

Erratum zu „Oracle 8i und Java“

Dieser Artikel wurde geschrieben, um die Inhalte des Buches auf die derzeit erhältliche Version 2.0 des „Oracle JDeveloper 2.0“ anzupassen. Zum Zeitpunkt der Erstellung des Buches stand diese Version kurz vor ihrer Auslieferung. Da sich die Menüführung und auch das Layout einiger Komponenten, insbesondere des „Deployment Wizard“ (dt. „Weitergabe-Experte“) inzwischen etwas geändert haben, wurden die relevanten Teile der Beschreibung des Werkzeuges hier noch einmal mit aktuellen Bildschirmgrafiken einbezogen.

Steven Ponndorf und Dr. Wolf-Gert Matthäus

5. Weitergabe

5.4 Der Weitergabe-Experte

Mit dem *Weitergabe-Experten* („Simple Archiv Profile Wizard“) ist es möglich, zu jeder Anwendung ein passendes Archiv anzulegen, das alle Klassen enthält, die der Fremdnutzer der Applikation selbst nicht besitzt.

Wiederholen wir dafür noch einmal die Erzeugung des einfachen Beispiels, wie es sich unter Verwendung der Verkaufsversion des JDevelopers konkret darstellt. Dabei wird zuerst von den beiden möglichen beiden Varianten der *JDeveloper 2.0 (JDK 1.1.7)* gestartet.

Einschub: Einführungsbeispiel

Folgende Applikation ist zu entwickeln: Es soll eine Benutzeroberfläche erzeugt werden, deren Arbeitsfläche einen *gelben Hintergrund* hat. Auf dieser Arbeitsfläche soll sich eine *Schaltfläche* („button“) befinden, über die per Mausklick die *Beschriftung der Titelleiste* geändert werden kann.

Vorarbeit

Zuerst wird der *Oracle JDeveloper 2.0* gestartet.

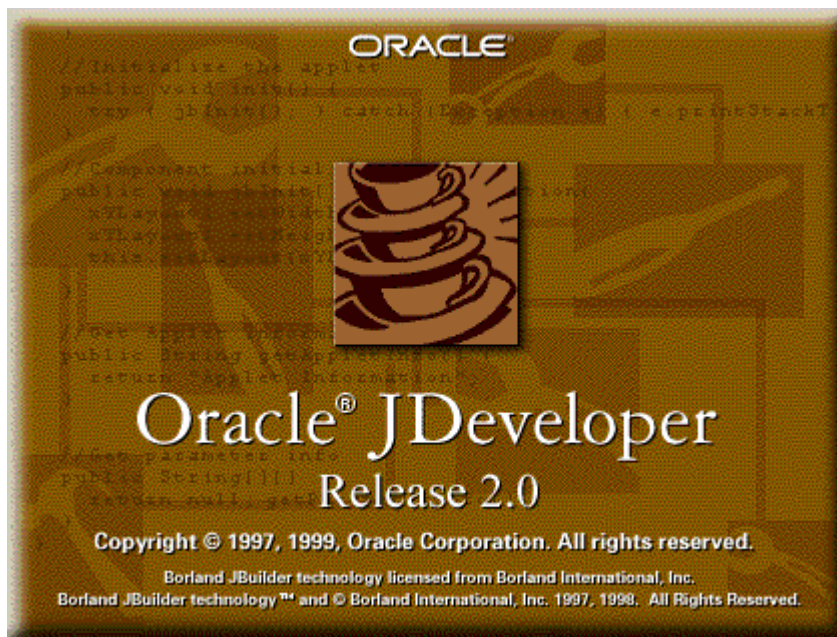


Abbildung E.1: Startbild des Oracle JDevelopers 2.0

Im Menü DATEI („File“) wählt man NEUES PROJEKT („New Project“), und im ersten Schritt des *Projekt-Experten* („Project Wizard“) wird zusätzlich festgelegt, daß eine *Applikation* hergestellt werden soll (siehe Abbildung E.2):

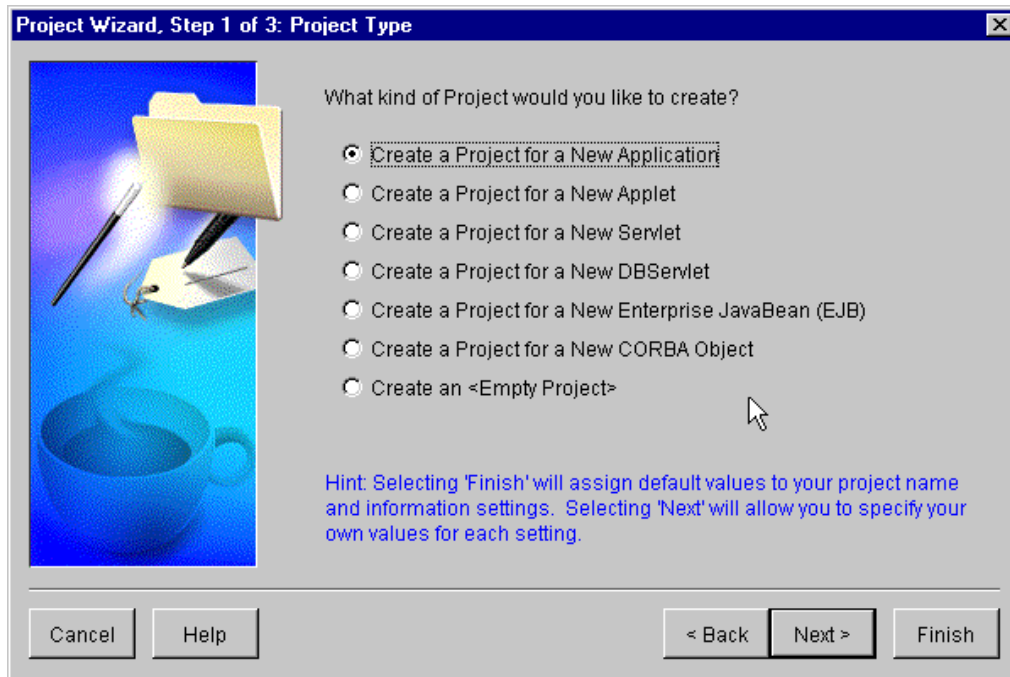


Abbildung E.2: Projekt-Experte, Schritt 1

In Schritt 2 des Projekt-Experten werden die grundsätzlichen Angaben zum Projekt erfragt:

