

Benutzungs- oberflächen

Bonusaufgabe

Prof. Dr.-Ing. Holger Vogelsang

Sommersemester 2016

**Don't
panic**

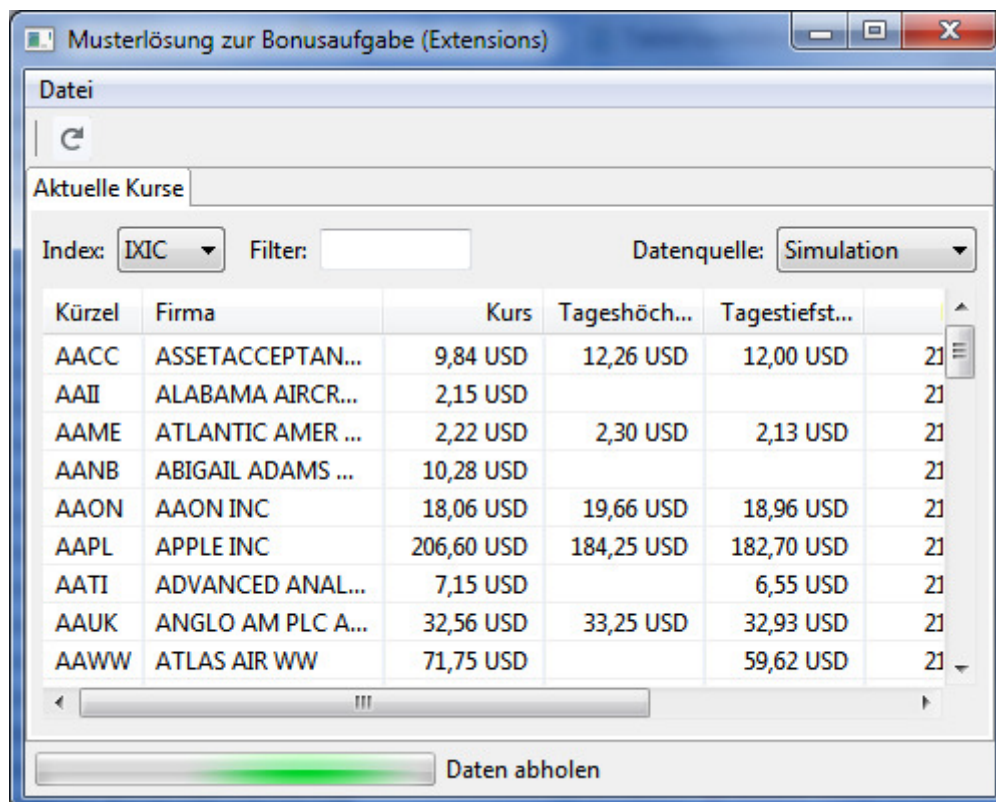
Inhaltsverzeichnis

1	Einleitung	4
2	OSGi-Service (optional)	5
2.1	Vorbereitungen	5
2.2	Plug-in zum Auslesen der Daten	8
2.3	Plug-in zur testweisen Ausgabe der Daten	10
3	RCP-Anwendung	11
3.1	Vorbereitungen	11
3.2	Layout	13
3.3	Ereignisbehandlung	14
3.4	Tabelle	15
3.5	Anbindung der Datenquellen	17
3.6	Internationalisierung	18
3.7	Multithreading	18
3.8	Sperren von Einträgen (optional)	19
4	Plug-in für die RCP-Anwendung (optional)	21

Einleitung

Die Bearbeitung des Übungsblattes für das Wahlfach ist freiwillig, wobei Sie allerdings für die Abgabe der kompletten und korrekten Lösung vor der Klausur einen Bonus von zehn Punkte in der Klausur erhalten. Die optionalen Aufgabenteile müssen nicht bearbeitet werden, sind aber eine gute Klausurvorbereitung. Eine kleine Einführung in das Thema Multithreading mit Java finden Sie im Skriptfragment.

Die Lösungen dieser Aufgaben ergeben einen sehr einfachen Rich-Fat-Client, der in der Lage ist, Börsenkurse abzuholen und tabellarisch darzustellen. Die Teilaufgaben führen schrittweise zur Lösung hin. Die folgende Abbildung zeigt die Musterlösung.



Kürzel	Firma	Kurs	Tageshöch...	Tagestiefst...	
AACC	ASSETACCEPTAN...	9,84 USD	12,26 USD	12,00 USD	21
AAII	ALABAMA AIRCR...	2,15 USD			21
AAME	ATLANTIC AMER ...	2,22 USD	2,30 USD	2,13 USD	21
AANB	ABIGAIL ADAMS ...	10,28 USD			21
AAON	AAON INC	18,06 USD	19,66 USD	18,96 USD	21
AAPL	APPLE INC	206,60 USD	184,25 USD	182,70 USD	21
AATI	ADVANCED ANAL...	7,15 USD		6,55 USD	21
AAUK	ANGLO AM PLC A...	32,56 USD	33,25 USD	32,93 USD	21
AAWW	ATLAS AIR WW	71,75 USD		59,62 USD	21

Abbildung 1.1: Darstellung der Börsenkurse in in der Musterlösung

OSGi-Service (optional)

Die Börsenkurse werden vom Yahoo-Server <http://quote.yahoo.com/> als einfache Liste abgeholt. Die Einträge der Liste sind durch Kommata getrennt, wobei die Auswahl der gewünschten Daten über Requestparameter erfolgt. Damit Sie sich mit diesen Details nicht beschäftigen müssen, übernimmt das bereitgestellte Plug-in `StockQuotesYahoo` das Abholen der Kurse. Da im Rahmen des Projektes auch Kursänderungen dargestellt werden sollen und die Börse nicht immer geöffnet ist, existiert ein Simulationsmodus innerhalb des Plug-ins `StockQuotesSimulation`. Das dritte vorhandene Plug-in `StockQuotes` stellt gemeinsame Schnittstellen zur Verfügung.

