

10 Management der OSGi Service Platform

In den Tutorials der zurückliegenden Kapitel haben Sie die Equinox-Konsole bereits als einfachen Management Agent kennengelernt, mit dem Sie die OSGi Service Platform zur Laufzeit überwachen und manipulieren können. Dabei haben wir die Konsole zur Anzeige des Framework-Status und zur Steuerung des Bundle-Lebenszyklus verwendet. Die Equinox-Konsole stellt darüber hinaus eine Reihe weiterer Kommandos bereit, die zur Administration der OSGi Service Platform eingesetzt werden können.

Motivation

Alternativ zur Equinox-Konsole, die ein integraler Bestandteil der Equinox-Distribution ist, existieren im Open-Source-Bereich eine Reihe von alternativen Management Agents, die ebenfalls zusammen mit Eclipse Equinox eingesetzt werden können.

In diesem Kapitel gehen wir sowohl auf die Möglichkeiten der Equinox-Konsole als auch auf die Verwendung alternativer Management Agents ein. Sie lernen die Kommandos der Equinox-Konsole kennen und erfahren, wie Sie die Equinox-Konsole um eigene Kommandos erweitern können. Darüber hinaus zeigen wir Ihnen am Beispiel des Knopflerfish-Desktops, wie Sie einen anderen Management Agent in Eclipse Equinox installieren und zur Administration der OSGi Service Platform verwenden.

Die in diesem Kapitel vorgestellten Konzepte sind in weiten Teilen Equinox-spezifisch, da in der OSGi-Spezifikation nicht festgelegt ist, welche Funktionalität Management Agents bereitzustellen haben oder wie Sie diese bereitstellen.

Einordnung

Stattdessen beschreibt die Spezifikation, wie die programmatische Schnittstelle (API) der OSGi Service Platform aussieht, über die die potenziellen Management Agents Zugriff auf das OSGi Framework bzw. die Standard Services erhalten. Aus diesem Grund sind Management Agents, die ausschließlich diese Schnittstelle verwenden, implementierungsübergreifend nutzbar.

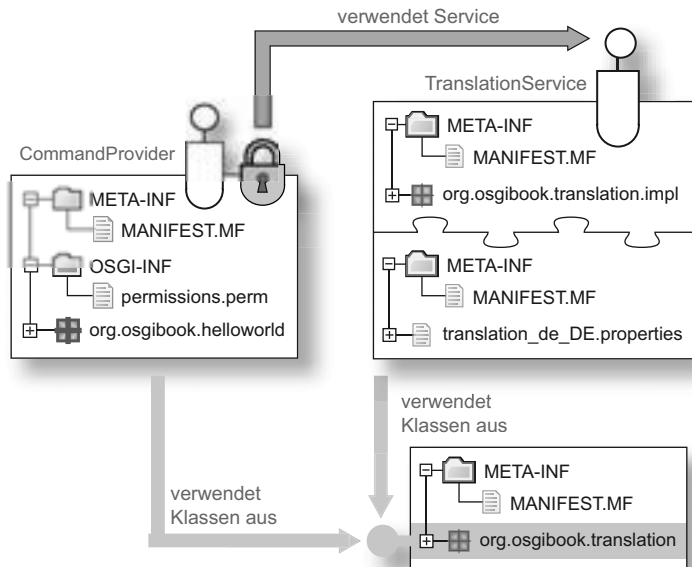
10.1 Tutorial: Die Equinox-Konsole um ein greet-Kommando erweitern

Überblick Im Tutorial zu diesem Kapitel erweitern wir die Equinox-Konsole um einen neuen, eigenen Befehl greet. Wir bauen das »Hello World«-Beispiel so um, dass der Gruß nicht mehr implizit beim Starten und Stoppen des Bundles `org.osgibook.helloworld` ausgegeben, sondern explizit durch das neue Kommando auf der Equinox-Konsole aufgerufen wird.

Um die Equinox-Konsole um den neuen Befehl zu erweitern, müssen wir einen `CommandProvider` implementieren, der das `greet`-Kommando für die Equinox-Konsole bereitstellt. Der `CommandProvider` wird als Service an der Service Registry angemeldet und stellt so das `greet`-Kommando über das Whiteboard-Pattern (vgl. Unterkapitel 7.6) in der Equinox-Konsole bereit (vgl. Abb. 10–1).

