First, you really do not want to be managing the XML files.

Here's an example of using Jetty 9 with it's split of two directories ${jetty.home} and ${jetty.base}.

We follow the home vs base concepts that all other web containers follow.

Where home is the binaries, and base is your configuration.

You start by downloading the jetty-home archive (tarball or zip)

This is the most fundamental image for running standalone Jetty.

The older jetty-distribution archive is just jetty-home with the demo-base example added to it.

$ mkdir jetty-stuff

$ cd jetty-stuff

$ curl -O <https://repo1.maven.org/maven2/org/eclipse/jetty/jetty-home/9.4.30.v20200611/jetty-home-9.4.30.v20200611.tar.gz>

$ tar -zxvf jetty-home-9.4.30.v20200611.tar.gz

$ cd jetty-home-9.4.30.v20200611

$ export JETTY\_HOME=`jetty-stuff/jetty-home-9.4.30.v20200611`

At this point you have a jetty-stuff/jetty-home-9.4.30.v20200611/ directory.

**IMPORTANT:** Leave this directory alone, don't change it, don't edit it, don't add things to it, don't delete things from it, etc.

Consider this directory to be read-only.

The jetty-stuff/jetty-home-9.4.30.v20200611/ is your ${jetty.home} directory.

It has all of the pieces that make up the standalone jetty experience, but no configuration.

Now we have to make our ${jetty.base} directory.

This directory should not be nested inside or outside of the ${jetty.home} directory.

$ cd jetty-stuff

$ mkdir mybase

$ cd mybase

$ java -jar $JETTY\_HOME/start.jar --create-startd --add-to-start=http,deploy,webapp,threadpool

MKDIR : ${jetty.base}/start.d  
INFO  : webapp          transitively enabled, ini template available with --add-to-start=webapp  
INFO  : server          transitively enabled, ini template available with --add-to-start=server  
INFO  : security        transitively enabled  
INFO  : servlet         transitively enabled  
INFO  : http            initialized in ${jetty.base}/start.d/http.ini  
INFO  : threadpool      initialized in ${jetty.base}/start.d/threadpool.ini  
INFO  : bytebufferpool  transitively enabled, ini template available with --add-to-start=bytebufferpool  
INFO  : deploy          initialized in ${jetty.base}/start.d/deploy.ini  
MKDIR : ${jetty.base}/webapps  
INFO  : Base directory was modified

$ tree -F

.  
├── start.d/  
│   ├── deploy.ini  
│   ├── http.ini  
│   └── threadpool.ini  
└── webapps/

At this point you have a basic configured ${jetty.base} without any specific webapps deployed.

You can see your configuration by using --list-config ...

$ cd jetty-stuff/mybase

$ java -jar $JETTY\_HOME/start.jar --list-config

Java Environment:  
-----------------  
 java.home = /home/joakim/java/jvm/jdk-11.0.7+10 (null)  
 java.vm.vendor = AdoptOpenJDK (null)  
 java.vm.version = 11.0.7+10 (null)  
 [java.vm.name](http://java.vm.name/) = OpenJDK 64-Bit Server VM (null)  
 [java.vm.info](http://java.vm.info/) = mixed mode (null)  
 [java.runtime.name](http://java.runtime.name/) = OpenJDK Runtime Environment (null)  
 java.runtime.version = 11.0.7+10 (null)  
 java.io.tmpdir = /tmp (null)  
 user.dir = /home/joakim/demo/jetty-stuff/mybase (null)  
 user.language = en (null)  
 user.country = US (null)  
  
Jetty Environment:  
-----------------  
 jetty.version = 9.4.30.v20200611  
 jetty.tag.version = master  
 jetty.home = /home/joakim/demo/jetty-stuff/jetty-home-9.4.30.v20200611  
 jetty.base = /home/joakim/demo/jetty-stuff/mybase  
  
Config Search Order:  
--------------------  
 <command-line>  
 ${jetty.base} -> /home/joakim/demo/jetty-stuff/mybase  
 ${jetty.home} -> /home/joakim/demo/jetty-stuff/jetty-home-9.4.30.v20200611  
  
  
JVM Arguments:  
--------------  
 (no jvm args specified)  
  