In dieser Teilaufgabe beschäftigen Sie sich zunächst nur mit OSGi und verwenden die bereitgestellten Plug-ins.

2.1 Vorbereitungen

Laden Sie aus dem Ilias die vorgegebenen Plug-ins herunter, damit Sie nicht selbst die Kommunikation und Simulation erstellen müssen. Sie finden die Plug-ins im Ilias unter „IW 332 Benutzungsoberflächen“ → „Bonusaufgabe“ → „Benutzungsoberflächen-Plug-ins.zip“. Den Inhalt dieser ZIP-Datei fügen Sie bitte Ihrer Target-Plattform am besten in einem separaten Ordner hinzu. Damit Sie sich die Plug-ins auch im Quelltext ansehen können, ist dieser jeweils in Form separater Plug-ins in der ZIP-Datei vorhanden. Noch einfacher ist es, wenn Sie die Plug-ins über diese „Update Site“ in Ihre „Target Platform“ aufnehmen:

<http://www.iwi.hs-karlsruhe.de/Intranetaccess/public/stockquotes/updates> (Achtung: Zeilenumbruch entfernen)

2.1.1 Schritt 1

Erstellen Sie jetzt in Eclipse ein neues Plug-in-Projekt: „New“ → „Project...“ → „Plug-in Development“ → „Plug-in Project“. Geben Sie Ihrem Projekt einen sinnvollen Namen. Wichtig ist, dass Sie als „Target Platform“ den Haken vor „an OSGi framework“ setzen.

2.1 Vorbereitungen

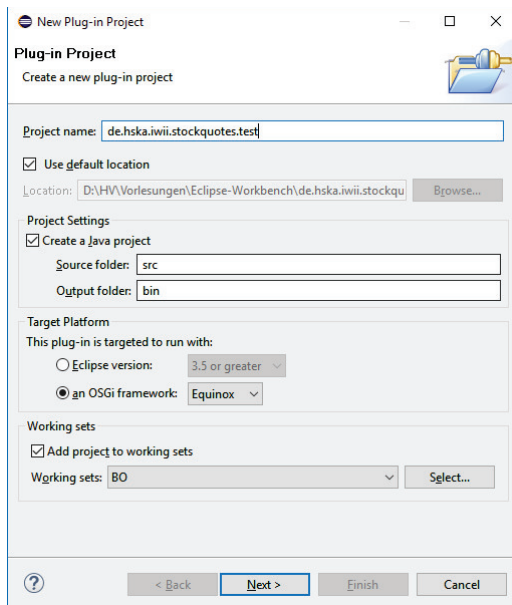


Abbildung 2.1: OSGi-Projekt (Schritt 1)

2.1.2 Schritt 2

In diesem Schritt geben Sie Ihrem Plug-in eine Version und achten darauf, dass der „Activator“ erzeugt wird. Klassennamen und Paket können Sie frei wählen, wobei für einfache Projekte die Voreinstellung sicherlich nicht schlecht ist.

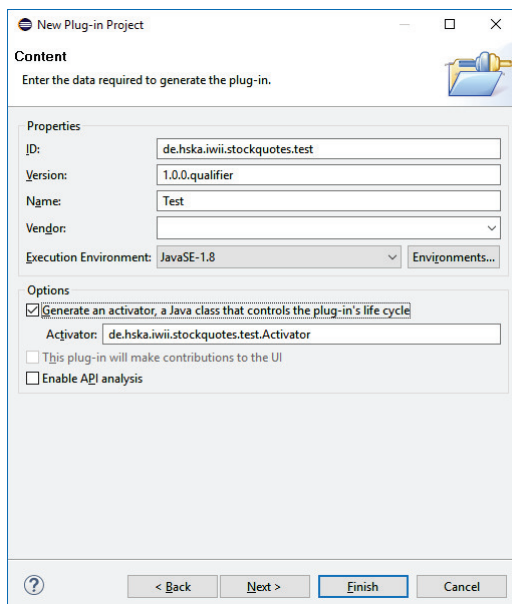


Abbildung 2.2: OSGi-Projekt (Schritt 2)

2.1.3 Schritt 3

Damit Sie „etwas sehen können“, bietet es sich an, das Plug-in auf Basis eines Beispiels erzeugen zu lassen. Sinnvoll ist im Falle des synchronen Aufrufs das „Hello OSGi Bundle“, im Falle des asynchronen Aufrufs das „OSGi EventAdmin Service Example“.

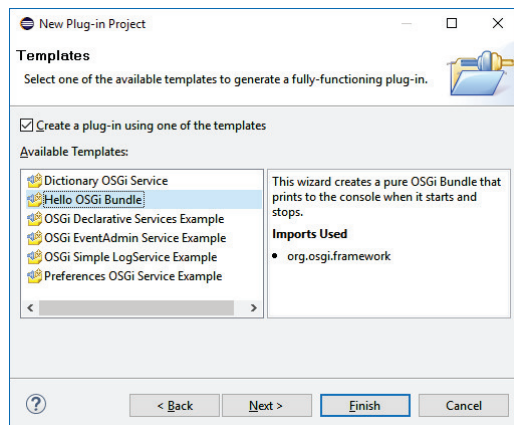


Abbildung 2.3: OSGi-Projekt (Schritt 3)

Im letzten Dialog können noch Ausgabetexte für das Beispiel gewählt werden. Diese sind nicht wichtig, weil sie später ohne wieder entfernen werden.