Abb. 10–1

Übersicht über die beteiligten Bundles



Die Plug-in-Projekte

Wir erweitern im Tutorial zu diesem Kapitel lediglich den Activator des Plug-in-Projektes `org.osgibook.helloworld`. Alle weiteren Plug-in-Projekte bleiben unverändert. Für den Fall, dass Sie das Beispiel mit der in Kapitel 9 angelegten »Hello World Example (SecurityManager)«-Launch-Konfiguration starten möchten, müssen Sie jedoch zusätzlich die lokalen Ausführungsrechte des Plug-in-Projektes `org.osgibook.helloworld` anpassen.

Schritt 1: Anpassen der start()- und stop()-Methoden

Da die Grußbotschaft nicht mehr beim Start und beim Stopp des »Hello World«-Bundles ausgegeben werden soll, benötigen Sie im Gegensatz zum bisherigen Beispiel keinen Service Tracker mehr, der das Ausgeben der Grußbotschaft beim An- und Abmelden des Translation Service übernimmt. Stattdessen verwenden wir einen »einfachen« Service Tracker, über den wir den Translation Service abfragen, sobald ein Anwender das greet-Kommando auf der Konsole ausgeführt hat.

Entfernen Sie deshalb den bislang verwendeten TranslationServiceTracker aus dem Activator. Instanzieren Sie in der start()-Methode stattdessen einen normalen Service Tracker (vgl. Listing 10–1):

```
package org.osgibook.helloworld;

[...]
```

```
public class Activator implements BundleActivator {

    private BundleContext bundleContext;

    private ServiceTracker translationServiceTracker;

    public void start(BundleContext context) throws Exception {
        this.bundleContext = context;

        translationServiceTracker = new ServiceTracker(context,
            TranslationService.class.getName(), null);
        translationServiceTracker.open();
    }

    public void stop(BundleContext context) throws Exception {
        translationServiceTracker.close();
    }

    [...]
}
```

Listing 10–1

Anpassungen im
Activator des »Hello
World«-Bundles I

Schritt 2: Implementierung des CommandProvider-Interface

Der Equinox-Konsole können eigene Kommando hinzugefügt werden, indem eine Klasse vom Typ `org.eclipse.osgi.framework.console.CommandProvider` implementiert und ein Exemplar dieser Klasse als Service an der Service Registry angemeldet wird.

In diesem Tutorial implementieren wir der Einfachheit halber das CommandProvider-Interface direkt am Activator des »Hello World«-Bundles. Beachten Sie, dass Sie dafür im Bundle Manifest das Package `org.eclipse.osgi.framework.console` importieren müssen, damit die entsprechenden Klassen zur Verfügung stehen:

Import der
benötigten Packages

```
Import-Package: org.eclipse.osgi.framework.console
```

*Implementierung des
CommandProvider-
Interface*

Die einzige Methode, die vom `CommandProvider`-Interface definiert wird, ist die Methode `getHelp()`, die in unserem Fall die Beschreibung des Kommandos `greet` zurückliefert. In der `start()`-Methode des `Activator` muss die `CommandProvider`-Instanz, in unserem Fall also der `Activator` selber, unter dem `CommandProvider`-Interface an der Service Registry angemeldet werden (vgl. Listing 10–2).

Listing 10–2

*Anpassungen im
Activator des »Hello
World«-Bundles II*

```
package org.osgibook.helloworld;

import org.eclipse.osgi.framework.console.CommandProvider;

[...]
```

```
public class Activator implements BundleActivator, CommandProvider
{
    [...]

    public void start(BundleContext context) throws Exception {
        this.bundleContext = context;

        context.registerService(CommandProvider.class.getName(),
            this, null);

        translationServiceTracker = new ServiceTracker(context,
            TranslationService.class.getName(), null);
        translationServiceTracker.open();
    }

    [...]

    public String getHelp() {
        StringBuilder help = new StringBuilder();
        help.append("\n--- Hello World Commands ---");
        help.append("\n\tgreet [hello|goodbye] - ");
        help.append("display the \"Hello World !\" message");
        return help.toString();
    }
}
```

