A CSP Model Spring Dynamic Modules (v0.1)

Glyn Normington

July 18, 2008

The aim is to model the concurrency features of Spring DM.

Contents

1 Bundles

The events in this model should be just sufficient to model the interaction between bundles and Spring DM.

We recycle bundles to avoid artificial deadlock.

$$channel\ b\_started, b\_stopping : bundle$$

$$\begin{aligned}
&Bundles = \\
&\quad let \\
&\qquad Bundle(i) = b_started.i \rightarrow b_stopping.i \rightarrow Bundle(i) \\
&\quad within \\
&\qquad |||_{i:bundle}\ Bundle(i)
\end{aligned}$$

2 Service Registry

$$channel\ sr\_publish, sr\_lookup, sr\_retract : service$$

$$ServiceRegistration(s) = sr\_publish.s \rightarrow PublishedServiceRegistration(s)$$

$$\begin{aligned}
&PublishedServiceRegistration(s) = \\
&\quad sr_lookup.s \rightarrow PublishedServiceRegistration(s) \ \Box\ sr_retract.s \rightarrow ServiceRegistration(s)
\end{aligned}$$

$$ServiceRegistry = |||_{s:service}\ ServiceRegistration(s)$$

3 Application Contexts

Recycle application contexts to avoid artifical deadlock, but only within a single bundle start/stop cycle.

$$channel\ ac\_start, ac\_stop, ac\_refresh, ac\_close, ac\_fail, ac\_timeout : application\_context$$

$$StopAppCtx(a) = ac\_stop.a \rightarrow AppCtx(a)$$

$$\begin{aligned}
&AppCtx(a) = \\
&\quad let \\
&\qquad Normal = ac_refresh.a \rightarrow ServiceLookup(a, service) \\
&\qquad Error = ac_fail.a \rightarrow StopAppCtx(a) \\
&\quad within \\
&\qquad ac_start.a \rightarrow (Normal\ \Box\ Error)
\end{aligned}$$

$$ServiceLookup(a, \emptyset) = ServicePublish(a, \emptyset, \emptyset)$$

$$\begin{array}{l} ServiceLookup(a, remaining) = \\ \quad let \\ \quad\quad Timeout = ac\_timeout.a \rightarrow StopAppCtx(a) \\ \quad\quad Lookup = (\bigsqcap_{s:remaining} sr\_lookup.s \rightarrow ServiceLookup(a, remaining \setminus \{s\})) \triangleright Timeout \\ \quad within \\ \quad\quad ServicePublish(a, remaining \cap get\_services(a), \emptyset) \sqcap Lookup \end{array}$$

$$ServicePublish(a, \emptyset, published) = RunningAppCtx(a, published)$$

$$\begin{array}{l} ServicePublish(a, remaining, published) = \\ \quad let \\ \quad\quad PublishMore(s) = ServicePublish(a, remaining \setminus \{s\}, published \cup \{s\}) \\ \quad\quad Publish = \bigsqcap_{s:remaining} sr\_publish.s \rightarrow PublishMore(s) \\ \quad within \\ \quad\quad RunningAppCtx(a, published) \sqcap Publish \end{array}$$

$$RunningAppCtx(a, published) = ac\_stop.a \rightarrow ac\_close.a \rightarrow ServiceTerm(a, published)$$

$$ServiceTerm(a, \emptyset) = AppCtx(a)$$

$$ServiceTerm(a, published) = \Box_{s:published} sr\_retract.s \rightarrow ServiceTerm(a, published \setminus \{s\})$$

$$AppCtxs = |||_{a:application\_context} AppCtx(a)$$

4 Bundle to application context multiplexing

Multiplex bundle stopped and starting events to application context start and stop events, respectively.

$$\begin{array}{l} Multiplex(i) = \\ \quad let \\ \quad\quad ac = get\_app\_ctxs(i) \\ \quad\quad StartAppCtxs(\emptyset) = Multiplex(i) \\ \quad\quad StartAppCtxs(acs) = \Box_{a:acs} ac\_start.a \rightarrow StartAppCtxs(acs \setminus \{a\}) \\ \quad\quad StopAppCtxs(\emptyset) = Multiplex(i) \\ \quad\quad StopAppCtxs(acs) = \Box_{a:acs} ac\_stop.a \rightarrow StopAppCtxs(acs \setminus \{a\}) \\ \quad within \\ \quad\quad b\_started.i \rightarrow StartAppCtxs(ac) \Box b\_stopping.i \rightarrow StopAppCtxs(ac) \end{array}$$

$$Multiplexer = |||_{i:bundle} Multiplex(i)$$

5 System

A system comprising bundles, multiplexer, application contexts, and the service registry.

$$MultiplexedBundles = (Bundles \,|[\, \{|\ b\_started, b\_stopping\ |\} \,]|\, Multiplexer) \setminus \{|\ b\_started, b\_stopping\ |\}$$

$$EmbeddedAppCtxs = (AppCtxs \,|[\, \{|\ sr\_publish, sr\_lookup, sr\_retract\ |\} \,]|\, ServiceRegistry) \setminus \{|\ sr\_publish, sr\_lookup, sr\_retract\ |\}$$

$$System = MultiplexedBundles \,|[\, \{|\ ac\_start, ac\_stop\ |\} \,]|\, EmbeddedAppCtxs$$