xio Documentation

Release 1.0.0

xio

Jan 23, 2017

Contents

Contents:

CHAPTER 1

XIO Quickstart

1.1 Maven Artifacts

1.2 Simple HTTP Application

CHAPTER 2

Application

The Application object holds all of the global state and configuration data across multiple server objects. It should be configured and instantiated by an ApplicationBootstrap instance.

```
1 Application application = new ApplicationBootstrap("example.application")
2   .addServer("echo", (serverBootstrap) -> serverBootstrap.addToPipeline(new␣
  ↪EchoPipeline()))
3   .addServer("http", (serverBootstrap) -> serverBootstrap.addToPipeline(new␣
  ↪HttpPipeline()))
4   .build();
```

Each application will be created with a *configuration server* which may be used to update *dynamic configuration* values while the application is running.

CHAPTER 3

Server

The XioServer object holds the state and configuration data for a single server instance. It should be configured and instantiated by a XioServerBootstrap object.

```
1 XioServer server = XioServerBootstrap.fromConfig("example.application")
2   .addToPipeline(new XioHttp1_1Pipeline())
3   .build();
```

Each server will be created with a XioServerInstrumentation object which can be used to interrogate the server about the bound InetSocketAddress and the application protocol that the server is configured for.

The workhorse of the Server is the *pipeline*, in the previous example we create a simple http 1.1 pipeline that will response with 404 to any request.

CHAPTER 4

Client

The Client object holds the state and configuration data for a single abstract client. Depending on the concrete implementation the client could be connected to multiple servers in a cluster, or just a single server. It should be configured and instantiated by a XioClientBootstrap object.

```
XioClient client = new XioClientBootstrap(new NioEventLoopGroup())
  .setAddress(new InetSocketAddress("10.10.10.10", 443))
  .handler(new SimpleInboundChannelHandler())
  .build();
```

The handler defines how the client will interact with the remote server. By default clients will use HTTP as their application protocol.

CHAPTER 5

Core

CHAPTER 6

Configuration

6.1 Static Configuration

XIO strives to use static configuration values over hard coded constants whenever possible. Static configuration in XIO is done with Typesafe Config. XIO provides sensible defaults static configuration values.

```
xio {

  // default values for application limits
  applicationLimits {
    // maximum number of connections across all servers in the application
    maxConnections = 15000
  }

  // default values for server limits
  serverLimits {
    // maximum number of connections for a single server instance
    maxConnections = 500
    // maximum frame size per connection
    maxFrameSize = 9600
    // triggered when no read was performed for the specified period of time. Specify␣
↪0 to disable.
    maxReadIdleTime = 60seconds
    // triggered when no write was performed for the specified period of time.␣
↪Specify 0 to disable.
    maxWriteIdleTime = 60seconds
    // triggered when neither read nor write was performed for the specified period␣
↪of time. Specify 0 to disable.
    maxAllIdleTime = 60seconds
  }

  // default values for application settings
  applicationSettings {
    // location of the zookeeper cluster DEPRECATED
    zookeeperCluster = ""
    zookeeper {
      // location of the zookeeper cluster
      cluster = "localhost:2181"
      client {
        retry {
          // zookeeper client retry policy
          policy = RetryOneTime
```

```
          // policy must match one of the following sections:
          BoundedExponentialBackoffRetry {
            // see: https://curator.apache.org/apidocs/org/apache/curator/retry/
↪BoundedExponentialBackoffRetry.html
            baseSleepDuration = 2seconds
            maxSleepDuration = 10seconds
            maxRetries = 10
          }
          ExponentialBackoffRetry {
            // https://curator.apache.org/apidocs/org/apache/curator/retry/
↪ExponentialBackoffRetry.html
            baseSleepDuration = 2seconds
            maxRetries = 10
          }
          RetryForever {
            // https://curator.apache.org/apidocs/org/apache/curator/retry/
↪RetryForever.html
            sleepDuration = 2seconds
          }
          RetryNTimes {
            // https://curator.apache.org/apidocs/org/apache/curator/retry/
↪RetryNTimes.html
            n = 10
            sleepDuration = 2seconds
          }
          RetryOneTime {
            // https://curator.apache.org/apidocs/org/apache/curator/retry/
↪RetryOneTime.html
            sleepDuration = 2seconds
          }
          RetryUntilElapsed {
            // https://curator.apache.org/apidocs/org/apache/curator/retry/
↪RetryUntilElapsed.html
            maxElapsedDuration = 10seconds
            sleepDuration = 2seconds
          }
        }
      }
    }
    // number of boss threads to create
    bossThreads = 5
    // boss thread name format
    bossNameFormat = "xio-application-boss-%d"
    // number of worker threads to create
    workerThreads = 10
    // worker thread name format
    workerNameFormat = "xio-application-worker-%d"
    // settings for dynamic configuration manager
    configurationManager {
      ipFilter {
        // path to monitor for ip filter rules
        path = "/xio/ipFilterRules"
      }
      http1Filter {
        // path to monitor for http filter rules
        path = "/xio/http1FilterRules"
      }
    }
```

```
86      // settings for configuration update server
87      configurationUpdateServer {
88        // update server is disabled by default
89        enabled = false
90        // update server is bound to port 9999 on loopback by default
91        bindIp = 127.0.0.1
92        bindPort = 9999
93        // update server will coalesce changes and persist them every 5 seconds by␣
    ↪default
94        writeInterval = 5seconds
95      }
96      // settings for muxing client
97      requestMuxer {
98        messagesPerBatch = 100
99        drainMessageQInterval = 1millisecond
100       multiplierIncrementInterval = 500milliseconds
101       multiplierDecrementInterval = 750milliseconds
102       rebuildConnectionLoopInterval = 250milliseconds
103     }
104   }
105
106   serverSettings {
107     // servers bind to port 80 on loopback by default
108     bindIp = 127.0.0.1
109     bindPort = 80
110     // DEPRECATED
111     bossThreads = 5
112     // DEPRECATED
113     workerThreads = 10
114     // xio message logger is enabled by default
115     messageLoggerEnabled = true
116     // load self signed cert by default
117     tls { include classpath("tls.conf") }
118   }
119
120   applicationTemplate {
121     // application name defaults to blank
122     name = ""
123     limits = ${xio.applicationLimits}
124     settings = ${xio.applicationSettings}
125   }
126
127   serverTemplate {
128     // server name defaults to blank
129     name = ""
130     limits = ${xio.serverLimits}
131     settings = ${xio.serverSettings}
132   }
133
134   // example of how to build an xio application from templates
135   exampleApplication = ${xio.applicationTemplate} {
136     name = "example application"
137     servers {
138       exampleServer = ${xio.serverTemplate} {name = "example server"}
139     }
140   }
141
142   exampleServer = ${xio.serverTemplate} {name = "example"}
```

```
143
144   servers = [
145   #  ${exampleServer}
146   ]
147
148   testApplication = ${xio.applicationTemplate} {
149     name = "test application"
150     servers {
151       testServer = ${xio.serverTemplate} {
152         name = "test server"
153         settings {
154           bindPort = 0
155         }
156       }
157     }
158   }
159 }
```

6.2 Dynamic Configuration

6.2.1 Configuration Server

6.2.2 Configuration Client

CHAPTER 7

XIO Examples

7.1 example-trailhead

CHAPTER 8

XIO Logging

8.1 log4j-formatter

CHAPTER 9

Hacking xio

9.1 Maven

9.1.1 mvn

9.1.2 IntelliJ

9.2 Makefile

9.2.1 emacs

9.3 Building the docs

CHAPTER 10

Indices and tables

- genindex
- search