Validate Configuration against Build Artifacts:

An Essential Step for your Build Process

by
Harold Fortuin, Ph.D.
Principal Software Engineer
Fortuitous Consulting Services, Inc.

JavaEE projects rely on (XML) config files which reference project-specific Java .class'es.

Examples...

Common Config Files

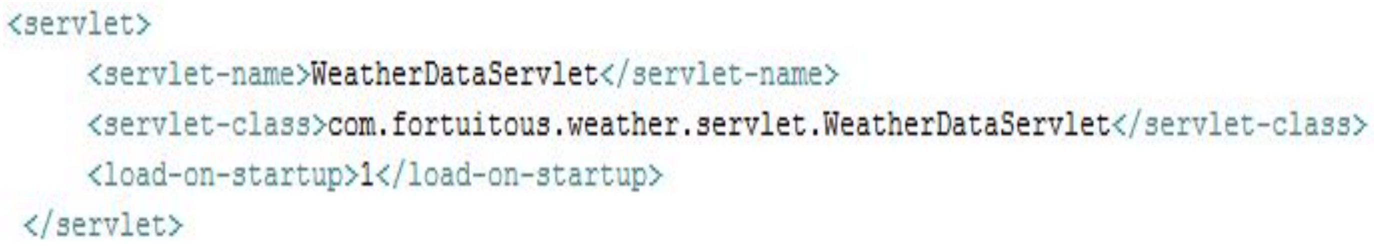
web.xml - in **all** servlets/JSP WARs
(until Servlets 3.0 - then optional)

```
<servlet-class>com.ms.BillsServlet
</servlet-class>
<filter-class>com.ms.LoginFilter
</filter-class>
```

Common Config Files

web.xml - in **all** servlets/JSP WARs

(until Servlets 3.0 - then optional)



```
<servlet>
    <servlet-name>WeatherDataServlet</servlet-name>
    <servlet-class>com.fortuitous.weather.servlet.WeatherDataServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
 </servlet>
```

Common Config Files

beans.xml - **Spring** bean for database access

```
<bean id="Organization"
class="com.fortuitous.weather.Organization" />
```

Typical JavaEE build processes include the following steps in order...

- compile Java (and precompile JSPs) into classes
- bundle classes and configuration files into binaries (JARs, modular JARs, WARs, and/or EARs)
- copy WAR/EAR/JARs into deployment directories

Then after deployment:

- Start the JavaEE Application Server instance (if needed)
- Execute tests

before exposing app to internal/external customers

Avoid time-consuming errors in builds and deployments by catching them as early as possible

In my experience, a number of build validation steps are done post-compile on various JavaEE projects.

But certainly one is missing...

The step to Validate the references to Java .class'es made in configuration files!

Without such
validation,
app servers
will fail
at runtime…

Either:

- fail to start at all
- fail under particular use case(s)

Wasting valuable resources!

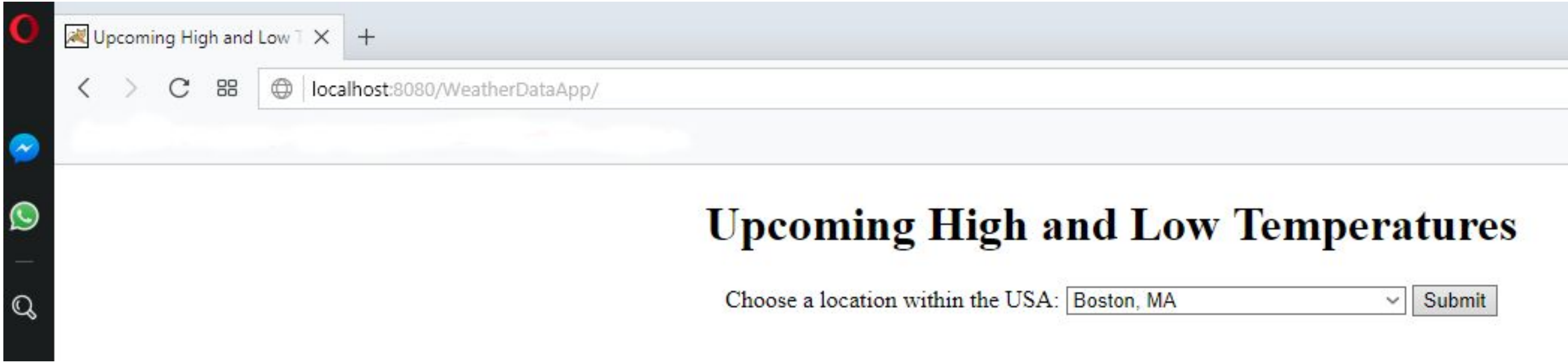
So I wrote a Java tool to be run after your project's compile to do this validation, now available at:

https://github.com/tekniqul/validateConfigurationAgainstBuild

For now, configured to validate build configuration in WeatherDataApp available from

https://github.com/tekniqul/WeatherDataApp

InputValues.jsp (with JSTL)