Schritt 3: Implementierung des greet-Kommandos

Um ein neues Kommando für die Konsole zur Verfügung zu stellen, muss der `Command Provider` eine öffentliche Methode implementieren, die nach dem Kommando benannt ist und mit einem führenden Unterstrich beginnt – in unserem Fall also `_greet`. Die Signatur der Methode muss genau einen Parameter vom Typ `org.eclipse.osgi.framework.console.CommandInterpreter` besitzen.

Dieses `CommandInterpreter`-Objekt wird genutzt, um auf weitere Argumente, die der Benutzer zusammen mit dem Kommando eingegeben hat, zuzugreifen. Außerdem können Sie darüber mit der Konsole interagieren, also z.B. Texte ausgeben oder andere Kommandos ausführen.

Bitte legen Sie also die Methode `_greet()` an, die die Eingabe des Benutzers entgegennimmt und eine entsprechende Grußbotschaft auf der Konsole ausgibt. Die bisherige `greet()`-Methode können Sie aus der `Activator`-Klasse entfernen, da sie nicht mehr benötigt wird.

```
package org.osgi.framework.helloworld;

import org.eclipse.osgi.framework.console.CommandInterpreter;

[...]
```

```
public class Activator
    implements BundleActivator, CommandProvider {

    [...]

    public void _greet(CommandInterpreter ci) {
        TranslationService translationService =
            (TranslationService) translationServiceTracker
                .getService();
        if (translationService == null) {
            commandInterpreter.println
                ("TranslationService z.Zt. nicht verfuegbar.");
            return;
        }
        String key = commandInterpreter.nextArgument();
        if (key == null) {
            // Wenn keine Nachricht eingegeben wurde, hello ausgeben
            key = "hello";
        }
        String translatedMsg =
            translationService.getTranslation(key);
        String message = String.format(translatedMsg,
            bundleContext.getBundle().getSymbolicName());
        // Komplette Nachricht auf der Konsole ausgeben
        commandInterpreter.println(message);
    }
}
```

Listing 10-3*Das _greet-Kommando***Schritt 4: Anpassen der benötigten Ausführungsrechte**

Falls Sie das geänderte Beispiel mit der in Kapitel 9 erstellten »Hello World Example (SecurityManager)«-Launch-Konfiguration starten möchten, dann müssen Sie die bereits definierten Rechte des Bundles `org.osgibook.helloworld` erweitern. Fügen Sie dazu in der Datei `OSGI-`

INF/permission.perm des Bundles org.osgibook.helloworld den folgenden Eintrag hinzu:

```
(org.osgi.framework.ServicePermission ↵  
"org.eclipse.osgi.framework.console.CommandProvider" "register")
```

Schritt 5: Ausführen des Beispiels

Wenn Sie das Beispiel nun über die »Hello World Example«- bzw. die »Hello World Example (SecurityManager)«-Launch-Konfiguration ausführen, wird beim Start des »Hello World«-Bundles keine Grußbotschaft mehr ausgegeben:

```
osgi>
```

Erst wenn Sie den Befehl greet auf der Konsole eingeben, erscheint die Grußbotschaft in der bekannten Form auf der Konsole:

```
osgi> greet hello  
Hallo OSGi-Welt sagt Bundle org.osgibook.helloworld!
```

```
osgi> greet goodbye  
Tschüß OSGi-Welt sagt Bundle org.osgibook.helloworld!
```

10.2 Management Agents im Überblick

Die Verwaltung eines OSGi Frameworks und die Manipulation von Bundle-Zuständen erfolgt über einen sog. *Management Agent*. Ein Management Agent implementiert eine Benutzungsschnittstelle zum OSGi Framework, über die das Framework »von außen« administriert werden kann.