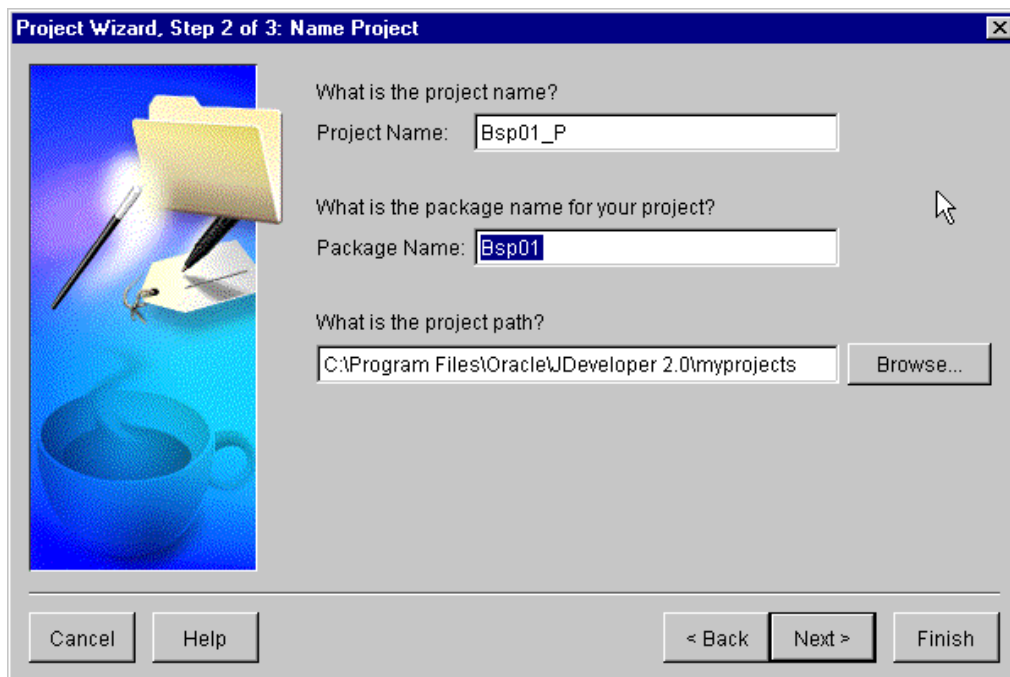


Abbildung E.3: Schritt 2 des Projekt-Experten

Der Schritt 3 des Projekt-Experten (Abb. E.4) schließlich erfragt weitere Angaben zum Projekt:

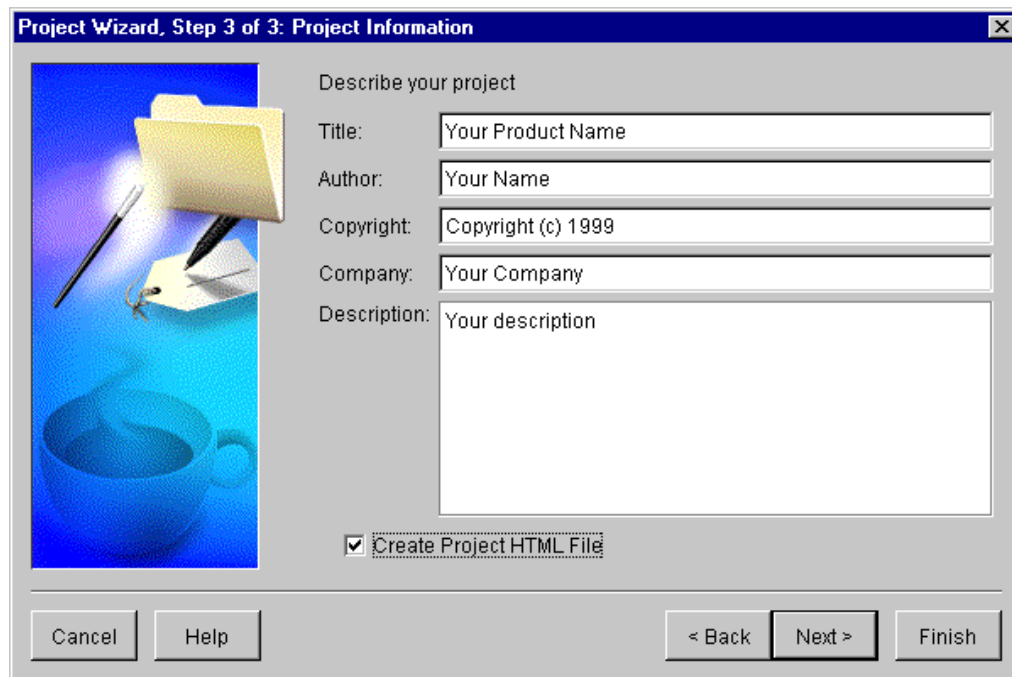


Abbildung E.4: Projekt-Experte, Schritt 3

Ist die Option *Erzeuge eine HTML-Datei zum Projekt* („Create Project HTML File“) ausgewählt, dann werden die gemachten Angaben später automatisch als Kommentarblock in den Kopf des generierten Programmes übernommen und sie bilden außerdem den Inhalt einer HTML-Datei zur *Projektdokumentation*.

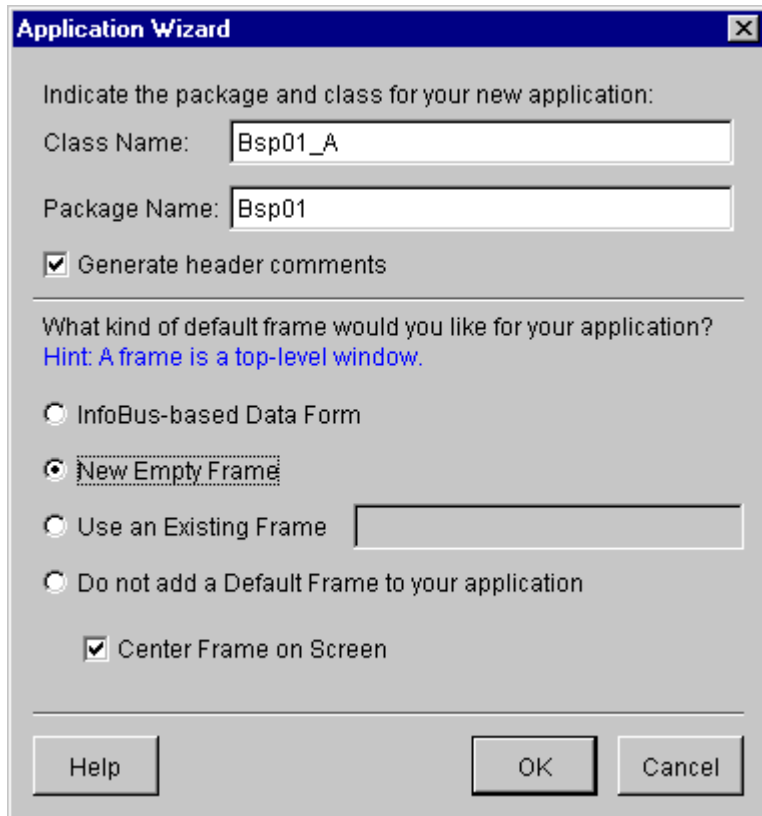


Abbildung E.5: Anwendungs-Assistent

Damit sind die drei Schritte des Projekt-Assistenten abgeschlossen.

Nun hilft der *Anwendungs-Assistent* („Application Wizard“) weiter, dessen Startbild in Abbildung E.5 zu sehen ist.

