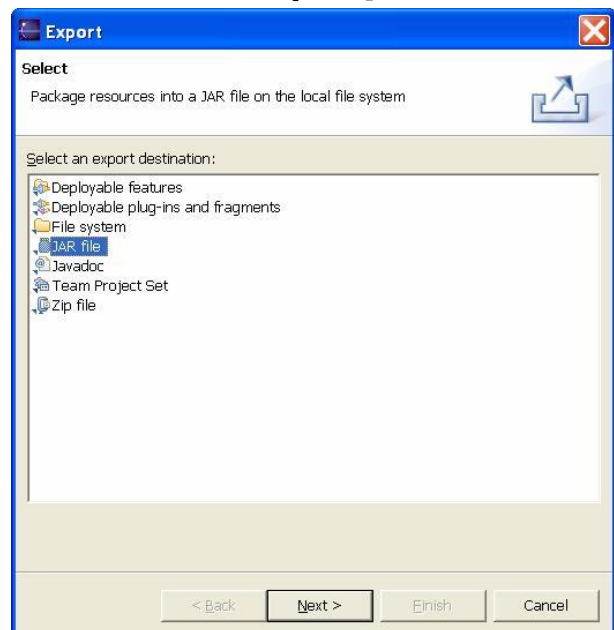


# Exportieren

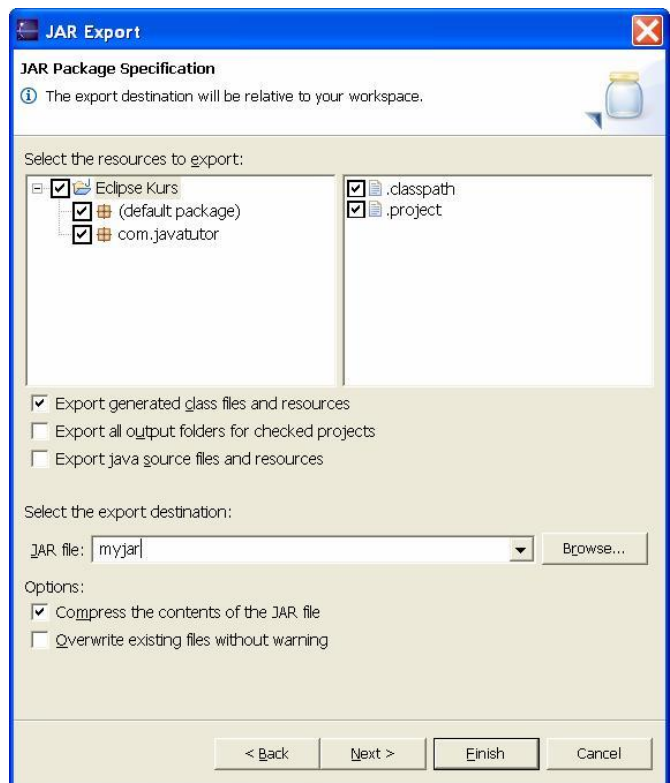
## Export-Möglichkeiten

- ☞ Eclipse kann ein Projekt in unterschiedlichen Varianten exportieren.
- ☞ Um den Export zu beginnen, wählt man **File | Export...**
- ☞ Interessant sind
  - ▶ Java-Archive (Jar)
  - ▶ Java-Dokumentation



# Java-Archive erstellen (1)

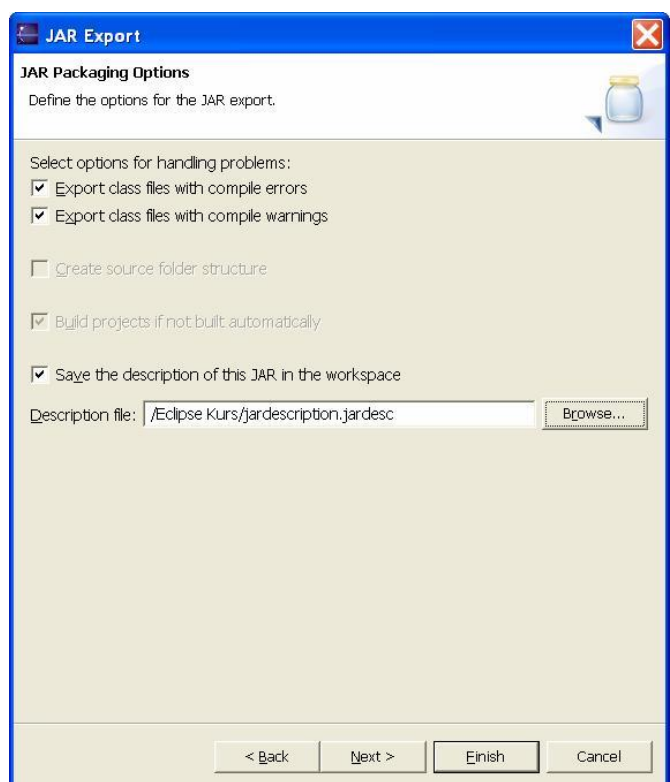
- ☞ Im Fall der Java-Archive lässt sich auswählen, aus welchem Projekt welche Dateien in einem Java-Archiv zusammengefasst werden sollen.
- ☞ Von den Angaben im Dialog muss lediglich der Dateiname des generierten Archivs angegeben werden.