System Properties:  
------------------  
 (no system properties specified)  
  
Properties:  
-----------  
 java.version = 11.0.7  
 java.version.major = 11  
 java.version.micro = 7  
 java.version.minor = 0  
 java.version.platform = 11  
 jetty.base = /home/joakim/demo/jetty-stuff/mydemobase  
 jetty.base.uri = file:///home/joakim/demo/jetty-stuff/mydemobase  
 jetty.home = /home/joakim/demo/jetty-stuff/jetty-home-9.4.30.v20200611  
 jetty.home.uri = file:///home/joakim/demo/jetty-stuff/jetty-home-9.4.30.v20200611  
  
Jetty Server Classpath:  
-----------------------  
Version Information on 11 entries in the classpath.  
Note: order presented here is how they would appear on the classpath.  
      changes to the --module=name command line options will be reflected here.  
 0:                    3.1.0 | ${jetty.home}/lib/servlet-api-3.1.jar  
 1:                 3.1.0.M0 | ${jetty.home}/lib/jetty-schemas-3.1.jar  
 2:         9.4.30.v20200611 | ${jetty.home}/lib/jetty-http-9.4.30.v20200611.jar  
 3:         9.4.30.v20200611 | ${jetty.home}/lib/jetty-server-9.4.30.v20200611.jar  
 4:         9.4.30.v20200611 | ${jetty.home}/lib/jetty-xml-9.4.30.v20200611.jar  
 5:         9.4.30.v20200611 | ${jetty.home}/lib/jetty-util-9.4.30.v20200611.jar  
 6:         9.4.30.v20200611 | ${jetty.home}/lib/jetty-io-9.4.30.v20200611.jar  
 7:         9.4.30.v20200611 | ${jetty.home}/lib/jetty-security-9.4.30.v20200611.jar  
 8:         9.4.30.v20200611 | ${jetty.home}/lib/jetty-servlet-9.4.30.v20200611.jar  
 9:         9.4.30.v20200611 | ${jetty.home}/lib/jetty-webapp-9.4.30.v20200611.jar  
10:         9.4.30.v20200611 | ${jetty.home}/lib/jetty-deploy-9.4.30.v20200611.jar  
  
Jetty Active XMLs:  
------------------  
 ${jetty.home}/etc/jetty-bytebufferpool.xml  
 ${jetty.home}/etc/jetty-threadpool.xml  
 ${jetty.home}/etc/jetty.xml  
 ${jetty.home}/etc/jetty-webapp.xml  
 ${jetty.home}/etc/jetty-deploy.xml  
 ${jetty.home}/etc/jetty-http.xml

This shows you that the XML is used from ${jetty.home}, as-is.

As for the modules I picked "http,deploy.webapp,threadpool" those are just some of the predefined modules present on jetty-home.

You can list the available modules with --list-modules

$ cd jetty-stuff/mybase

$ java -jar $JETTY\_HOME/start.jar --list-modules

Some other common modules are ..

 \* `resources` for making a `${jetty.base}/resources/` directory available on the server classpath

 \* `ext` for making a ${jetty.base}/lib/ext/\*.jar files available on the server classpath.

 \* `console-capture` for capturing all console output to a logging file

 \* `annotations` for enabling annotation scanning of bytecode for Servlet annotations

 \* `gzip` for enabling gzip compression of response, and gzip decompression of requests.

 \* `https` for enabling an HTTPS connector

 \* `http2` for enabling an HTTP2 connector

The modules themselves can have ...

 \* Dependencies on other modules

 \* Properties for configuring the module

 \* XML (which is properly ordered based on other XML)

 \* Required libraries (downloaded if not present, after a license check with you)

 \* Required directories (created if not present)

You can use the `--add-to-start=<modules>` at any point to add more modules to your configuration.

To remove a module, just delete the file in ${jetty.base}/start.d/ that enables the module.

Note that only modules you specifically enable are present in the start.d directory, transitive modules from those specifically mentioned are also enabled, but not represented by a file in the start.d directory.

An example of that would be the "server" module.

You can see this at the end of the output from `--list-modules`

$ cd jetty-stuff/mybase

$ java -jar $JETTY\_HOME/start.jar --list-modules

...(snip)...