Er verlangt *einen Eintrag* und eine *wichtige Entscheidung*: Der *Klassen-Name* („Class Name“) definiert den Dateinamen der geplanten Applikation, und im unteren Teil des Eingabefensters muß entschieden werden, ob ein *Neuer, leerer Rahmen* („New Empty Frame“) – also eine neue Benutzeroberfläche – angefertigt werden soll, ob die Applikation in einen *Vorhandenen Rahmen* eingepaßt werden soll („Use an Existing Frame“) oder ob die Applikation *Keinen Standardrahmen* erhalten soll („Do not add a Default Frame to your application“).

Der Eintrag *Datenmaske basiert auf Rahmen* („InfoBus-based Data Form“) wird später in Kapitel benutzt, wenn mit Hilfe des *Datenmasken-Assistenten* („Data Form Wizard“) ein Programm zur Datenbankabfrage hergestellt werden soll. Damit ist der **zweite Teil der Vorarbeiten** beendet.

Im dritten und letzten Teil der Vorarbeiten sind für die beabsichtigte Benutzeroberfläche weitere Festlegungen zu treffen.

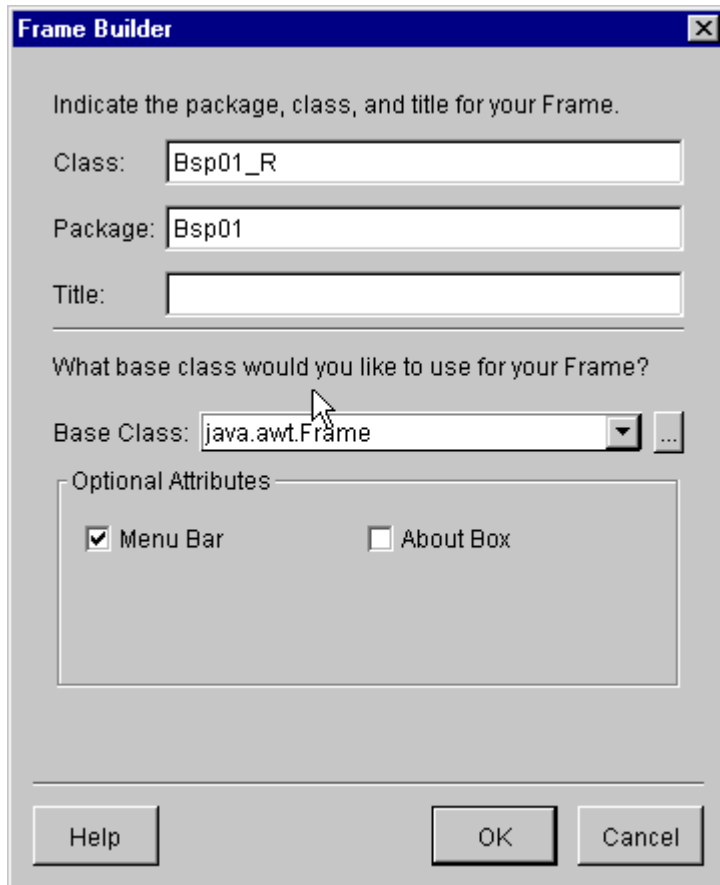


Abbildung E.6: Rahmen-Assistent für die Benutzeroberfläche

Abbildung E.6 zeigt den *Rahmen-Assistenten*. Hier wurde mit BSP01_R der Dateiname für die Quelltextdatei der Benutzeroberfläche festgelegt. Automatisch wird hier die *Menüleiste* („Menu Bar“) generiert. Auf eine Information über eine *About Box* wurde verzichtet, ebenso wurde die Titelzeile (vorerst) leer gelassen.

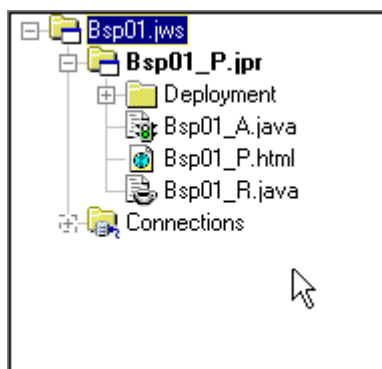


Abbildung E.7: Dateien des Projekts

Damit ist auch der **dritte Schritt der Vorbereitung** beendet; nachdem auch der Arbeitsbereich den Namen Bsp01 bekam und alle Dateien gesichert wurden, zeigt uns der *Navigator* (siehe Abbildung E.7) die bereits existierenden *vier Dateien*:

- BSP01_A.JAVA enthält das Hauptprogramm des Projekts.
- BSP01_R.JAVA enthält die Benutzeroberfläche.
- BSP01_P.JPR enthält die Organisationsangaben des Projekts.
- BSP01_P.HTML enthält die Dokumentation zum Projekt.

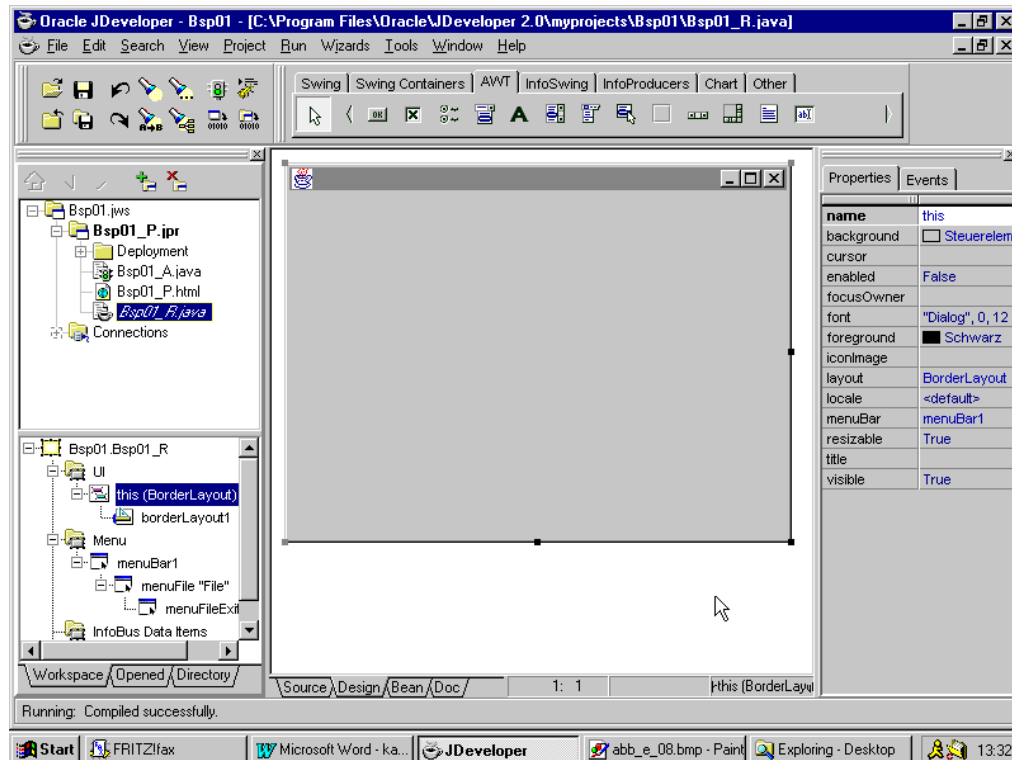


Abbildung E.8: Navigator und Komponentenbaum, Startformular, Inspektor

Visueller Entwurf der Benutzeroberfläche

Zum Entwurf der Benutzeroberfläche wird zuerst in dem *Navigator* des JDevelopers 2.0, der sich in Abbildung E.8 am linken Bildschirmrand in der Mitte befindet und stets eine *Übersicht über die am Projekt beteiligten Dateien* liefert, die zugehörige Datei BSP01_R.JAVA markiert. Im zentralen Fenster, dem *Projekt-Editor*, wird dazu das Register ENTWURF („Design“) gewählt.