2.1.4 Schritt 4 – Start des Plug-ins

Jetzt sollte sich das Plug-in starten lassen. Leider erzeugt Eclipse hier häufig nicht sonderlich korrekte Start-Konfigurationen. Wenn Sie also in der Konsole eine Meldung sehen, nach der bestimmte Abhängigkeiten nicht aufgelöst werden können, dann müssen Sie die Start-Konfiguration reparieren:

- Nach einem Rechtsklick mit der Maus auf das Projekt wählen Sie „Run As“ → „Run Configurations...“ aus.
- Erstellen Sie eine neue Konfiguration für den Typ „OSGi Framework“.
- Wählen Sie dann im Tab „Bundles“ nur die folgenden Plug-ins aus:
 - eigenes Plug-in des Projektes
 - `org.apache.felix.gogo.runtime`
 - `org.apache.felix.gogo.shell`
 - `org.eclipse.equinox.console`
 - `org.eclipse.osgi`
 - `org.eclipse.osgi.compatibility.state`
 - `org.eclipse.equinox.ds`
 - `org.eclipse.equinox.common`
 - `org.eclipse.equinox.event`
 - `org.eclipse.equinox.util`
 - `org.eclipse.osgi.services`

Wenn Sie deutschsprachige Meldungen sehen wollen, dann können Sie für die sechs zuletzt genannten Plug-ins zusätzlich die entsprechenden auch die Sprach-Bundles auswählen (selber Name mit angehängtem `.nl_de`). Sie sollten das Bundle `org.eclipse.equinox.event` sofort starten. Dazu tragen Sie in der „Run-Configuration“ als „Auto Start“ den Wert „true“ ein und nehmen als „Start Level“ den Wert „3“. So steht der intern in den Börsenkurs-Plugins verwendete Event-Admin-Dienst den Plugins zur Verfügung.

- In den Pool-Räumen kann es vorkommen, dass OSGi versucht, in ein Standardverzeichnis zu schreiben, für das Sie keine Schreibrechte besitzen. Sie erkennen das daran, dass sich Ihr Programm mit einer Ausnahme meldet. Um OSGi ein anderes Ausgabeverzeichnis vorzugeben, können Sie `-data <verzeichnis>` als Programmargument in der Startkonfiguration angeben, wobei Sie für `<verzeichnis>` Schreibrechte besitzen müssen. Eine gute Wahl ist ein Verzeichnis unterhalb von `n:\.`

Jetzt lässt sich das Projekt korrekt starten. Wenn Sie sich den „Activator“ ansehen, dann erkennen Sie auch, dass er beim Start des Plug-ins die Meldung ausgibt.

Die Konsole dient nicht nur der Ausgabe der Meldungen, sie ist eine komplette OSGi-Konsole. Mit `stop de.hska.iwii.stockquotes.test` (oder wie immer Ihr Plug-in auch heißt) wird das Plug-in gestoppt. `help` führt Ihnen alle Befehle der Konsole auf.

2.2 Plug-in zum Auslesen der Daten

Erweitern Sie Sie nun das Plug-in aus der Teilaufgabe 2.1, so dass Sie die bereitgestellten Plug-ins verwenden.

2.2.1 Service-Tracker

In diesem Schritt erweitern Sie den „Activator“, damit er die Aktivierung und Deaktivierung der Simulations- und Yahoo-Bundles registriert.

- Machen Sie Ihr Plug-in von den bereitgestellten abhängig.
- Erstellen Sie einen eigenen Service-Tracker, der auf Plug-ins reagiert, die den Dienst `IStockQuotes` bereitstellen. Diese Schnittstelle wird von dem Simulations- und dem Yahoo-Plug-in implementiert. So werden Sie informiert, wenn sich die Yahoo- und Simulations-Plug-ins aktivieren bzw. deaktivieren. Den Service-Tracker aktivieren Sie am besten aus Ihrer Activator-Klasse heraus.
- Wenn Sie das Plug-in jetzt starten, werden Sie unter Umständen viele Fehlermeldungen sehen. Öffnen Sie die Start-Konfiguration und achten Sie darauf, dass Sie die Plug-ins

```
de.hska.iwii.stockquotes.net
de.hska.iwii.stockquotes.net.simulation
de.hska.iwii.stockquotes.net.yahoo
```

ebenfalls starten. Damit Sie nicht alle weiteren Abhängigkeiten manuell herstellen müssen, können Sie mit „Validate Bundles“ prüfen, welche fehlen oder gleich mit „Add Required Bundles“ die fehlenden Abhängigkeiten ergänzen lassen. Anschließend sollte sich das Projekt starten lassen. Wenn Sie auf der OSGi-Konsole das Simulations- oder Yahoo-Bundle mit `stop` bzw. `start` anhalten bzw. wieder starten, dann sehen Sie, dass der Service-Tracker darauf reagiert.

Sollten Sie statt eines Service-Trackers lieber den OSGi-eigenen Dependency-Injection-Mechanismus verwenden wollen, so spricht nichts dagegen. Möchten Sie auch zulassen, dass sich die Yahoo- und Simulations-Bundles stoppen und erneut starten lassen, dann müssen Sie als „policy“ in der Datei `component.xml` den Wert `dynamic` angeben: `policy="dynamic"`. Der Wert `static` bewirkt nämlich, dass Ihr eigenes Plug-in beim Stoppen der Yahoo- und Simulations-Plug-ins auch gestoppt wird und beim Starten erneut alle Objekte Ihres Plug-ins erzeugt werden. Alle Daten sind danach verloren.

