

Commenti al Laboratorio 3. XML con SAX

Metodo `ContentHandler.characters()`

Il metodo `characters(char[] ch, int start, int length)`, che viene ridefinito in `DefaultHandler` e nelle classi che la estendono, serve per scegliere quale azione fare quando viene incontrato un nodo di tipo `CDATA`.

Il parametro `ch` è un buffer di caratteri che contiene tutto il documento XML: è quindi necessario utilizzare gli interi `start` e `length` per andare a prendere la porzione di `CDATA` relativa al nodo esaminato.

Esercizio: provate a modificare `CompRoute` e a stampare tutto `ch` senza effettuare il controllo su `start` e `length`.

Estendere le classi e implementare le interfacce

Uno degli errori più comuni riscontrati è la mancata estensione di classi e l'implementazione di interfacce. Si tratta inoltre di uno degli errori più difficili da individuare, specie quando si gestiscono progetti molto grandi.

Cercate quindi di fare molta attenzione nella definizione della firma della classe e dei metodi.

Ad esempio, togliendo `extends DefaultHandler` dalla firma di `CompRoute`, si ottiene il seguente errore:

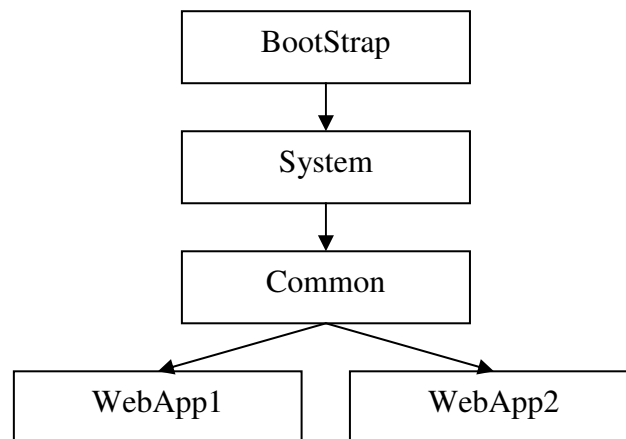
```
C:\...\lab3\CompRoute.java:34: cannot find symbol
symbol   : method parse(java.lang.String,XMLRoute.CompRoute)
location: class javax.xml.parsers.SAXParser
        sp.parse("xml/itinerario.xml", me);
```

Laboratorio 4pre. Tomcat (30/10/2007)

In questo laboratorio prenderemo confidenza con l'uso del servlet container Tomcat e proveremo a fare il deploy di una semplice web application.

Riferimento: Appendice A delle dispense.

Nota: a partire da Tomcat 6.0, il meccanismo del class loading è stato semplificato: i class loader Common, Catalina e Shared sono stati riuniti in un unico class loader Common, per cui la nuova gerarchia risulta essere:



Nota: nella directory `TOMCAT_HOME\server\webapps\manager` si trova la web application che implementa il manager. Nel descrittore `web.xml` di questa webapp si può notare l'uso del ruolo `manager` definito in `tomcat-users.xml`.

Esercizio 1: scaricate il file `EchoServlet.zip` ed effettuate il deploy della webapp contenuta al suo interno, prima off-line copiando la struttura delle directory e quindi on-line, generando un file `war` da riga di comando con il tool `jar`.

Esercizio 2: utilizzando NetBeans ed il server Tomcat bundled, create una webapp contenente una servlet che stampa l'ora corrente. Provate a modificarla aggiungendo la possibilità di specificare un parametro attraverso l'URL (ad esempio il nome dell'utente). Ad esempio, all'indirizzo:

```
http://localhost:8084/HelloWorld/HelloWorldServlet?name=Alessio
```

l'output dovrebbe essere qualcosa come:

```
Hello Alessio, it's 22:16:53.
```

Suggerimento: per ottenere la stringa contenente l'orario corrente, utilizzate il seguente codice:

```
java.text.SimpleDateFormat formatter =  
    new java.text.SimpleDateFormat("HH:mm:ss");  
java.util.Date orario = new java.util.Date();  
String orarioStringa = formatter.format(orario);
```

Effettuate varie prove e osservate il comportamento del HTTP Monitor integrato in NetBeans.