Der Oracle JDeveloper 2.0 stellt standardmäßig ein *Startformular* (This) zur Verfügung, das nun in der Bildmitte angeordnet ist. Dieses Startformular This enthält in unserem Falle schon *eine Komponente* – die *Menüleiste* („Menu Bar“), die im Entwurf hier aber noch nicht zu erkennen ist.

Um eine große, im Moment noch völlig freie innere *Arbeitsfläche* (bevelPanel1) herzustellen, wird im Register AWT das Symbol *Panel* ausgewählt (Abb. E.9):

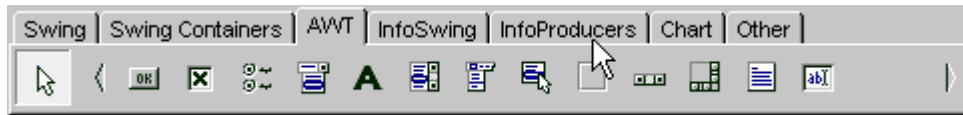


Abbildung E.9: Komponentpalette des Registers AWT

Zieht man dann auf der Benutzeroberfläche `This` das Panel auf, so liefert dieses Panel die große Arbeitsfläche unterhalb der Menüleiste, es wird in der Komponentpalette entsprechend eingetragen:

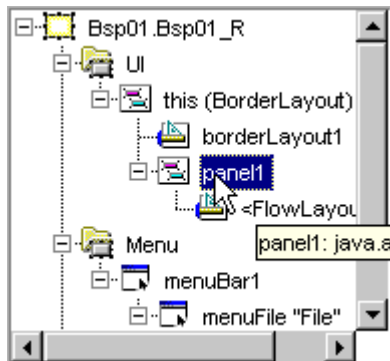


Abbildung E.10: Eingetragene Benutzeroberfläche in der Komponentpalette

Im *Komponentenbaum* („Component tree“) erkennt man nun die Hierarchien der vorhandenen Objekte. Um sie sich vorstellen zu können, denke man an aufeinandergelegte Folien: Zuunterst liegt die Folie `UI` der Benutzeroberfläche („UI = User Interface“), darüber die Folie `This` des Formulars, das hergestellt werden soll. Darauf liegt dann die Folie `panel1`. Sie beschreibt die große freie Fläche im Innern von `This`.

Die im Entwurf allgemein nicht sichtbaren Menüs sind in ihrer Hierarchie gesondert aufgeführt.

Bevor wir beginnen, auf der Arbeitsfläche gewisse Bedienelemente zu plazieren, sollte diese auf das dafür beste Layout umgestellt werden. Dazu wird entweder die Arbeitsfläche angewählt oder im Komponentenbaum `panel1` ausgewählt. Beides hat dieselbe Wirkung – rechts am Bildrand zeigt der *Inspektor* die Eigenschaften der Benutzeroberfläche `UI` an. In der Zeile *Layout* sollte am besten auf `XYLayout` umgestellt werden (siehe Abbildung E.11).



Abbildung E.11: Inspektor und Layout-Einstellung vom Panel

Nun wollen wir auf der Arbeitsfläche eine *Schaltfläche* („Button“) plazieren. Dafür wird zuerst in der Komponentenpalette AWT das Symbol „Button“ ausgewählt. Anschließend wird auf die Mitte der Arbeitsfläche geklickt und die Schaltfläche mit gedrückter Maustaste auf die gewünschte Größe gezoomt (siehe Abbildung E.12).

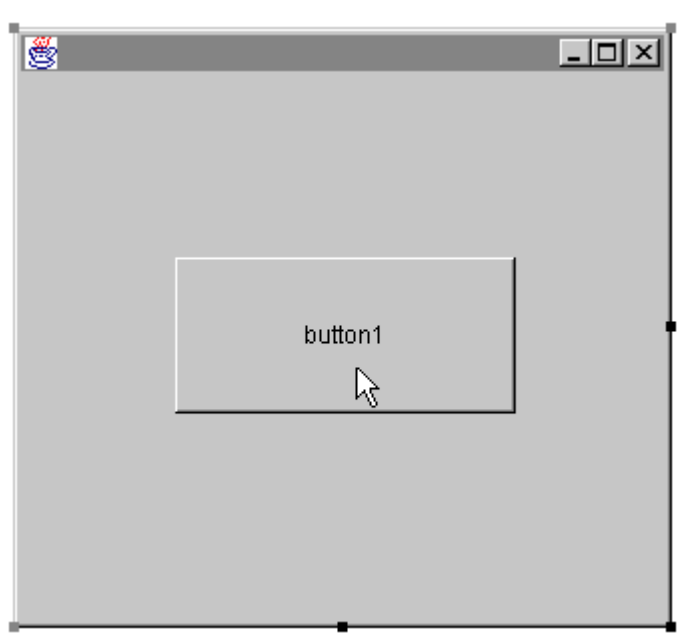


Abbildung E.12: Komponente `ButtonControl1` auf der Arbeitsfläche

Nun ist der erste Schritt beendet. Alle Dateien werden mit DATEI | ALLES SPEICHERN gesichert („File | Save All“).

Der erste Prototyp unserer geplanten Applikation ist bereits fertig.

Der Prototyp lässt sich bereits in dieser Stufe der Entwicklung *in seiner Funktionsfähigkeit prüfen*. Wählt man nämlich AUSFÜHREN („Run“) und AUSFÜHREN

„BSP01_A“ oder kürzer (Shift+F9) dann wird das Projekt, bestehend aus den beiden Java-Dateien für Hauptprogramm und Benutzeroberfläche, übersetzt und ausgeführt.

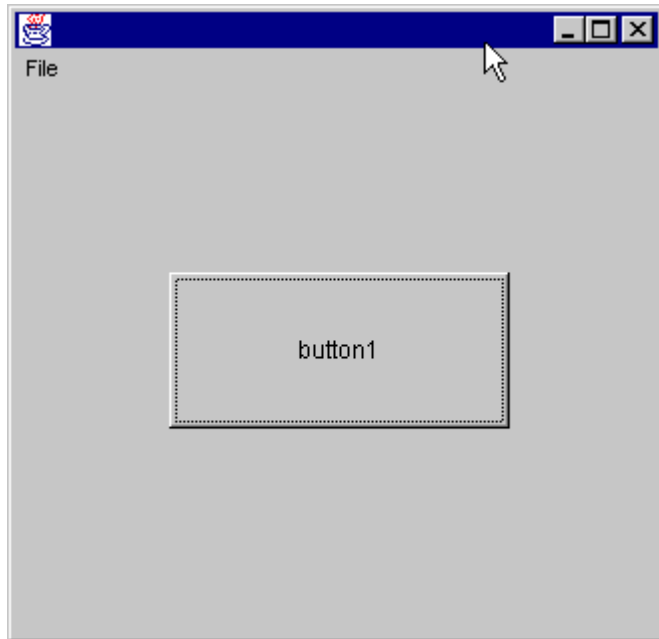


Abbildung E.13: Prototyp

Nun sieht man in Abbildung E.13 auch die automatisch generierte Menüleiste mit dem selbständig entstandenen Menüeintrag FILE (DATEI), man kann nun mit der Maus die einzelnen Objekte betätigen und sieht die Einträge in den Menüs.