2.2.2 Zugriff auf Simulations- und Yahoo-Daten

Erstellen Sie eine weitere Klasse im Projekt, die mit den bereitgestellten Plug-ins kommuniziert. Diese Klasse stellt Ihre Schnittstelle zur Verwendung der beiden Plug-ins (Simulation und Echtzeitdaten) dar. Die Klasse muss folgendes können:

- zwischen beiden Datenquellen auswählen
- Daten aus einer Quelle auslesen
- Börsenindex aus einer Quelle auswählen
- die Namen aller Quellen zurückgeben

Dazu stellen Ihnen die beiden Datenquellen eine kleine API zur Verfügung, die durch das Interface `IStockQuotes` repräsentiert wird. Die Implementierungsdetails der Klassen sind zur Verwendung unwichtig. Schnittstelle `IStockQuotes`:

1. Erfolgt die Verbindung zum Server über einen Proxy (z.B. innerhalb der Hochschule), dann müssen zunächst die Proxy-Daten eingetragen werden.

- `void setProxy(String inetAddr, int port):`
Die Methode benötigt den Domainname oder die IP-Adresse des Proxy-Servers sowie dessen Portnummer. Innerhalb der Hochschule lauten die Angaben für den Domainnamen `proxy.hs-karlsruhe.de` sowie `8888` für den Port.
- `void setProxyAuthentication(String user, String password):`
Der Aufruf speichert sich intern den Nutzernamen sowie das Passwort für die Proxy-Anmeldung.

Die Proxy-Einstellungen werden im Simulationsbetrieb ignoriert.

2. Im nächsten Schritt werden die Daten vom Server abgeholt bzw. durch die Simulation aus statischen Daten ermittelt. Dazu implementieren die Bundles eine asynchrone und eine synchrone Variante zum Abholen.

- Synchrones Abholen bedeutet, dass die Methode als Ergebnis die neuen Daten zurückgibt. Nachteilig kann sein, dass der Aufruf wegen einer Netzwerkverbindung unter Umständen langsam ist. Signatur:

```
List<StockData> requestDataSync(String stockIndex)
```

- Der Parameter `stockIndex` ist der Name des Börsenindexes, von dem die Daten abgeholt werden sollen. Damit Sie nicht manuell basteln müssen, sind die drei Konstanten `DOW_JONES`, `NASDAQ` und `FTSE100` in der Schnittstelle `IStockQuotes` vordefiniert. `ALL_STOCK_INDEXES` beinhaltet ein Array der Namen aller drei Börsenindizes.
- Als Rückgabewert erhalten Sie eine `List` mit den Kursdaten. Die Klasse `StockData` enthält neben den aktuellen Kursen auch die jeweils vorherigen Werte, damit Sie später in der Tabelle die Art der Kursänderung anzeigen können.
- Eine Ausnahme tritt bei Verbindungsfehlern auf.

- Das asynchrone Abholen führt dazu, dass die Methode intern selbst einen neuen Thread startet, der die Daten im Hintergrund abholt. Der Aufruf kehrt sofort zurück, liefert aber keine Daten zurück, weil diese ja noch abgeholt werden. Sie erhalten die neuen Daten durch den Event-Admin-Service mitgeteilt. Dieser sendet eine Nachricht mit dem Topic "de/hska/iwii/stockdata" (siehe Konstante `IStockQuotes.EVENT_TOPIC`). Die neuen Kursdaten erhalten Sie, wie immer beim Event-Admin-Service, als Property in einem Dictionary übergeben. Die Bundles verwenden den Property-Name "stockdata" (siehe Konstante `IStockQuotes.EVENT_STOCK_DATA`), dem eine List mit Objekten der Klasse `StockData` zugeordnet ist. Auch hier sind die Werte des vorherigen Aufrufs in den jeweiligen `StockData`-Objekten gespeichert. Die Property "fetch_exception" (siehe Konstante `IStockQuotes.EVENT_STOCK_FETCH_EXCEPTION`) enthält eine Information darüber, ob beim Abholen eine Ausnahme auftrat. War das der Fall, dann ist der Property das Exception-Objekt zugeordnet und die Kursdaten fehlen, ansonsten fehlt die Property. Signatur der Methode:

```
void requestDataAsync(String stockIndex):
```

Der Parameter `stockIndex` ist derselbe wie beim synchronen Aufruf. Die Methode startet nur dann einen Abholvorgang, wenn gerade kein anderer läuft.

3. Wenn Sie zwischen Simulation und echtem Server umschalten, dann müssen Sie in der neuen Datenquelle die eventuell noch vorhandenen alten Daten durch Aufruf der Methode `reset()` löschen.

2.3 Plug-in zur testweisen Ausgabe der Daten

Erstellen Sie ein neues OSGi-Projekt, mit dessen Hilfe Sie Ihr Plug-in aus Aufgabenteil 2.2 testen können.

3

RCP-Anwendung

Nachdem Sie inzwischen Erfahrungen mit OSGi gesammelt haben, werden eine RCP-Anwendung generiert und dessen Oberfläche in einem Fenster erzeugt. In dieser Anwendung werden später die bereitgestellten drei Plug-ins verwendet.

3.1 Vorbereitungen

Die folgenden Schritte zeigen Ihnen die wichtigsten Punkte, die Sie beim automatischen Erzeugen einer RCP-Anwendung beachten müssen.

3.1.1 Schritt 1

Erstellen Sie in Eclipse ein neues Plug-in-Projekt: „New“ → „Project...“ → „Eclipse 4“ → „Eclipse 4 Application Project“.

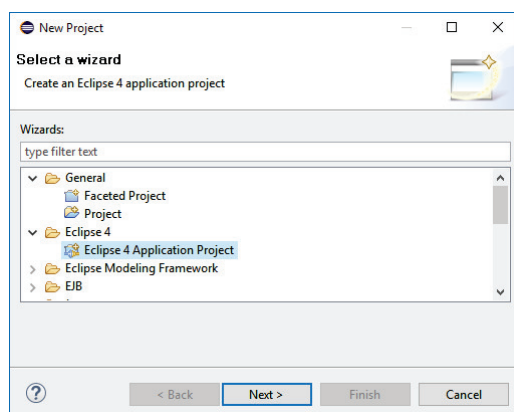


Abbildung 3.1: RCP-Projekt (Schritt 1)

3.1.2 Schritt 2

Geben Sie Ihrem Projekt einen sinnvollen Namen und passen Sie eventuell das Zielverzeichnis an.