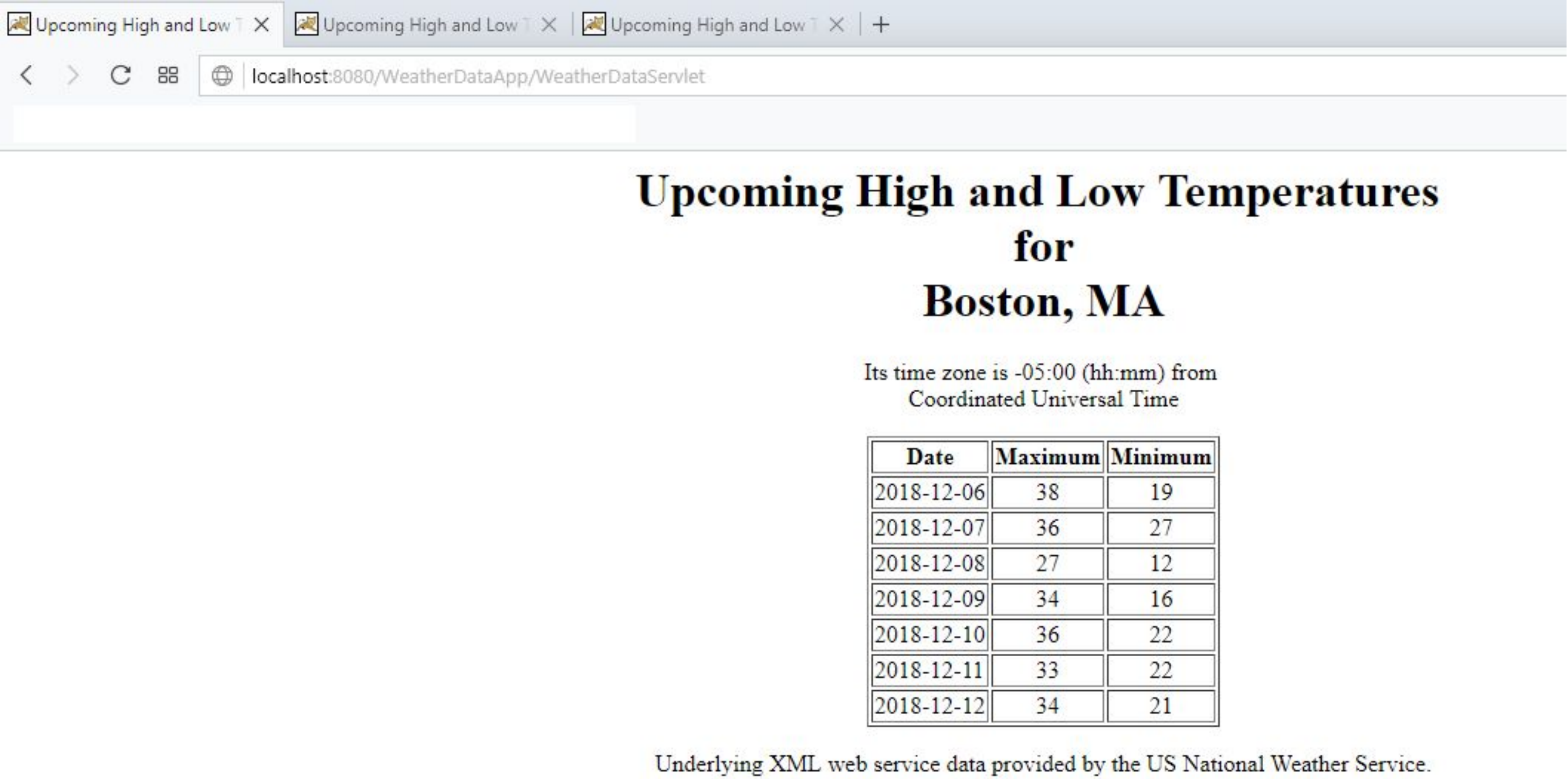


Upcoming High and Low Temperatures

Choose a location within the USA: Boston, MA

https://github.com/tekniqul/WeatherDataApp

Results.jsp (with JSTL)



Upcoming High and Low Temperatures for Boston, MA

Its time zone is -05:00 (hh:mm) from Coordinated Universal Time

Date	Maximum	Minimum
2018-12-06	38	19
2018-12-07	36	27
2018-12-08	27	12
2018-12-09	34	16
2018-12-10	36	22
2018-12-11	33	22
2018-12-12	34	21

Underlying XML web service data provided by the US National Weather Service.