Zuweisen der Eigenschaften

Um die Arbeitsfläche in unserer Applikation gelb einzufärben, wird sie mit der Maus angeklickt. Im Komponentenbaum links unten erscheint die zugehörige Bezeichnung `Panel1` – das ist der automatisch vergebene Name für diese Komponente. Rechts unten blendet sich der *Inspektor* auf. Mit ihm können die Eigenschaften der Komponente verändert werden.

Der ursprüngliche Eintrag bei *Hintergrundfarbe* („background“) – er lautete *Hellgrau* bzw. „Light gray“ – wird also geändert in *Gelb* („Yellow“).

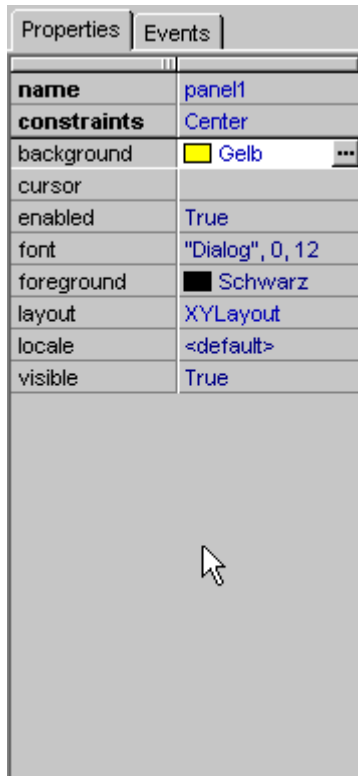


Abbildung E.14: Inspektor, Register EIGENSCHAFTEN

Damit wurde beispielhaft vorgeführt, wie Eigenschaften von Komponenten verändert werden können. Wiederum kann nun der veränderte Prototyp erprobt werden – wie gewünscht leuchtet das Arbeitsfenster gelb und der Button, da nicht verändert, bleibt bei seiner voreingestellten hellgrauen Farbe.

```

this.add(panel1, BorderLayout.CENTER);
panel1.add(button1, new XYConstraints(77, 92, 170, 78));
this.setMenuBar(menuBar1);
}

void fileExit_actionPerformed(ActionEvent e) {
    System.exit(0);
}

void button1_actionPerformed(ActionEvent e) {
    |
}
}

```

Abbildung E.15: Generierter Quellcode mit Einfügeposition

Ereignisbehandlungen schreiben

Ein Doppelklick auf die Schaltfläche verändert den Inhalt des Editierfensters: Dort, wo eben noch wegen des eingestellten Modus ENTWURF („Design“) der Entwurf der Benutzeroberfläche zu sehen war, erscheint nun ein Quelltext-Stück. Die Cursor-Position markiert dabei die Stelle, an der *eigener Quelltext* einzufügen ist.

Wir benötigen nur eine einzige Zeile, um unsere beabsichtigte Wirkung beim Mausklick auf den Button, nämlich die Änderung der Überschrift, zu erzielen:

```
this.setTitle("Hallo, liebe Java- und Oracle-Freunde");
```

Sind diese Zeile eingetragen und wiederum alle Dateien gesichert, so empfiehlt sich noch einmal ein einfacher Klick auf die Schaltfläche und ein Blick auf den *Inspektor*, diesmal auf das Register EREIGNISSE („Events“).

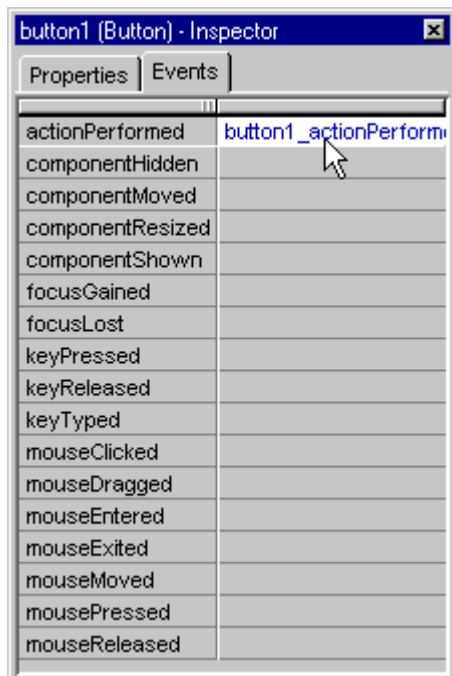


Abbildung E.16: Inspektor, Register EREIGNISSE

Wie zu sehen, wurde also für das Ereignis actionPerformed (dazu gehört natürlich auch der Mausklick) ein entsprechender Bezug eingetragen.

Nun ist unsere Aufgabe bereits gelöst. Wir sichern wiederum alle Dateien.

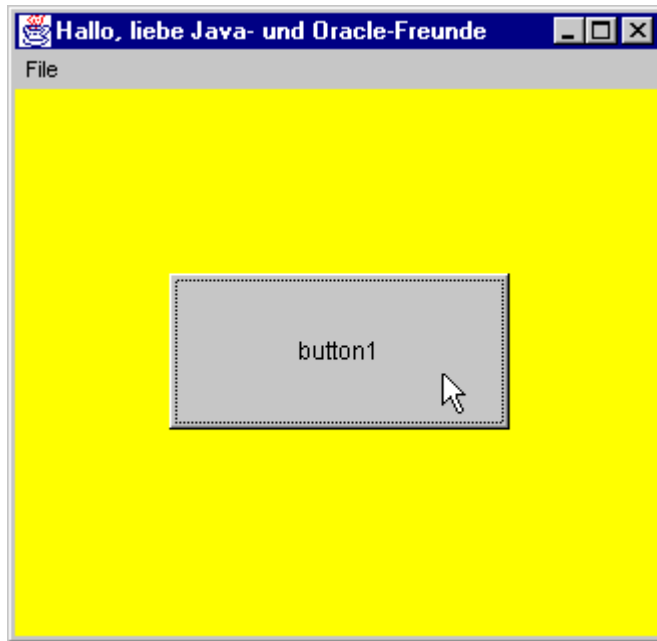


Abbildung E.17: Benutzeroberfläche mit gewünschter Arbeitsweise

Dann lassen wir das Projekt ausführen – und wir erhalten, wie in Abbildung E.17 zu erkennen, in der vorher leeren Titelzeile nach dem Mausklick auf die Schaltfläche („Button“) die programmierte Überschrift.