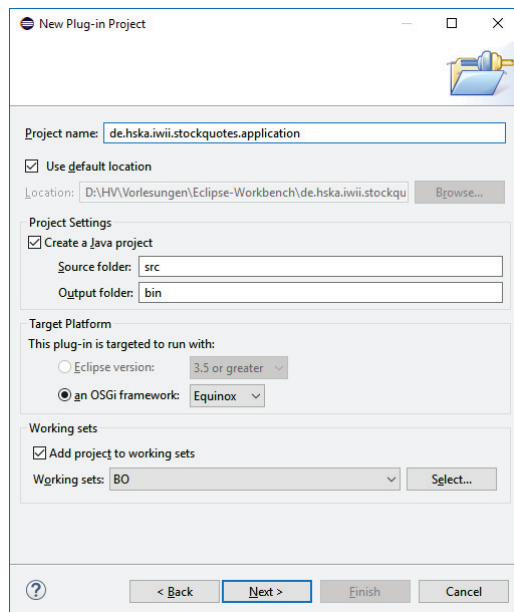


Abbildung 3.2: RCP-Projekt (Schritt 2)

3.1.3 Schritt 3

In Schritt 3 geben Sie Ihrem Plug-in eine Version. Einen „Activator“ benötigen Sie vermutlich nicht.

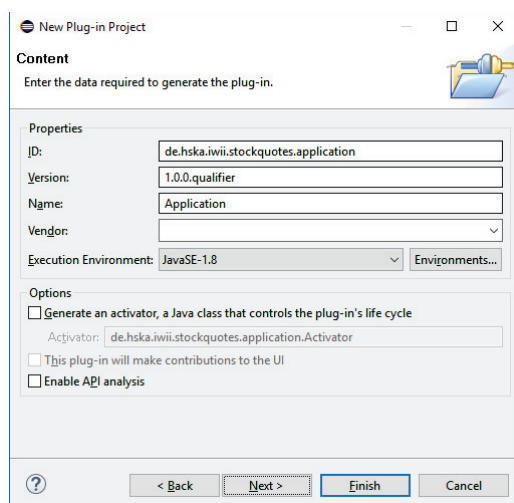


Abbildung 3.3: RCP-Projekt (Schritt 3)

3.1.4 Schritt 4

Auf der vierten Seite des Wizards sollten Sie „Create Sample Content“ auswählen, damit die Anwendung schon Menüs usw. besitzt. Diese Elemente könnten Sie später auch manuell ergänzen. Aber so sehen Sie schon einmal etwas, wenn Sie die Anwendung starten.

3.2 Layout

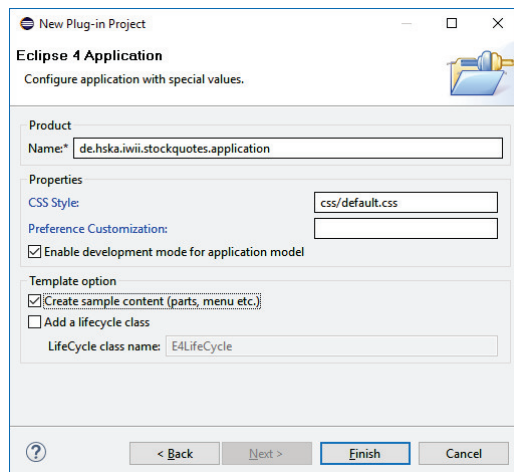


Abbildung 3.4: RCP-Projekt (Schritt 4)

3.1.5 Schritt 5 – Start der Anwendung

Jetzt sollte sich die Anwendung starten lassen. Auf Basis dieses Gerüsts können Sie Ihre Anwendung bauen. In der Produkt-Datei sehen Sie im Tab „Launching“ unter „Program Arguments“ den Eintrag `-clearPersistedState`. Damit werden bei jedem Programmstart alle automatisch gespeicherten Einstellungen (Positionen der Fenster usw.) gelöscht. Ansonsten werden Sie feststellen, dass nicht jede Änderung oder Erweiterung an Ihrem Programm sichtbar wird, weil mit gespeicherten Einstellungen gearbeitet wird.

3.2 Layout

In dieser Teilaufgabe wird das Layout der Oberfläche mittels Layout-Managern erstellt.

3.2.1 View

Diese Ansicht innerhalb des Fenster übernimmt die Darstellung der Börseninformationen. Sie sollte ungefähr so aussehen wie in der folgenden Abbildung 3.5. Die Tabelle erhält später einen sinnvollen Inhalt. Rechts sehen Sie Programm mit dem zu ergänzenen Menü.