3

# Java-Archive erstellen (2)

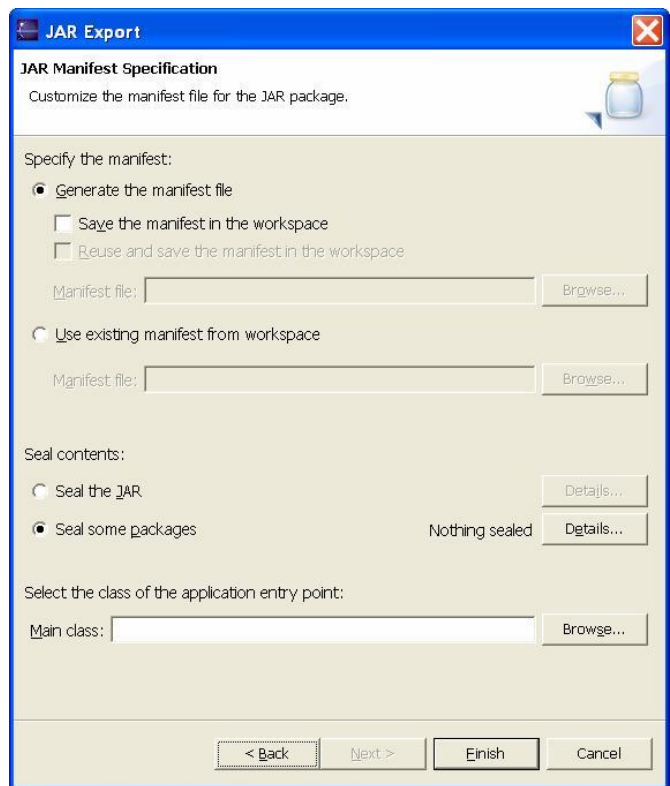
- ☞ Das nächste Dialog ist wenig interessant.



4

# Java-Archive erstellen (3)

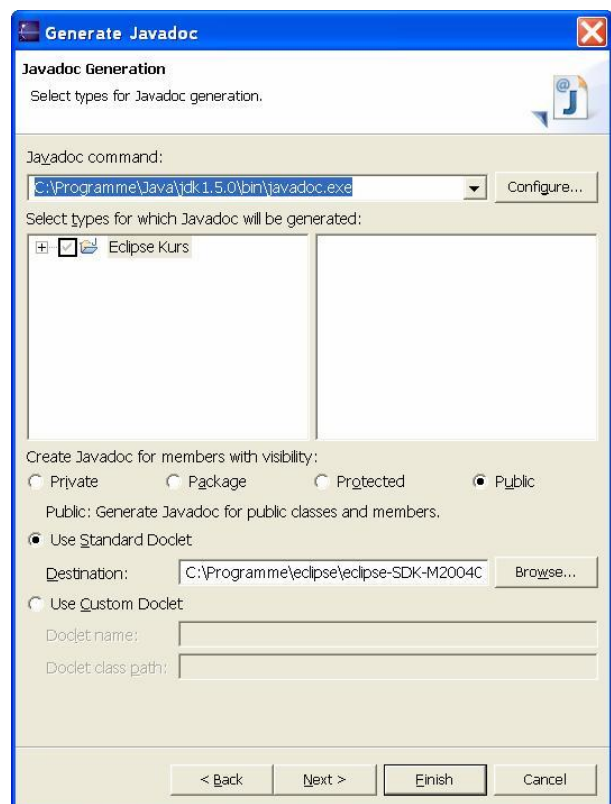
- ☞ Der Dialog ist schon interessanter!
- ☞ Oftmals müssen externe Manifeste mit integriert werden. Dann kann man dieses Manifest angeben.
- ☞ Das Paket lässt sich als Gesamtheit schützen (**seal the jar**), oder auch einige Teile.
- ☞ Eine **Main class** lässt sich auswählen, damit sie per Doppelklick gestartet werden kann.