Ende von: Einschub - Einführungsbeispiel

Gehen wir gleichermaßen noch einmal davon aus, daß wir den Fremdnutzer dadurch simulieren, daß unsere Quelltextdateien in einem Verzeichnis D:\MYPROJECTS\BSP01 und unsere Nutzerklassen in einem Verzeichnis D:\MYCLASSES\BSP01 vorhanden seien.

Zur erfolgreichen Simulation eines Fremdnutzers müssen wir zuerst in das Verzeichnis D:\MYCLASSES wechseln, dann den CLASSPATH auf dieses Verzeichnis richten und schließlich die Ausführung versuchen:

```

MS-DOS
D:\myclasses>set CLASSPATH=D:\myclasses
D:\myclasses>java Bsp01.Bsp01_A
java.lang.NoClassDefFoundError: oracle/jdeveloper/layout/XYLayout
    at Bsp01.Bsp01_R.<init><Bsp01_R.java:13>
    at Bsp01.Bsp01_A.<init><Bsp01_A.java:17>
    at Bsp01.Bsp01_A.main<Bsp01_A.java:33>
D:\myclasses>

```

Abbildung 5.30: Versuch der Ausführung durch simulierten Fremdnutzer

Wir wollen nun mit Hilfe des Weitergabe-Experten ein Archiv anlegen, es soll den Namen BSP01.JAR tragen und im Verzeichnis D:\MYCLASSES angelegt sein. Dazu lö-

schen wir erst einmal alle bisher vorhandene Klassen-Dateien in diesem Verzeichnis. *Sie werden überhaupt nicht mehr benötigt!*

Nun benutzen wir wieder den JDeveloper 2.0 und öffnen die Projektdatei BSP01.PRJ. Die Übersicht (vgl. Abbildung 5.31) sowie eine Kontrollausführung ergeben, daß alle notwendigen Dateien vorhanden sind.



Abbildung 5.31: Dateien des Projekts

Nun wird gemäß Abbildung 5.32 im Menü PROJECT die Eigenschaft WEITERGABE („Deploy“) ausgewählt.

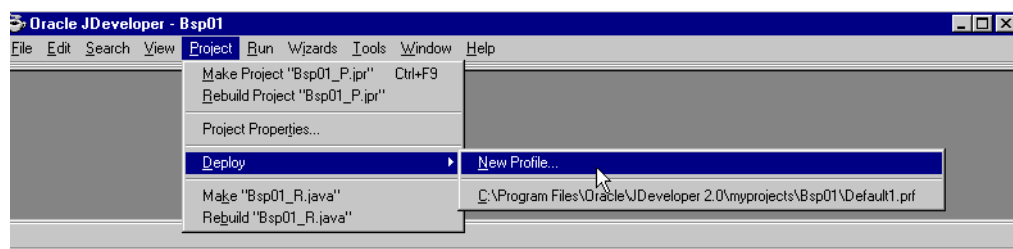


Abbildung 5.32: Start des Weitergabe-Experten

Das weitere Vorgehen ist recht einfach. Im ersten Schritt fragt der Weitergabe-Experte nach dem „Weitergabeziel“, bzw. der Bestimmung der Applikation – angeboten wird als erste Option ein einfaches Archiv („simple archive file“) (siehe Abbildung 5.33). Dieses soll auch hier ausgewählt werden.

Der zweite, dritte und vierte Schritt des Weitergabe-Experten kann anfangs schnell übergangen werden, indem jeweils die angebotene Voreinstellung bestätigt wird.

Dabei soll bereits jetzt darauf hingewiesen werden, daß der vierte Schritt beachtliche Möglichkeiten bietet, die Größe des zu erzeugenden Archivs zu beeinflussen. Doch vorerst wollen wir die vorgeschlagenen Werte annehmen und uns dem fünften und letzten Schritt des Weitergabe-Experten zuwenden (siehe Abbildung 5.34).

In diesem Schritt muß mitgeteilt werden, in welches Verzeichnis das Archiv kommen und welchen Namen es dort erhalten soll. Dabei ist zu beachten, daß eine Archivdatei mit der Extension .JAR gleichermaßen Dateisystem wie enthaltene Dateien speichert.