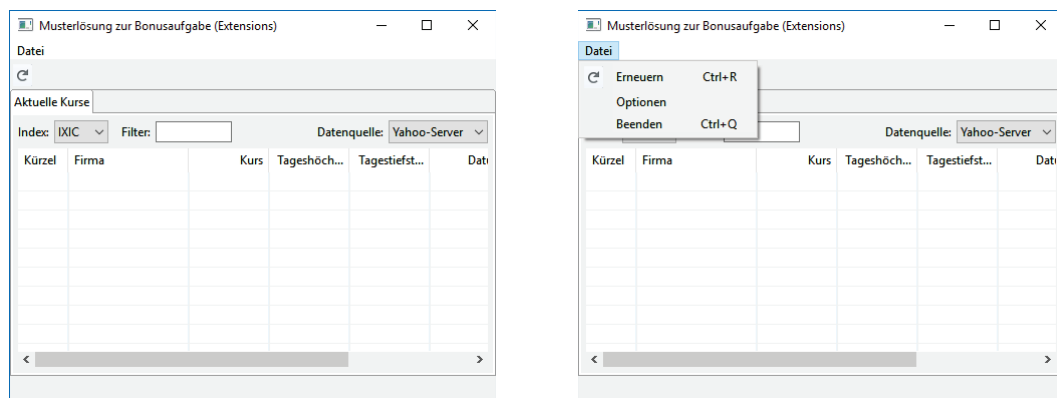


Abbildung 3.5: Darstellung der Börsenkurse

- Zur Vereinfachung können Sie gerne einen GUI-Editor wie den „WindowBuilder“ verwenden.
- In der Mitte befindet sich eine noch leere Tabelle mit Scroll-Balken. Die Tabelle wird in Abschnitt 3.4 mit Daten gefüllt.
- Die Combo-Box enthält die Einträge „IXIC“ (NASDAQ), „DJI“ (Dow Jones) und „FTSE“ (FTSE100). Sie dient später dazu, zwischen diesen Indizes umzuschalten. Tragen Sie vorerst einmal manuell die Börsenindizes in die Combo-Box ein. Später werden Sie die Werte aus den bereitgestellten Plug-ins zur Laufzeit auslesen.
- Das Textfeld „Filter“ erlaubt Ihnen, in der Tabelle nur die Börsenkurse anzuzeigen, die mit einer bestimmten Buchstabenfolge anfangen.
- Am unteren Rand des Fensters sehen Sie eine Statuszeile mit einem Fortschrittsbalken sowie einer textuellen Beschreibung der gerade laufenden Aktion.

3.2.2 Konfigurationsdialog

Der Zugang zum Börsenserver erfolgt über HTTP. Innerhalb der Hochschule ist dazu die Verwendung des Proxy-Servers zwingend erforderlich. In diesem Dialog werden die Proxy-Einstellungen vorgenommen. In der Praxis würde man hier die Klassen aus der Preferences-API verwenden. Diese wurden aber in der Vorlesung aus Zeitgründen weggelassen. Der Dialog sollte ungefähr so aussehen, wenn der Proxy verwendet bzw. nicht verwendet wird (alle Felder sowie die Label sind „enabled“):

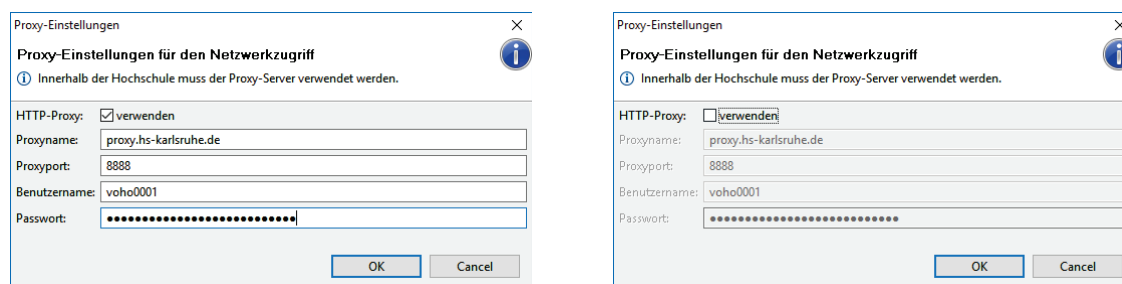


Abbildung 3.6: Eingeschalteter (links) und ausgeschalteter (rechts) Proxy

Diese Funktionalität wird später ergänzt. Weiterhin wird der Dialog in einem der folgenden Schritte durch Aufruf des Menüpunktes „Optionen“ angezeigt.

3.3 Ereignisbehandlung

Hier wird die Anwendung „mit Leben“ gefüllt, damit sie auf Benutzereingaben reagieren kann.

3.3.1 Ereignisbehandlung mit Handlern

Erstellen Sie die Handler-Klassen für die Menüeinträge sowie für die Toolbar-Taste.

- „Erneuern“: Lassen Sie die Klasse für das „Erneuern“ (Daten erneut abholen) zunächst nur eine Meldung auf der Konsole ausgeben.
- „Beenden“: Der Handler für die Beendigung der Anwendung wurde automatisch erzeugt und muss nicht mehr bearbeitet werden.
- „Optionen“: Lassen Sie den Handler den Proxy-Dialog anzeigen.

3.3.2 Ereignisbehandlung in der Ansicht

In der Ansicht müssen Sie diese Ereignisse behandeln:

- Auswahl des Börsenindexes
- textuelle Änderungen im Filter
- Auswahl der Datenquelle (Simulation oder Echtzeitdaten von Yahoo)

3.3.3 Ereignisbehandlung im Optionen-Dialog

Erstellen Sie ein Modell für die einzugebenden Daten und synchronisieren Sie diese Werte mit dem Modell. Bauen Sie auch die Umschaltung zwischen der Verwendung und Deaktivierung des Proxy-Servers ein.

3.4 Tabelle

In dieser Teilaufgabe befüllen Sie die Tabelle zunächst mit eigenen Daten.

3.4.1 Vorbereitung

Stellen Sie in der Konfiguration Ihres Projekts Abhängigkeiten zu den vorhandenen Plugins `de.hska.iwii.StockQuotes`, `de.hska.iwii.StockQuotesSimulation` und `de.hska.iwii.StockQuotesYahoo` her. So können Sie deren öffentliche Klassen und Schnittstellen nutzen.