ValidateConfigByClass.bat provides example command-line arguments of expected filetypes

(.jar built with Java 10.0.2)

```
java -jar ValidateConfigByClass.jar classXPaths.xml configurationFiles.xml
```

1st argument: classXPaths.xml - contains *n* xPath elements per containing XML filetype element

```
<configFiles>
   <web-xml>
      <xPath>//servlet-class</xPath>
      <ignorePackage>org.springframework</ignorePackage>
…
   <beans-xml>
      <xPath>//bean/@class</xPath>
…
```

```
<configFiles>
    <web-xml>
        <ignorePackage>org.springframework</ignorePackage>
```

n ignorePackage elements per containing XML filetype element -

Bypass validation for these classes in your 3rd party .jar's

2nd argument: configurationFiles.xml - contains *n* configurationFile elements, each with a location subelement and a classRootDirectory subelement

```
<configurationFile type="web-xml">
   <location>WeatherDataApp_WAR/WEB-INF/web.xml</location>
   <classRootDirectory>WeatherDataApp_WAR/WEB-INF/classes</classRootDirectory>
</configurationFile>
<configurationFile type="beans-xml">
   <location>WeatherDataApp_WAR/WEB-INF/beans.xml</location>
...
```

Run ValidateConfigByClass.bat

(logs to console & log.html)

```
java -jar ValidateConfigByClass.jar classXPaths.xml configurationFiles.xml
```

Review BAD_ValidateConfigByClass.bat

(logs to console & log.html)

```
java -jar ValidateConfigByClass.jar classXPaths.xml BAD_configurationFiles.xml
```

2nd argument: BAD_configurationFiles.xml -

Refers to BAD_ XML files in WeatherDataApp with misspelled .class references - deliberately test failures to validate

```
<configurationFile type="web-xml">
 <location>WeatherDataApp_WAR/WEB-INF/BAD_web.xml</location>
…
<configurationFile type="beans-xml">
   <location>WeatherDataApp_WAR/WEB-INF/BAD_beans.xml</location>
```

v1.0 Design Considerations

- Entirely driven by configuration
- Parser knows nothing of given app XMLs
- configurationFiles.xml separates deployment directories vs. classXPaths.xml for XPath config

Maven Integration: exec-maven-plugin

```
<plugin>
        <groupId>org.codehaus.mojo</groupId>
        <artifactId>exec-maven-plugin</artifactId>
        <version>1.6.0</version>
        <executions>
          <execution>
            <goals>
              <goal>java</goal>
        ...
        <configuration>
            <mainClass>com.fortuitous.buildValidation.ValidateConfigVsClass.Main</mainClass>
            <!-- by default it is true as well -->
            <includeProjectDependencies>true</includeProjectDependencies>
            <arguments>
              <argument>classXPaths.xml</argument>
              <argument>configurationFiles.xml</argument>
       ...
```

Next Steps?

- Add JSP validation
- Upload maven integration
- Increase performance with StAX parser + common XML config know-how “baked in”?

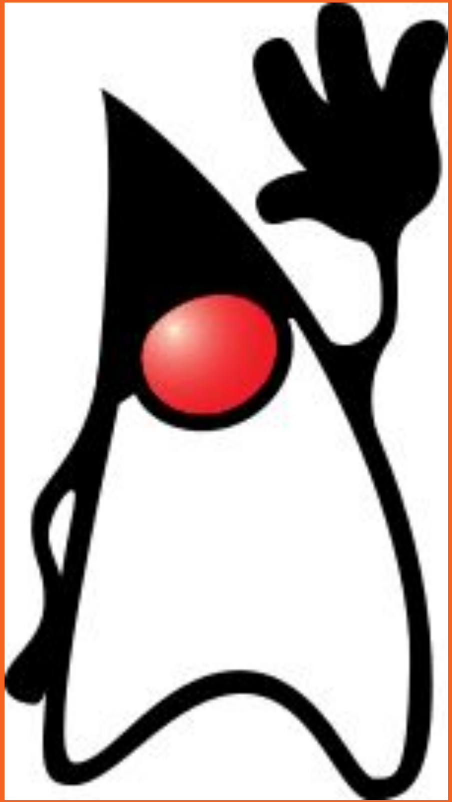
Thanks!

shutterstock.com • 432906004

"WE DON'T WANT ANOTHER INCIDENT."

THAT'S A GREAT IDEA!

OH, NO... NO... NO... TELL ME IT ISN'T SO!



Bye now!

Harold Fortuin, Ph.D.
Principal Software Engineer
Fortuitous Consulting Services, Inc.

http://fortuitous-consulting.biz
http://www.github.com/tekniqul