5

# API-Dokumentation

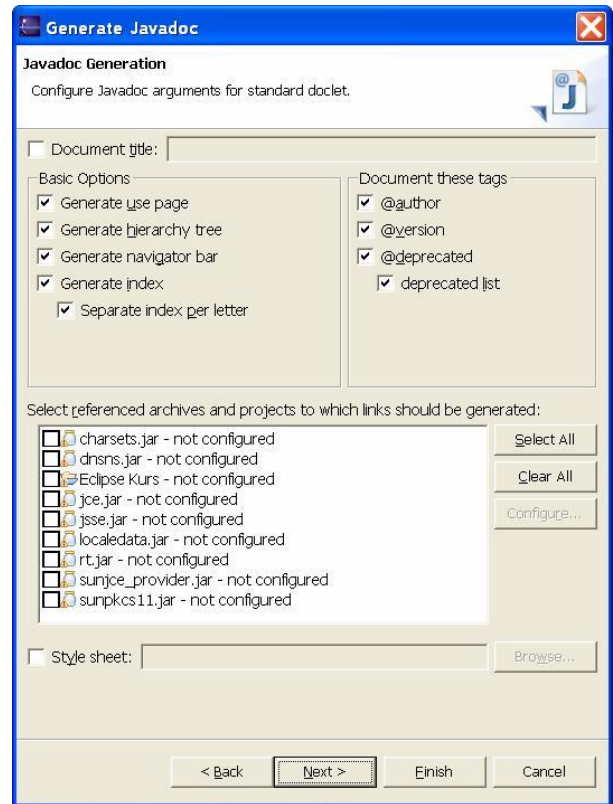
- ☞ Die JavaDoc lässt sich nur dann erstellen, wenn statt dem JRE das SDK eingebunden wurde.
  - ▶ Nur das liefert das Tool javadoc mit.
- ☞ Man bestimmt, welche Elemente mit welcher Sichtbarkeit aufgenommen werden.
- ☞ Ein alternatives Doclet lässt sich optional festlegen.



6

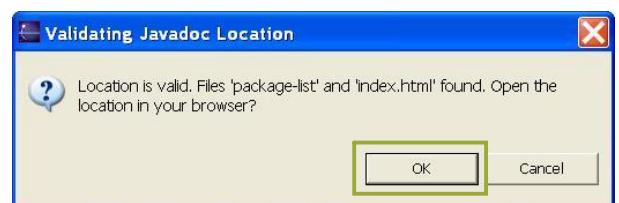
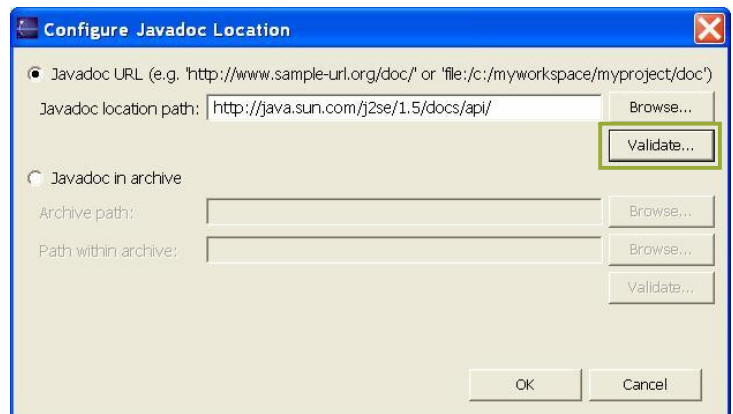
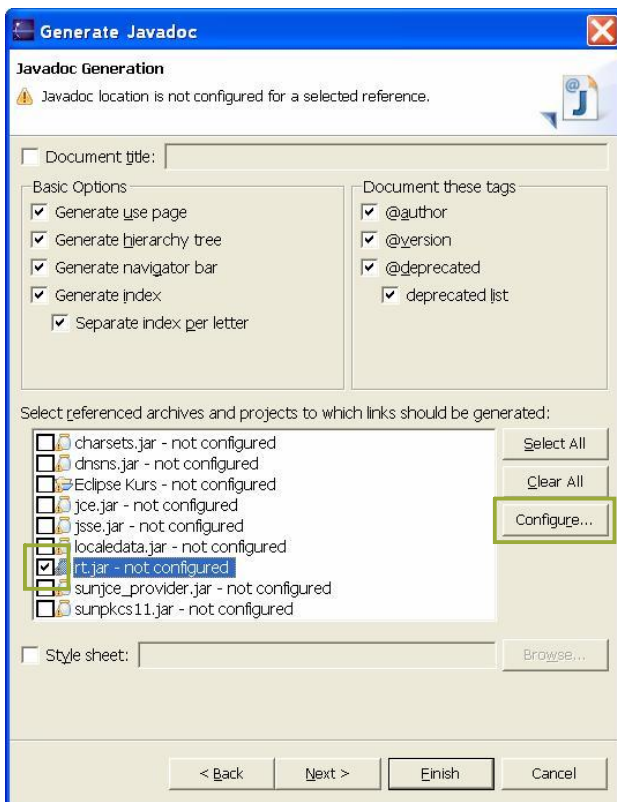
# Der Dialog Generate Javadoc (1)

