

[Bila Bertemu LinkageError dalam Java Web Service](#)

Jika kita mempelajari Java yang didalamnya terdapat banyak sekali cabang, maka kita akan menemui salah satunya yaitu Java Web Service atau yang lebih terbiasa disingkat menjadi JWS. Seiring dengan kenaikan versi dari JDK-nya itu sendiri maka file library yang diperlukan untuk membangun JWS pun akan mengalami kenaikan versi. Dan dengan kenaikan versi tersebut tidak serta merta library itu akan mengalami kematangan tanpa disertai oleh error dan bug.

Dalam kesempatan ini saya akan sedikit membahas error yang ditemui jika kita membangun aplikasi Web-Service menggunakan JWS, tapi perlu diketahui error ini baru ditemukan pada JDK versi 5 dan 6, tetapi jika kita membangun aplikasi tersebut menggunakan JDK 1.4 error ini tidak ditemukan. Pesan error tersebut berupa printStackTrace yang dapat terlihat dari console atau runtime output dari IDE yang digunakan. Dengan bentuk pesan error sebagai berikut:

```
Exception in thread "main" java.lang.LinkageError:
JAXB 2.0 API is being loaded from the bootstrap classloader, but this RI (from
jar:file:/C:/Program%20Files/glassfish-v2/lib/webservices-
rt.jar!/com/sun/xml/bind/v2/model/impl/ModelBuilder.class) needs 2.1 API.
Use the endorsed directory mechanism to place jaxb-api.jar in the bootstrap
classloader. (See http://java.sun.com/j2se/1.5.0/docs/guide/standards/)

at com.sun.xml.bind.v2.model.impl.ModelBuilder.<clinit>(ModelBuilder.java:172)
at
com.sun.xml.bind.v2.runtime.JAXBContextImpl.getTypeInfoSet(JAXBContextImpl.java:
422)
at com.sun.xml.bind.v2.runtime.JAXBContextImpl.<init>(JAXBContextImpl.java:286)
at com.sun.xml.bind.v2.ContextFactory.createContext(ContextFactory.java:139)
at com.sun.xml.bind.v2.ContextFactory.createContext(ContextFactory.java:117)
at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:39)
at
sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.jav
a:25)
at java.lang.reflect.Method.invoke(Method.java:597)
at javax.xml.bind.ContextFinder.newInstance(ContextFinder.java:210)
at javax.xml.bind.ContextFinder.find(ContextFinder.java:368)
at javax.xml.bind.JAXBContext.newInstance(JAXBContext.java:574)
at javax.xml.bind.JAXBContext.newInstance(JAXBContext.java:522)
at com.sun.xml.ws.spi.ProviderImpl$2.run(ProviderImpl.java:220)
at com.sun.xml.ws.spi.ProviderImpl$2.run(ProviderImpl.java:218)
at java.security.AccessController.doPrivileged(Native Method)
at com.sun.xml.ws.spi.ProviderImpl.getEPRJaxbContext(ProviderImpl.java:217)
at com.sun.xml.ws.spi.ProviderImpl.<clinit>(ProviderImpl.java:88)
at sun.reflect.NativeConstructorAccessorImpl.newInstance0(Native Method)
at
sun.reflect.NativeConstructorAccessorImpl.newInstance(NativeConstructorAccessori
mpl.java:39)
at
sun.reflect.DelegatingConstructorAccessorImpl.newInstance(DelegatingConstructorA
ccessorImpl.java:27)
at java.lang.reflect.Constructor.newInstance(Constructor.java:513)
at java.lang.Class.newInstance0(Class.java:355)
at java.lang.Class.newInstance(Class.java:308)
at javax.xml.ws.spi.FactoryFinder.newInstance(FactoryFinder.java:36)
```

```
at javax.xml.ws.spi.FactoryFinder.find(FactoryFinder.java:95)
at javax.xml.ws.spi.Provider.provider(Provider.java:76)
at javax.xml.ws.Service.<init>(Service.java:57)
at com.siriuserp.ws.SiriusWSService.<init>(SiriusWSService.java:46)
at xxx.test.client.ReadData.main(ReadData.java:1001)
```

Dari error `printStackTrace` diatas dapat kita lihat pada enam baris pertama, disitu diterangkan penyebab dari error tersebut, yaitu disaat aplikasi JWS melakukan startup maka secara default aplikasi tersebut akan mencari library – library pendukung, diantara library tersebut diperlukan dua buah library dengan nama file `jaxb-api.jar` dan `jaxws-api.jar`, secara default `webservice-rt.jar` memerlukan library `jax***.jar` tersebut yang bernomor versi 2.1 sedangkan java bootstrap class loader hanya menemukan yang bernomor versi 2.0 atau bahkan tidak menemukan sama sekali file – file tersebut. Masih dalam `printStackTrace` tersebut diterangkan bahwa masalah tersebut dapat diperbaiki dengan mekanisme “endorsed” yaitu kita dapat membuat sebuah directory dibawah directory sebagai berikut “`<JAVA_HOME>/jre/lib/endorsed`” jika kita masih bekerja dalam tahap developing, tetapi jika sudah masuk ke tahap produksi atau aplikasi yang sudah berjalan maka directory sebagai berikut “`<JAVA_HOME>/lib/endorsed`”, kemudian masukan dua buah file *.jar tersebut ke dalam directory, lalu lakukan restart aplikasi java atau restart operating system. Kemudian aplikasi tersebut dapat dicoba kembali.

Atau jika programmer tidak ingin merubah konfigurasi basic dari JDK maka dapat dilakukan dengan teknik lain yaitu menambahkan java option disaat mengeksekusi aplikasi java tersebut dengan perintah sebagai berikut:

```
WSIMPORT_OPTS=-Djava.endorsed.dirs=%JAXWS_HOME%/lib
WSGEN_OPTS=-Djava.endorsed.dirs=%JAXWS_HOME%/lib
```

Dengan asumsi bahwa isi dari `JAXWS_HOME` diarahkan terhadap kedua file `jaxb-api.jar` dan `jaxws-api.jar` tersebut disimpan. Atau untuk melihat petunjuk resmi yang dikeluarkan oleh sun mengenai mekanisme endorsed dapat melihatnya dalam link berikut <http://java.sun.com/j2se/1.5.0/docs/guide/standards/>. Sedangkan untuk download file jar tersebut selain yang berasal dari situs resmi sun dapat juga dijumpai dalam link berikut <http://www.java2s.com/Code/Jar/CatalogJar.htm>.

Mungkin saat ini hanya sekian ilmu yang bisa dibagi dengan pembaca semoga bermanfaat, jika ada kekurangan atau masukan yang dapat meningkatkan kemampuan jangan sungkan untuk memberikan komentar.