Enabled Modules:  
================  
    0) bytebufferpool  transitive provider of bytebufferpool for server  
                       init template available with --add-to-start=bytebufferpool  
    1) threadpool      ${jetty.base}/start.d/threadpool.ini  
    2) server          **transitive provider of server** for http  
                       **transitive provider of server** for servlet  
                       init template available with --add-to-start=server  
    3) security        transitive provider of security for webapp  
    4) servlet         transitive provider of servlet for webapp  
    5) webapp          transitive provider of webapp for deploy  
                       init template available with --add-to-start=webapp  
    6) deploy          ${jetty.base}/start.d/deploy.ini  
    7) http            ${jetty.base}/start.d/http.ini

The text "init template available" means that there are properties that can be configured for that module.

You can add `server` after the fact with ...

$ cd jetty-stuff/mybase

$ java -jar $JETTY\_HOME/start.jar --list-modules

INFO  : server          initialized in ${jetty.base}/start.d/server.ini  
INFO  : Base directory was modified

Now let's take a look at your configuration details, and adjust some threadpool numbers.

$ cat start.d/threadpool.ini   
# ---------------------------------------   
# Module: threadpool  
# Enables the Server thread pool.  
# ---------------------------------------   
--module=threadpool  
  
  
### Server Thread Pool Configuration  
## Minimum Number of Threads  
#jetty.threadPool.minThreads=10  
  
## Maximum Number of Threads  
#jetty.threadPool.maxThreads=200  
  
## Number of reserved threads (-1 for heuristic)  
# jetty.threadPool.reservedThreads=-1  
  
## Thread Idle Timeout (in milliseconds)  
#jetty.threadPool.idleTimeout=60000  
  
## Whether to Output a Detailed Dump  
#jetty.threadPool.detailedDump=false

Lets edit that file and uncomment the minThreads and detailedDump and set them to non-default values.

Such as minThreads of 16 and detailedDump=true

$ edit start.d/threadpool.ini

$ java -jar $JETTY\_HOME/start.jar --list-config | grep thread  
 jetty.threadPool.detailedDump = false  
 jetty.threadPool.minThreads = 16  
 ${jetty.home}/etc/jetty-threadpool.xml

Now copy some of your webapps into place ...

$ cp /path/to/myapp.war jetty-stuff/mybase/webapps/

And run Jetty with server dump enabled (to see what the server looks like after it has loaded)

The use of `jetty.server.dumpAfterStart=true` is just setting a property on the command line (vs declaring it in the start.d/\*.ini files)

You can test any property from the command line before you commit to using it in your configuration files.

You can even override the property in the configuration files with the command line.

The `Config Search Order` from `--list-config` tells you how things are found.

This is not limited to just properties, but also applies to libraries and xml files!

Example: If you provide a highly customized XML your ${jetty.base}/etc/ it will be used over the one in ${jetty.home}/etc/

$ cd jetty-stuff/mybase

$ java -jar $JETTY\_HOME/start.jar jetty.server.dumpAfterStart=true

...(snip lots of output)...

Now you have a running server instance based on $JETTY\_HOME with configuration from ${jetty.base} (which is jetty-stuff/mybase) here.

Now, why bother with having these directories separate?

Why does ${jetty.base} exist?

As I mentioned above, i'll get to it.

Easily change versions of Jetty!

Example with what you have above ...

$ jetty-stuff

$ curl -O <https://repo1.maven.org/maven2/org/eclipse/jetty/jetty-home/9.4.29.v20200521/jetty-home-9.4.29.v20200521.tar.gz>

$ tar -zxvf jetty-home-9.4.29.v20200521.tar.gz

$ cd mybase

$ java -jar ../jetty-home-9.4.29.v20200521/start.jar

Done, you've just changed versions of Jetty, and had no configurations to edit / copy over / etc ...

You can even go to Jetty 10 with minimal hassle.

(Going to Jetty 11 would require you to use the new Jakarta EE 9 namespace in your war files though)

In short, the recommended approach is to not manage XML, use Properties and modules.

Joakim Erdfelt / [joakim@xxxxxxxxxxx](mailto:joakim@xxxxxxxxxxx)