10.2.1 Management Bundles

Ein Management Agent ist selber in einem oder mehreren Bundles implementiert, die innerhalb des Frameworks installiert sind (den sog. *Management Bundles*). Wie ein Management Agent konkret implementiert wird oder welche Benutzungsschnittstelle er bereitstellt, ist in der OSGi-Spezifikation nicht festgelegt. Sie beschreibt lediglich die API, über die Management Bundles das Framework programmatisch verwalten und manipulieren können (vgl. Abb. 10–2).

Management Bundles, die ausschließlich die spezifizierten, »offiziellen« Schnittstellen der OSGi Service Platform nutzen, können entsprechend in beliebigen Implementierungen des OSGi Frameworks verwendet werden. So ist es möglich, Management Agents vollständig unabhängig von einer konkreten Framework-Implementierung zu entwickeln.

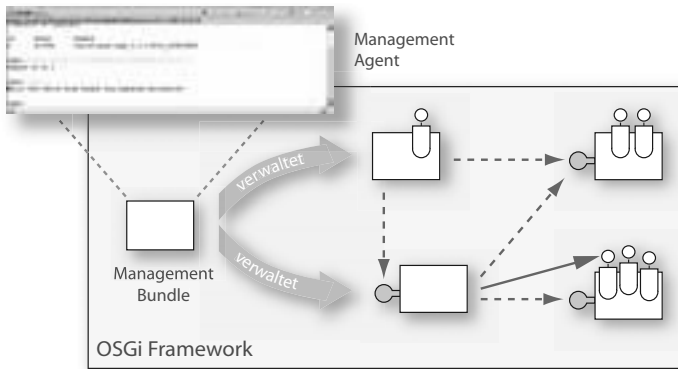


Abb. 10-2
Management Bundles im
OSGi Framework

10.2.2 Open-Source-Implementierungen

Im Open-Source-Bereich existieren ganz unterschiedliche Management-Agent-Implementierungen, die zusammen mit Eclipse Equinox nutzbar sind:

- **Equinox-Konsole:** Die Equinox-Konsole ist ein integraler Bestandteil des Equinox-Frameworks und ermöglicht einen einfachen, kommandozeilenbasierten Zugang zum Framework. Für die Remote-Administration des OSGi Frameworks kann die Konsole auch über Telnet betrieben werden.

In den nachfolgenden Unterkapiteln stellen wir Ihnen die verschiedenen Kommandos der Equinox-Konsole vor (Unterkapitel 10.3) und zeigen Ihnen, wie Sie die Equinox-Konsole um eigene Kommandos erweitern können (Unterkapitel 10.4).

- **Knopflerfish-Desktop:** Der Knopflerfish-Desktop [KNODES] ist Bestandteil der OSGi-Implementierung Knopflerfish und stellt eine auf Swing basierende, grafisch-interaktive Oberfläche zur Verfügung. Er kann losgelöst vom Knopflerfish-Framework in jedem R4-kompatiblen OSGi Framework eingesetzt werden.

Stellvertretend für die verschiedenen Management-Agent-Implementierungen zeigen wir Ihnen in Unterkapitel 10.5, wie Sie den Knopflerfish-Desktop als alternativen Management Agent in Eclipse Equinox installieren und ausführen.

- **Knopflerfish-HTTP/HTML-Konsole:** Ebenfalls Bestandteil der Knopflerfish-Implementierung ist die Knopflerfish-HTTP/HTML-Konsole [KNOHTM]. Die HTTP/HTML-Konsole implementiert ein Servlet, das auf jeder OSGi-Plattform ausgeführt werden kann, in der ein OSGi Http Service installiert ist. Es ermöglicht die Administration des OSGi Frameworks über das Web (vgl. Abb. 10-3). Damit die HTTP/HTML-Konsole in Eclipse Equinox installiert und ausgeführt werden kann, muss neben dem Bundle mit der

Standard-Service-API auch die Servlet-API und die Implementierung des Http Service installiert und gestartet sein.