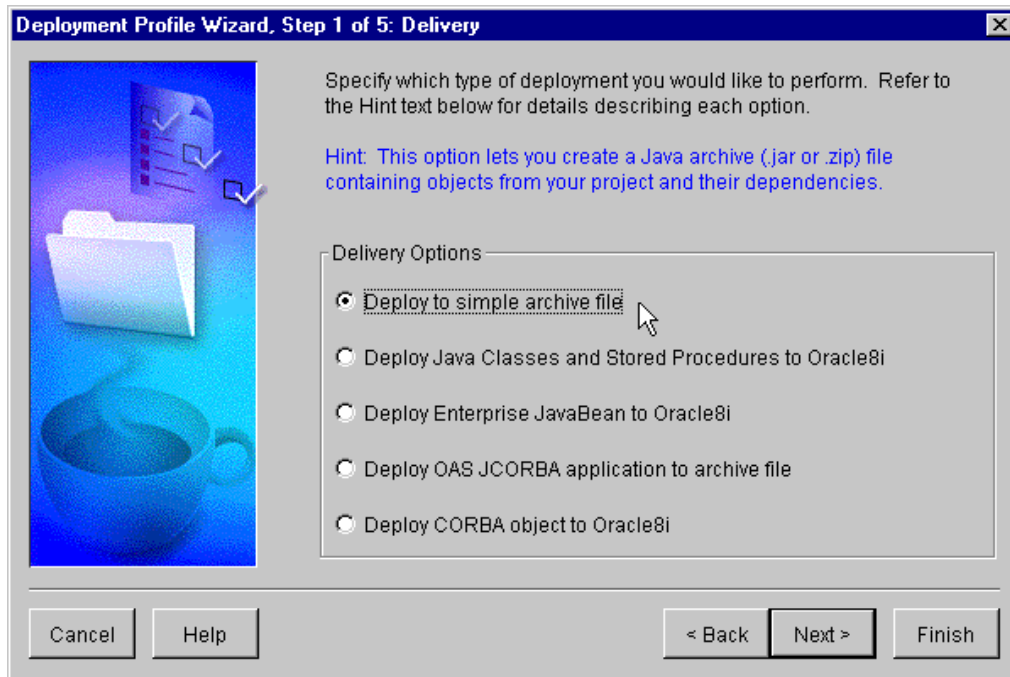


Abbildung 5.33: Weitergabe-Experte, Schritt 1

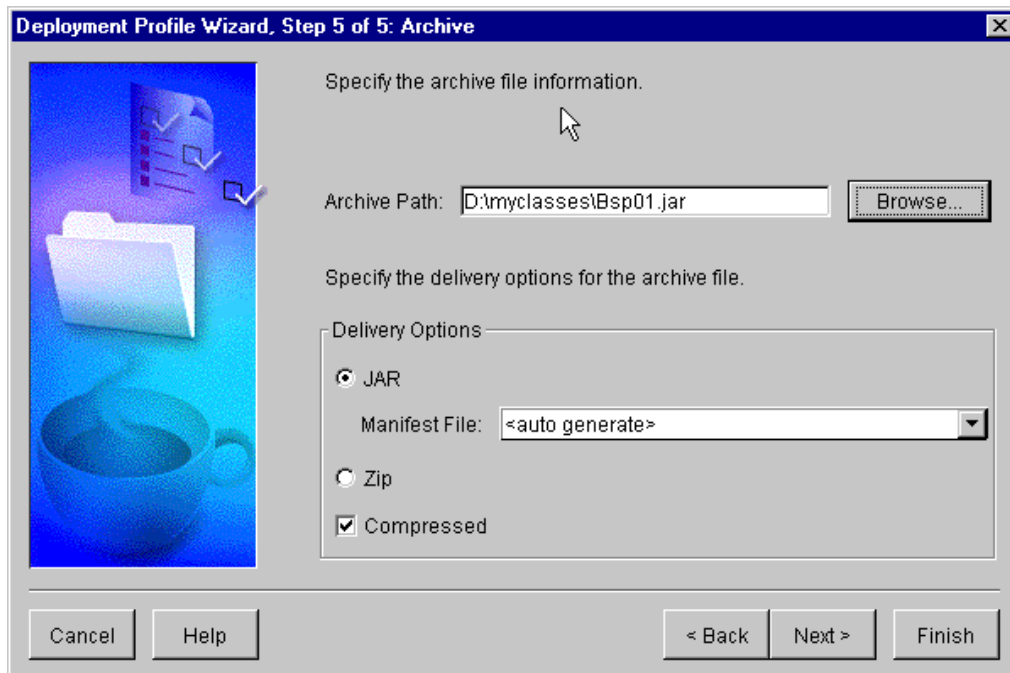


Abbildung 5.34: Weitergabe-Experte, fünfter und letzter Schritt

Nun kann schon mit Beenden („Finish“) die Herstellung des Archivs veranlaßt werden, der Nutzer wird dabei über den Verlauf der Arbeit informiert (Abb. 5.35).

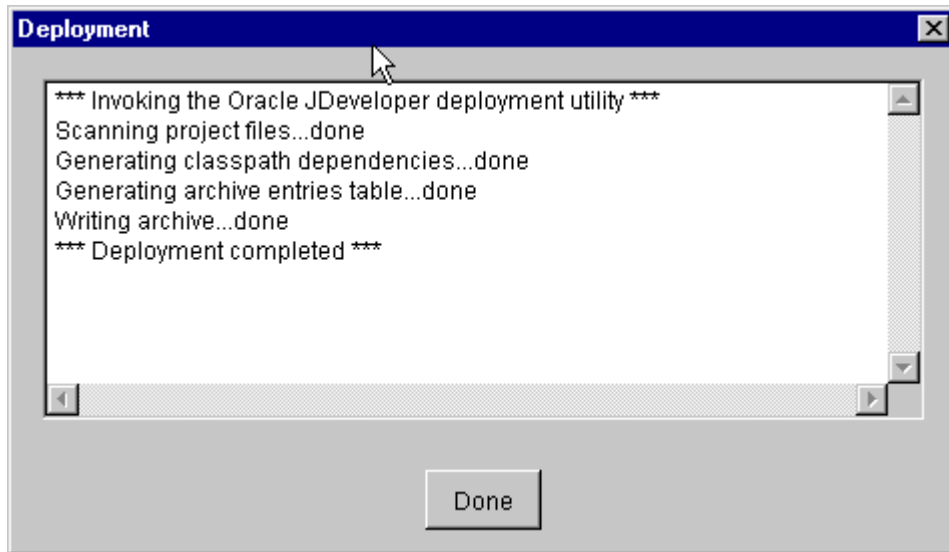


Abbildung 5.35: Erfolgsmeldung des Weitergabe-Experten

Im MS-DOS-Modus ist zu erkennen, daß die Archivdatei in dem gewünschten Verzeichnis angelegt worden ist (Abb. 5.36).



Abbildung 5.36: Archivdatei im angegebenen Zielverzeichnis

Es sei dabei darauf hingewiesen, daß tatsächlich nur diese eine Archivdatei zur Weitergabe benötigt wird, keine weitere Klassendatei, nichts weiter! Bevor nun aber kontrolliert werden kann, ob ein Fremdnutzer tatsächlich mit dieser Archivdatei alle nötigen Klassen zur Ausführung des Projektes erhalten hat, muß der CLASSPATH auf diese Datei gerichtet werden (Abb. 5.37).

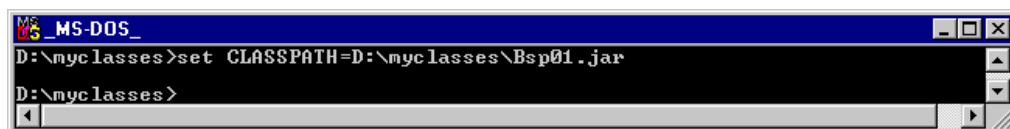


Abbildung 5.37: Klassenpfad auf die Archivdatei

Es klingt unlogisch, einen Pfad auf eine Datei zu richten, aber diese Datei enthält immerhin neben den benötigten Klassen auch das komplette Dateisystem, in dem diese Klassen abgespeichert sind!

Nun müssen wir uns erinnern:

Dem Paket („package“) hatte wir den Namen BSP01 gegeben (vgl. Abb. E.5)

Die Applikation hatte den Namen BSP01_A erhalten.

Diese beiden Angaben sind ausreichend: Die Anwendung wird gestartet, indem die Datei JAVA.EXE in einer DOS-Kommandozeile mit Paketnamen und Applikationsnamen, durch Punkt getrennt, aufgeführt wird:

```
Java Bsp01.Bsp01_A
```

Nun wird die weitergegebene Applikation vom Betriebssystem (hier DOS) aus gestartet (vgl. Abb 5.38):