- ☞ Die generierten HTML-Dateien können unterschiedlich aufbereitet werden.
- ☞ Es können unterschiedliche Seiten mit generierte werden.
- ☞ Für ein einheitliches Corporate-Identity lässt sich eine CSS-Datei für die HTML-Dateien angeben.



7

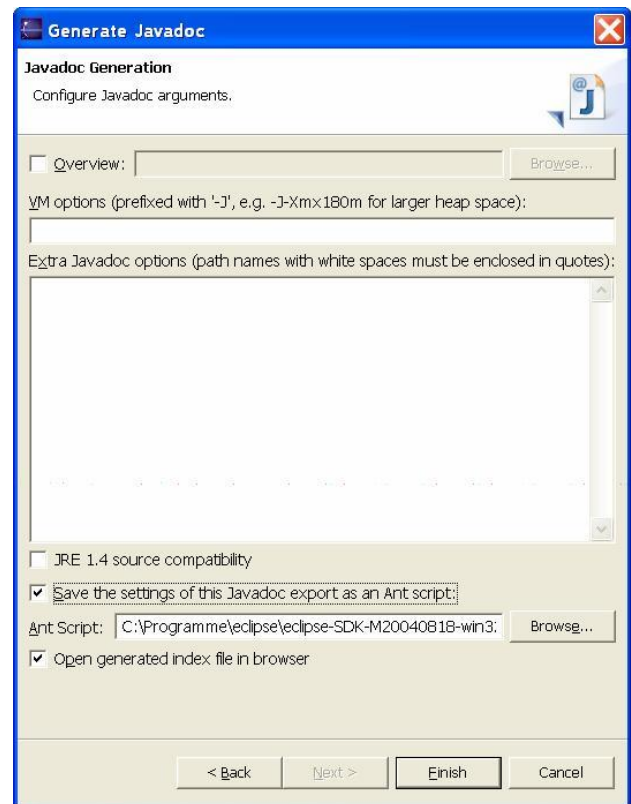
# JavaDoc auf die Sun-Klassen



8

# Der Dialog Generate Javadoc (2)

- Zusätzliche Argumente für das Kommandozeilenprogramm javadoc können festgelegt werden, doch das ist selten.
- Praktischer ist es schon, die Angaben in einem Ant-Skript festzuhalten.
  - ▶ Dann kann man leicht wieder die Dokumentation mit den gleichen Angaben durchführen.



9

## Generiertes Ant-Skript

```
<?xml version="1.0" encoding="UTF-8"?>
<project default="javadoc">
<target name="javadoc">
<javadoc access="public" author="true" classpath="."
  destdir="doc" nodeprecated="false"
  nodeprecatedlist="false" noindex="false"
  nonavbar="false" notree="false"
  packagenames="com.javatutor" sourcepath="."
  splitindex="true" use="true" version="true">
<link href="http://java.sun.com/j2se/1.5/docs/api/" />
</javadoc>
</target>
</project>
```

10

# Ergebnis im Browser

The screenshot shows a Mozilla Firefox browser window with the title "Person - Mozilla Firefox". The address bar contains the file path: `file:///C:/Programme/eclipse/eclipse-SDK-M20040818-win32/eclipse/workspace/Eclipse%20Kurs/doc/index.html?com:j`. The browser displays the Javadoc for the `Person` class, which is part of the `com.javatutor` package and extends `java.lang.Object`.

**All Classes**  
[Person](#)

## Class Person

```
java.lang.Object
└─ com.javatutor.Person
```

---

```
public class Person
extends Object
```

**Author:**  
Christian Ullenboom TODO To change the template for this generated type comment go to Window - Preferences - Java - Code Style - Code Templates

---

### Constructor Summary

<a href="#">Person()</a>
--------------------------

---

### Method Summary

static void	<a href="#">main</a> (String[] args)
-------------	--------------------------------------

---

### Methods inherited from class java.lang.Object

<a href="#">equals</a> , <a href="#">getClass</a> , <a href="#">hashCode</a> , <a href="#">notify</a> , <a href="#">notifyAll</a> , <a href="#">toString</a> , <a href="#">wait</a> , <a href="#">wait</a> , <a href="#">wait</a>
---

Done