Abb. 10-3

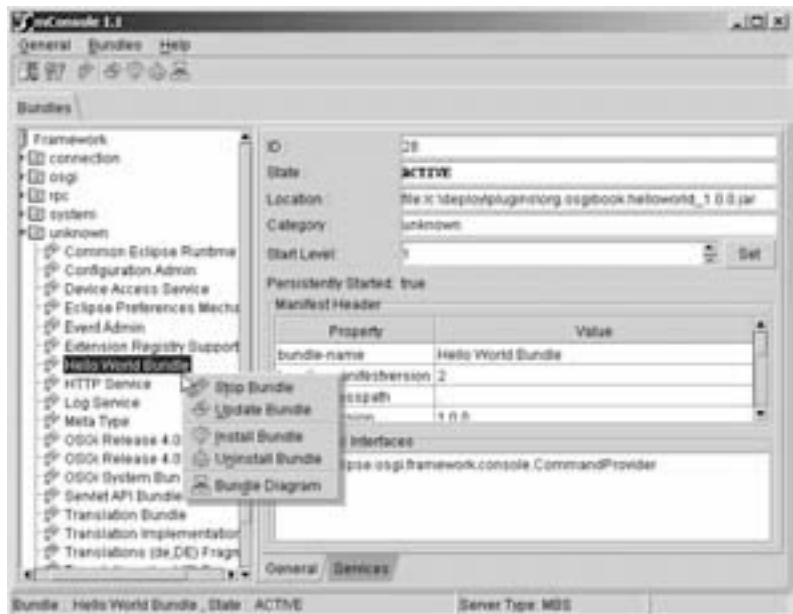
Die Knopflerfish-HTML/HTTP-Konsole



- **Prosynt mConsole:** Die leistungsfähige, grafisch-interaktive Prosynt mConsole ist Bestandteil der Prosynt mBedded Server Equinox Edition [PROMEE] und ermöglicht ebenfalls die entfernte Administration des OSGi Frameworks, wobei ein eigenes Kommunikationsprotokoll zum Einsatz kommt (vgl. Abb. 10-4).

Abb. 10-4

Die Prosynt mConsole



10.3 Die Equinox-Konsole im Detail

Die Equinox-Konsole implementiert eine einfache, kommandozeilenorientierte Benutzungsschnittstelle zur Verwaltung des Equinox OSGi Frameworks. Sie stellt eine Reihe von Befehlen zur Verfügung, mit denen der Zustand des Frameworks und der installierten Bundles zur Laufzeit abgefragt und manipuliert werden kann (vgl. auch [EQXCON]). Wir stellen Ihnen nachfolgend die Kommandos mit ihrer jeweiligen Syntax und einer erklärenden Beschreibung vor. Für einige Kommandos können Sie Abkürzungen verwenden, die in der Tabelle mit [] gekennzeichnet sind.

10.3.1 Kommandos zur Steuerung des Frameworks

Zur Steuerung des OSGi Frameworks stehen die in Tabelle 10–1 aufgeführten Kommandos zur Verfügung.

Kommando	Beschreibung
shutdown	Führt das OSGi Framework herunter. Der Zustand aller installierten Bundles wird dabei persistiert.
launch	Startet das OSGi Framework, wenn es nicht bereits gestartet ist. Der Zustand von persistierten Bundles wird dabei wiederhergestellt.
close	Führt das OSGi Framework herunter (shutdown) und beendet die virtuelle Maschine.
exit	Beendet die virtuelle Maschine unmittelbar (System.exit()).
init	Deinstalliert alle im Framework installierten Bundles. Dieses Kommando darf nur aufgerufen werden, wenn das Framework gestoppt ist.
gc	Führt eine Garbage Collection durch.
[setp]rop <name>=<wert>	Setzt die spezifizierte OSGi Property mit dem spezifizierten Wert.

Tab. 10–1
Kommandos zur
Steuerung des
Frameworks

10.3.2 Kommandos zur Steuerung von Bundles

Zur Steuerung von Bundles stehen die folgenden, in Tabelle 10–2 aufgeführten Befehle zur Verfügung. Bei den Bundle-spezifischen Kommandos müssen die Bundles, auf die sich das Kommando beziehen soll, über zusätzliche Kommandoparameter spezifiziert werden. Dabei sind alternativ folgende Angaben möglich:

- **Symbolischer Name:** Der symbolische Name des Bundles in der Form `symbolicname[@version]`.
Beispiel: `org.osgibook.helloworld@1.0.0`
- **Eindeutige Id:** Die eindeutige Id des Bundles, die bei seiner Installation vom Framework zugewiesen wurde.
Beispiel: `12`
- **Location:** Der Location-String des Bundles. Für gewöhnlich bezeichnet der Location-String den Ort, von dem das Bundle installiert wurde.
Beispiel: `file:/c:\\workspace\\org.osgibook.helloworld`

Tab. 10–2

Kommandos zur
Steuerung von Bundles

Kommando	Beschreibung
<code>[i]install <URL> {[s]tart}</code>	Installiert das Bundle mit der angegebenen URL und startet es ggf. Die URL kann unterschiedliche Protokolle besitzen, z.B. <code>file:/<Pfad zur Bundle></code> oder <code>http://<Pfad zum Bundle></code>
<code>[un]install (<id> <loc> <name>)+</code>	Deinstalliert das/die angegebene(n) Bundle(s).
<code>[sta]rt (<id> <loc> <name>)+</code>	Startet das/die angegebene(n) Bundle(s).
<code>[sto]p (<id> <loc> <name>)+</code>	Stoppt das/die angegebene(n) Bundle(s).
<code>[r]efresh (<id> <loc> <name>)+</code>	Löst die Abhängigkeiten des/der angegebene(n) Bundle(s) neu auf.
<code>[up]date * (<id> <loc> <name>)</code>	Aktualisiert das angegebene Bundle. Wenn als Argument der Wert <code>''</code> übergeben wird, dann werden alle installierten Bundles aktualisiert.

10.3.3 Kommandos zum Anzeigen von Statusinformationen

Zur Anzeige von Statusinformationen stehen die folgenden, in Tabelle 10–2 aufgeführten Befehle zur Verfügung.

Tab. 10–3

Kommandos zur Anzeige
von Statusinformationen

Kommando	Beschreibung
<code>ss {-s <STATE>} {<teil-des-symbolischen-Namens>}</code>	Zeigt die kurze Version der Statusanzeige an (short status). Wird über den Parameter <code>-s</code> ein Bundle-Zustand angegeben, werden nur die Bundles in diesem Zustand angezeigt. Zusätzlich kann ein String angegeben werden, der als Substring im symbolischen Namen der anzuzeigenden Bundles vorkommen muss.

Kommando	Beschreibung
[s]tatus {-s <STATE>} {<teil-des-symbolischen-Namens>}	Zeigt detaillierte Informationen zu den installierten Bundles und Services an. Wird über den Parameter -s ein Bundle-Zustand angegeben, werden nur die Bundles in diesem Zustand angezeigt. Zusätzlich kann ein String angegeben werden, der als Substring im symbolischen Namen der anzuzeigenden Bundles vorkommen muss.
[b]undle (<id> <loc> <name>)+	Zeigt detaillierte Informationen über das/die angegebene(n) Bundle(s) an.
bundles {-s <STATE>} {<teil-des-symbolischen-Namens>}	Zeigt detaillierte Informationen zu den installierten Bundles an. Wird über den Parameter -s ein Bundle-Zustand angegeben, werden nur die Bundles in diesem Zustand angezeigt. Zusätzlich kann ein String angegeben werden, der als Substring im symbolischen Namen der anzuzeigenden Bundles vorkommen muss.
(h)eaders (<id> <location> <symbolicname>)*	Zeigt alle Einträge der Manifest-Datei des/der angegebenen Bundle(s) an.
getPackages (<id> <location> <symbolicname>)	Zeigt alle von anderen Bundles exportierte Packages an, auf die vom angegebenen Bundle zugegriffen werden kann.
(p)ackages (<id> <location> <symbolicname>)	Zeigt alle exportierten Packages eines Bundles an. Zusätzlich wird für jedes exportierte Bundle angezeigt, von welchem Bundle das Package importiert wird.
requiredBundles <symbolicname>	Zeigt für das Bundle mit dem übergebenen symbolischen Namen alle Bundles an, die das Bundle über den Manifest Header Require-Bundle importieren.
diag (<id> <symbolicname>)*	Zeigt alle unerfüllten Bedingungen für das/die angegebene(n) Bundle(s) an.
(se)rvice(s) {<filter>}	Zeigt alle Services an, die dem angegebenen Filter genügen. Wird kein Filter angegeben, dann werden alle Services angezeigt.
(t)hreads	Zeigt die aktuellen Threads und Threadgroups an.
(pr)ops	Zeigt alle System-Properties an.
getprop {<anfang-des-namens>}	Zeigt alle System-Properties an, die mit dem übergebenen Namen anfangen. Wird kein Parameter übergeben, werden alle Properties ausgegeben.