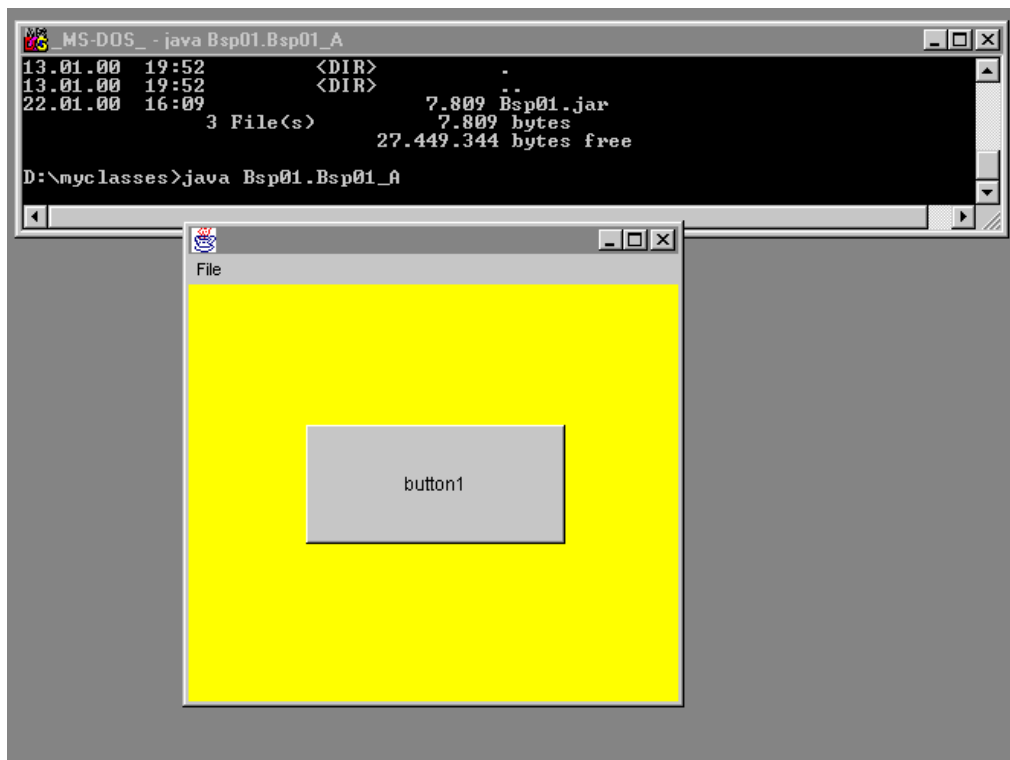


Abbildung 5.38: Start der weitergegebenen Applikation

Fassen wir noch einmal die (wenigen) Arbeitsschritte von der fertigen Applikation bis zur weitergabefähigen Archivdatei zusammen:

1. Herstellung der Applikation mit dem Oracle JDeveloper
2. Festlegung des Paketnamens („package“) und des Namens der Applikation
3. Anlegen eines Verzeichnisses um das Ablegen der Archivdatei zu testen
4. Erzeugen der Archivdatei mit dem Weitergabe-Experten, Ablage in dem Testverzeichnis
5. Wechsel in den MS-DOS-Modus
6. Setzen des CLASSPATH auf die Archivdatei (vollst. Pfad+Dateiname!)

7. Testendes Ausführen im MS-DOS-Modus, wobei Paket- und Applikationsnamen mit Punkt getrennt angegeben werden müssen

Abschließend nun einige Bemerkungen zur Größe der Archivdatei. Wenn das Archiv ohne eigene Entscheidungen (außer der Namensvergabe) gemäß dem *Standard des Weitergabe-Experten* hergestellt wird, dann kann es schon bei einfachen Applikationen relativ umfangreich werden. Der Grund dafür ist leicht zu finden: Um der Gefahr zu entgehen, daß beim Fremdnutzer doch womöglich eine wichtige Klasse fehlt, wird „des Guten zuviel getan“, sprich: Es werden vorsichtshalber alle denkbaren Klassen in die Archivdatei übernommen.

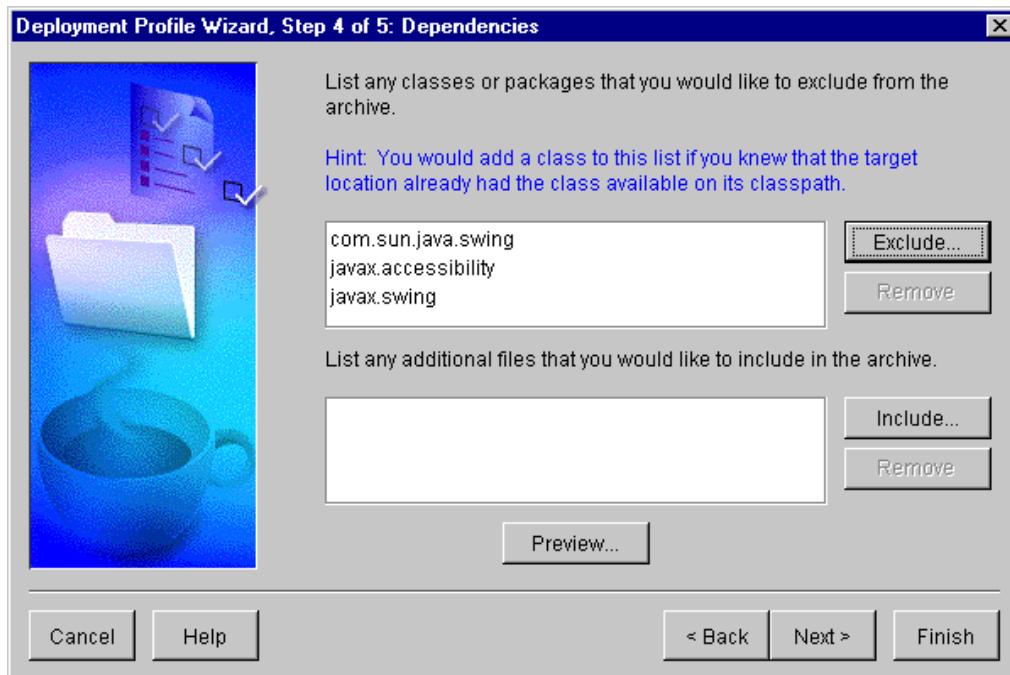


Abbildung 5.39: Weitergabe-Experte, vierter Schritt

Im vierten Schritt des Weitergabe-Experten, der oben durch Bestätigung der Voreinstellung übergangen wurde, finden sich jedoch *Möglichkeiten, die Größe des Archivs zu beeinflussen*.

Was ist zu tun? Überlegen wir: Das Archiv zu unserer Applikation soll *so klein wie möglich* werden. Wie klein es tatsächlich werden kann – das hängt aber von den Klassen ab, über die der Fremdnutzer, an den wir die Applikation weitergeben wollen, *bereits verfügt*. Und es hängt ebenso von den Klassen ab, die für unsere Applikation gebraucht werden. Deshalb bietet der Weitergabe-Experte im oberen Fenster eine *Liste von Klassenbibliotheken* an, die wahlweise **nicht berücksichtigt bzw. ausgeschlossen** werden können.

Wenn eine Klasse

- mit Sicherheit für die Applikation nicht gebraucht wird oder
- beim zukünftigen Nutzer bestimmt vorhanden ist,

dann wird die zugehörige Klassenbibliothek *nicht* aus der Liste entfernt!

Nehmen wir an, der Fremdnutzer besitze *alle Klassen*, die die Applikation benötigt. Folglich entfernen wir *nichts*. In diesem Falle braucht der Fremdnutzer nur die Projekt-klassen, alles andere hat er ja selbst. So lautete unsere Annahme.

Nehmen wir nun dagegen an, der Fremdnutzer habe *nichts*. Dann gibt es zwei mögliche Vorgehensweisen:

Wir streichen *alle* Klassenbibliotheken. Dann werden *ausnahmslos alle Bibliotheken* zur Archivbildung herangezogen, auch diejenigen, deren Klassen überhaupt nichts mit unserer Applikation zu tun haben. Das Archiv kann groß werden, sogar sehr groß – aber der Weg ist sicher.

Verfügen wir dagegen über weitere Informationen, dann sollten wir gezielt streichen.