3.4.2 Modell

Erweitern Sie die bisherige Lösung so, dass die später vom Server abgeholten oder per Simulation erzeugten Daten in der Tabelle angezeigt werden. Verwenden Sie dazu den MVC-Ansatz von JFace in Form des `TableViewer`s:

- Erstellen Sie ein eigenes Modell der Tabelle, das eine Anzahl von `StockData`-Objekten in einer `List` aufnimmt.
- Da bisher weder das Simulations- noch das Yahoo-Plug-in angebunden sind, befüllen Sie Ihr Modell mit einigen Dummy-Daten.
- Schreiben Sie einen `IStructuredContentProvider`, der Ihre Modelldaten für den `TableViewer` aufbereitet.

3.4.3 Darstellung

In dieser Aufgabe soll ein `ColumnLabelProvider` erstellt werden, der mit unterschiedlichen Hintergrundfarben anzeigt, wie sich der aktuelle Kurs verändert hat: Er ist gestiegen, abgesunken oder unverändert geblieben. Dazu benötigen Sie für alle Kurse deren vorherigen Wert. Dieser ist in der Klasse `StockData` als Attribut `previousPrice` abgelegt. Damit Sie die Veränderung leicht erkennen können, ist der Unterschied zum vorherigen Einlesen über die Methode `getPriceChange` der Klasse auslesbar. Die Methode liefert einen der drei Aufzählwerte je nach Änderung zurück:

- `StockData.CurrentPriceChange.UP`: Der Kurs ist seit der letzten Abfrage gestiegen.

3.4 Tabelle

- `StockData.CurrentPriceChange.DOWN`: Der Kurs ist seit der letzten Abfrage gesunken.
- `StockData.CurrentPriceChange.UNCHANGED`: Der Kurs ist entweder seit der letzten Abfrage unverändert geblieben, oder aber es handelt sich um die erste Abfrage. Dieser Wert wird auch dann zurückgegeben, wenn zwischenzeitlich ein anderer Börsenindex abgefragt wurde.

Der `ColumnLabelProvider` wird für die Spalte mit den aktuellen Kursdaten verwendet. Er zeigt die Kursänderungen durch eine Hintergrundfarbe an:

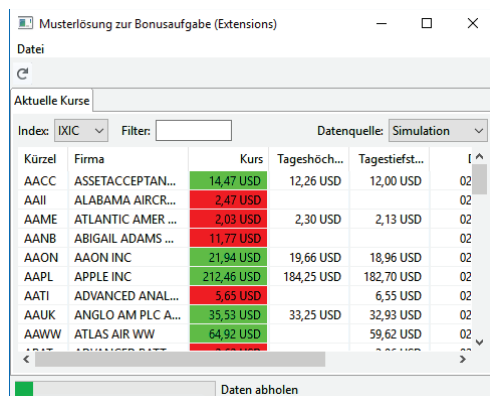
- keine besonderer Hintergrund: Der Kurs ist unverändert geblieben.
- grüne Hintergrundfarbe: Der Kurs hat sich erhöht.
- rote Hintergrundfarbe: Der Kurs ist gesunken.

Um den Aufwand der manuellen Ressourcen-Verwaltung zu umgehen, bietet es sich an, an dieser Stelle vordefinierte Systemfarben einzusetzen. Auf diese kann sehr einfach zugegriffen werden:

- Rot: `Display.getSystemColor(SWT.COLOR_RED)`
- Grün: `Display.getSystemColor(SWT.COLOR_GREEN)`

Da es sich um Systemfarben handelt, dürfen die Farbobjekte auch nicht mittels eines `dispose()`-Aufrufes freigegeben werden.

Schreiben Sie eine Klasse, die von `ColumnLabelProvider` erbt, und überschreiben Sie die Methoden `getText` und `getBackground` (siehe Beispiel in der Vorlesung). Das Ergebnis dieser Teilaufgabe sollte ungefähr so aussehen:



Kürzel	Firma	Kurs	Tageshöch...	Tagestiefst...	I
AACC	ASSETACCEPTAN...	14,47 USD	12,26 USD	12,00 USD	02
AAIL	ALABAMA AIRCR...	2,47 USD			02
AAME	ATLANTIC AMER ...	2,03 USD	2,30 USD	2,13 USD	02
AANB	ABIGAIL ADAMS ...	11,77 USD			02
AAON	AAON INC	21,94 USD	19,66 USD	18,96 USD	02
AAPL	APPLE INC	212,46 USD	184,25 USD	182,70 USD	02
AATI	ADVANCED ANAL...	5,65 USD		6,55 USD	02
AAUK	ANGLO AM PLC A...	35,53 USD	33,25 USD	32,93 USD	02
AAWW	ATLAS AIR WW	64,92 USD		59,62 USD	02

Abbildung 3.7: Hervorhebungen der Änderungen durch Farben

3.4.4 Filterung (optional)

Im Textfeld „Filter“ lassen sich die Anfangsbuchstaben der Firmenkurznamen eingeben, die Sie in der Tabelle sehen wollen. Alle anderen Einträge werden nicht dargestellt. Was Sie hier benötigen ist also ein Filter für die Tabelle, der bereits von JFace bereitgestellt wird. Nähere Informationen dazu finden Sie in den Vorlesungsunterlagen. Beachten Sie, dass die Filterung lediglich die Darstellung beeinflusst. Das Modell ist davon nicht betroffen. Die Abbildung 3.8 zeigt die Darstellung, wenn in der Tabelle nur noch die Kurse, deren Kurzname mit der Buchstabenfolge „ACA“ anfängt, zu sehen sind.


```
@Inject
private StockQuotesSimulation simulationSource;

@Inject
private StockQuotesYahoo yahooSource;
```

Prinzipiell ist es auch möglich, statt der konkreten Implementierungen die Schnittstelle `IStockQuotes` anzugeben. Allerdings wird dann zufällig einer der beiden Dienste ausgewählt.