10.3.4 Kommandos zur Manipulation der Startlevel

Zur Kontrolle und Manipulation der Bundle- und Framework-Startlevel stehen Ihnen die in Tabelle 10–4 aufgeführten Kommandos zur Verfügung.

Tab. 10–4
Kommandos zur
Manipulation der
Startlevel

Kommando	Beschreibung
sl (<id> <location> <symbolicname>)	Zeigt den Startlevel für das angegebene Bundle an. Wenn kein Bundle spezifiziert wird, dann wird der Startlevel des Frameworks angezeigt.
setfwsl <startlevel>	Setzt den aktuellen Startlevel des Frameworks.
setbsl <startlevel> (<id> <location>)	Setzt den Startlevel für ein Bundle. Das Bundle können Sie über dessen Id oder seine Location angeben.
setibsl <startlevel>	Setzt den Startlevel, der für ein Bundle verwendet wird, wenn für das Bundle kein Startlevel explizit angegeben wurde.

10.3.5 Weitere Kommandos

Zusätzlich zu den bereits vorgestellten Kommandos bietet die Equinox-Konsole die in Tabelle 10–5 aufgeführten Kommandos.

Tab. 10–5
Weitere Kommandos

Kommando	Beschreibung
exec	Führt ein Betriebssystem-Kommando in einem eigenen Prozess aus und wartet auf dessen Beendigung.
fork	Führt ein Betriebssystem-Kommando in einem eigenen Prozess aus und kehrt unmittelbar zurück.

10.4 Die Equinox-Konsole um eigene Befehle erweitern

Mit den in Abschnitt 10.3 vorgestellten Kommandos besitzt die Equinox-Konsole bereits eine große Anzahl an Befehlen, mit denen sich das Framework manipulieren und überwachen lässt. Trotzdem kann es notwendig oder wünschenswert sein, die Equinox-Konsole um eigene Kommandos zu erweitern, die speziell auf eine Problemdomäne zugeschnittene Anforderungen realisieren. Die Implementierung zusätzlicher Kommandos für die Equinox-Konsole ist durch die Anwendung des Whiteboard-Patterns (vgl. Abschnitt 7.6) einfach realisierbar.

10.4.1 Das Interface `CommandProvider`

Um ein (oder mehrere) Kommandos in der Equinox-Konsole bereitzustellen, müssen die Kommandos in einer Klasse vom Typ `org.eclipse.osgi.framework.console.CommandProvider` implementiert und unter diesem Interface an der OSGi Service Registry angemeldet werden. Die Equinox-Konsole erkennt die angemeldeten `CommandProvider` und delegiert die eingegebenen Kommandos entsprechend an die Services.

Das `CommandProvider`-Interface definiert lediglich die Methode `getHelp()` (vgl. Listing 10-4):

```
package org.eclipse.osgi.framework.console;

public interface CommandProvider {
    public String getHelp();
}
```

Listing 10-4*Das Interface**CommandProvider*

Die `getHelp()`-Methode liefert einen String zurück, der eine Beschreibung der Kommandos enthält, die Sie implementieren möchten. Equinox ruft diese Methode auf, wenn das `help`-Kommando von der Konsole ausgeführt wurde, um dem Anwender eine Beschreibung aller registrierten Kommandos auszugeben.

