wiki page http://cwiki.apache.org/confluence/display/CAMEL/Threading+Model
3. trunk: 925588. Added support for multiple thread pool profiles with inheritance from the default profile for options not explicit given
4. trunk: 925181. Thread pools is now managed from JMX
5. See more details here http://cwiki.apache.org/confluence/display/CAMEL/Camel+2.3+-+ThreadPool+Configuration
6. Added a threadPool in Spring XML so you can easily define custom pools and use in Camel routes {code:xml} <camelContext id="camel" xmlns="http://camel.apache.org/schema/spring"> <!-- define a custom thread pool --> <threadPool id="myPool" poolSize="2" maxPoolSize="4" threadName="myPool"/> <route> <from uri="direct:start"/> <!-- use the custom thread pool in the camel route --> <threads executorServiceRef="myOtherPool"> <to uri="mock:result"/> </threads> </route> </camelContext> {code} trunk: 919408.
7. Aligned DSL to unify thread pool configuration on EIPs by introducing a {{ExecutorServiceAware}} interface for the EIPs trunk: 919331.
8. trunk: 919389. threads DSL can now be much more configured with max pool size, keep alive timeout and the likes
9. trunk: 919382. Prefer to use CachedExecutorService as its the best general purpose pool. It can grown/shrink and recommended to be used by the JDK and concurrent experts.
10. trunk: 920791. the thread name can now be configured using a pattern like syntax
11. Having ExecutorServiceStrategy to handle shutdown of thread pools trunk: 922485.
12. Introduced {{ShutdownableService}} so having CamelContext shutdown when its stops. trunk: 922296.
13. trunk: 922185. preparing for pattern based configuration
14. **body:** EIP model now creates thread pool for its EIP processor, so they are pre created and we have better knowledge which route uses which pools. trunk: 922524.
    label: code-design
15. Made much more progress on this today. See wiki page for details http://cwiki.apache.org/confluence/display/CAMEL/Camel+2.3+-+ThreadPool+Configuration What is missing is the JMX part :)
16. I have added a default thread pool file which can be configured in both Java DSL and Spring XML. Here is a sample from Spring XML {code:xml} <camelContext id="camel" xmlns="http://camel.apache.org/schema/spring"> <!-- define the default thread pool profile to be used in Camel --> <threadPoolProfile id="myDefaultProfile" defaultProfile="true" poolSize="5" keepAliveTime="25" maxPoolSize="15" maxQueueSize="250" rejectedPolicy="Abort"/> <route> <from uri="direct:start"/> <threads/> <to uri="mock:result"/> </route> </camelContext> {code} The profile dictates that all the default kind of thread pools Camel creates will follow the profile. For example <threads/> etc. Of course background threads will still be a single threaded pool etc. In the future those profiles, can be enhanced to have a pattern like options so you can say, route X should use this profile, route Y that profile etc.
17. Closing all resolved tickets from 2010 or older