3.5.3 Test der Datenabholung

Beim Ändern der Auswahl einer der Combo-Boxen oder beim Klick auf die Taste bzw. den Menüpunkt zum „Erneuern“ starten Sie das Abholen der Daten. Die API dazu ist bereits in Abschnitt 2.2.2 beschrieben. Verwenden Sie zum Testen zunächst das synchrone Abholen, und geben Sie die empfangenen Daten testweise auf der Konsole aus.

3.5.4 Anbindung an das Tabellenmodell

Statt die Daten einfach auf der Konsole auszugeben, tragen Sie die neuen Daten jetzt direkt in den `TableViewer` ein. Damit sollte sich die Tabelle automatisch aktualisieren.

3.6 Internationalisierung

Die Anwendung wird darauf vorbereitet, dass sie automatisch mehrere Sprachen unterstützen.

3.6.1 Statische Texte

Passen Sie Ihr Programm so an, dass Sie die Sprachen „deutsch“ und „englisch“ unterstützen. Am besten verwenden Sie dazu die Eclipse-Funktion „Externalize Strings“, die sich im Kontextmenü unter dem Punkt „Source“ verbirgt. Geben Sie allen Schlüsseln sinnvolle Bezeichnungen.

3.6.2 Tabellendaten

Die Daten, die die Tabelle visualisiert, müssen länderspezifisch dargestellt werden. Dazu kann der `ColumnLabelProvider` Zahlen mittels `NumberFormat` und Datumswerte mit `DateFormat` formatieren. Führen Sie das beispielhaft anhand der Spalte mit den aktuellen Kursen durch. Die Spalte beinhaltet eine Währung, die aber immer US-Dollar ist. Lediglich die Darstellung der Zahl muss angepasst werden.

3.7 Multithreading

Für das Abholen der Daten sollen zwei unterschiedliche Techniken ausprobiert werden, die natürlich nicht gleichzeitig eingesetzt werden. Passen Sie Ihr Programm so an, dass in einem festen Zeitabstand die neuen Kurse abgeholt bzw. erzeugt werden. Dieses geschieht am besten durch die `Timer`-Klasse aus dem Paket `java.util`.

3.7.1 Synchrones Abholen mit Jobs

Beim synchronen Zugriff auf externe Server kann es natürlich passieren, dass der Zugriff eine unbestimmte und sehr lange Zeit benötigt. Wenn Sie die Daten innerhalb des UI-Threads abholen, ist Ihre Anwendung für die Zeit des Datenzugriffs blockiert, was in der Praxis nicht akzeptabel ist. Daher müssen die Datenzugriffe in einen eigenen Thread ausgelagert werden. Verwenden Sie daher den Timer, um in einem festen Zeitraster einen `Job` zu starten.

3.7.1.1 Zugriff

Der Job führt einen eigenen Thread aus und trägt nach Abholen der Daten diese in den `TableViewer` ein. Das muss allerdings dann innerhalb des UI-Threads erfolgen. Die Methode `List<StockData> requestDataSync(String stockIndex)` ist in Abschnitt 2.2 beschrieben worden.

3.7.1.2 Fortschrittsdarstellung (optional)

Der Job soll einen aktiven Abholvorgang in der Statuszeile anzeigen. Dazu müssen Sie einen eigenen `IProgressMonitor` schreiben und diesen den Fortschrittsbalken in der Statuszeile aktualisieren lassen. Eine Referenzimplementierung finden Sie im Projekt `de.hska.iwii.MultithreadingE4RCPPProject` in den Vorlesungsbeispielen.

3.7.2 Asynchrones Abholen mit Events

Die Methode `void requestDataAsync(String stockIndex)` wird für den asynchronen Zugriff verwendet. Sie startet intern einen Thread, der den eigentlichen Abholvorgang durchführt. Der Aufruf kehrt sofort zurück, weswegen er auch aus dem UI-Thread erfolgen kann. Liegt das Ergebnis vor, dann sendet der Thread ein OSGi-Event über den Event-Admin-Service. Das genaue Verhalten ist auch hier in Abschnitt 2.2.2 beschrieben. Zum Empfang erstellen Sie eine Methode, die ein OSGi-Event mit dem Topic des Ereignisses injiziert bekommt. Aus Aufwandsgründen verzichten Sie hier am besten auf den Fortschrittsbalken in der Statuszeile. Achten Sie unbedingt darauf, in Ihrer Datei `plugin.xml` das Paket `org.eclipse.e4.core.services.events` zu importieren. Eine Abhängigkeit zu dem Plug-in, das dieses Paket beinhaltet, führt dazu, dass die falsche Annotation `@UIEventTopic` verwendet wird. Beachten Sie bitte, dass der Simulations- und der Yahoo-Dienst als Nachrichteninhalte Objekte der Klasse `org.osgi.service.event.Event` verschicken und Sie in der Empfangsmethode genau dieses Objekt als Parameter erwarten:

```
public void receive(@UIEventTopic(IStockQuotes.EVENT_TOPIC)
                    final Event message) {
    // ...
}
```

3.8 Sperren von Einträgen (optional)

Ein manuelles Abholen von Daten ist nicht sinnvoll, wenn gerade ein Abholvorgang läuft. Daher sollten bei aktivem Job die entsprechenden Tasten und Menüeinträge gesperrt werden. Dieses lässt sich recht einfach dadurch umsetzen, dass die Handler für das

„Erneuern“ selbst durch eine mit `@CanExecute` annotierte Methode bestimmen, ob sie ausführbar sind oder nicht.

4

Plug-in für die RCP-Anwendung (optional)

Schreiben Sie ein sehr einfaches Plug-in mit einer beliebigen Ansicht und einem Modell-Fragment, was sich automatisch in Ihre Anwendung integriert. Als Ansicht können Sie beispielsweise das `Browser-Widget` verwenden. In einer „echten“ Anwendung könnte man Plug-ins mit Auswertefunktionen oder Graphendarstellungen der Kurse einbinden.