Um eigenen Kommandos für die Konsole zur Verfügung zu stellen, müssen Sie in Ihrem Command Provider für jedes Kommando eine öffentliche Methode implementieren, die genauso heißt wie das Kommando, allerdings mit einem Unterstrich beginnt. Die Signatur der Methode muss genau einen Parameter vom Typ `CommandInterpreter` besitzen. Für das `greet`-Kommando müssen Sie demnach die folgende Methode implementieren:

*Implementierung**der Kommandos*

```
public void _greet(CommandInterpreter commandInterpreter) {
    [...]
}
```

10.4.2 Das Interface `CommandInterpreter`

Das in der Implementierung des Kommandos übergebene `CommandInterpreter`-Objekt erlaubt es Ihnen, mit der Konsole zu interagieren:

```
package org.eclipse.osgi.framework.console;

import java.util.Dictionary;
import org.osgi.framework.Bundle;

public interface CommandInterpreter {
    public String nextArgument();
}
```

Listing 10-5*Das Interface**CommandInterpreter*

```
public Object execute(String cmd);

public void print(Object o);
public void println();
public void println(Object o);
public void printStackTrace(Throwable t);
public void printDictionary(Dictionary dic, String title);
public void printBundleResource(Bundle bundle, String resource);
}
```

Das `CommandInterpreter`-Objekt stellt Ihnen drei Typen von Methoden zur Verfügung:

- Mit der Methode **nextArgument()** können Sie auf die Argumente, die der Anwender Ihrem Kommando übergeben hat, zugreifen. Diese Methode können Sie so lange aufrufen, bis Sie null zurückgeliefert bekommen, in diesem Falle stehen keine weiteren Argumente zur Verfügung.
- Die Methode **execute()** ermöglicht es Ihnen per API Kommandos auszuführen. Sie können der Methode dazu das entsprechende Kommando mit all seinen Parametern übergeben – genauso, wie Sie es auch von der Equinox-Konsole aus tun würden.
- Die verschiedenen **print**-Methoden können Sie verwenden, um Texte auf der Equinox-Konsole auszugeben. Bei der Ausgabe der Texte berücksichtigt Equinox auch die Einstellung des `more`-Kommandos: Wenn Sie mehr Zeilen ausgeben, als der Anwender auf einmal sehen möchte, unterbricht Equinox die Ausgabe nach der entsprechenden Zeile so lange, bis der Anwender eine Taste zum Fortfahren gedrückt hat (analog zu den aus Windows und Unix bekannten »more«-Kommandos).

10.5 Exkurs: Der Knopflerfish-Desktop als alternativer Management Agent

In diesem Abschnitt stellen wir Ihnen mit dem *Knopflerfish-Desktop* einen alternativen Management Agent vor. Im Gegensatz zur kommandozeilenorientierten Schnittstelle der Equinox-Konsole bietet er eine auf Swing basierende, grafisch-interaktive Oberfläche (vgl. Abb. 10–5).

Der Knopflerfish-Desktop ist Bestandteil der OSGi-Implementierung Knopflerfish, die unter [KNOPF] verfügbar ist, kann aber losgelöst vom Knopflerfish-Framework in jedem R4-kompatiblen OSGi Framework eingesetzt werden.



Abb. 10-5
Der Knopflerfish-
Desktop

10.5.1 Installation des Knopflerfish-Desktops

Der Knopflerfish-Desktop besteht aus den zwei Bundles

- org.knopflerfish.bundle.util-LIB und
- org.knopflerfish.bundle.desktop,

die beide sowohl über die Homepage zu diesem Buch unter <http://www.osgibook.org/repo> als auch über die Knopflerfish-Homepage <http://www.knopflerfish.org/repo> verfügbar sind. Sie können die Bundles (mittels der Equinox-Konsole) über das HTTP-Protokoll direkt in Equinox installieren. Starten Sie dazu das Equinox Framework mit der Launch-Konfiguration »Equinox Console«. Zur Installation der beiden Knopflerfish-Bundles geben Sie nachfolgenden die Kommandos auf der Equinox-Konsole ein:

```
osgi>install http://www.osgibook.org/repo/jars/util/util- ↵
2.0.0.jar start
```

```
Bundle id is 1
```

```
osgi>install http://www.osgibook.org/repo/jars/desktop/ ↵
desktop_all-2.0.0.jar start
```

```
Bundle id is 2
```

Tip

Wenn Sie über einen Proxy auf das Internet zugreifen, müssen Sie den Proxy in der Launch-Konfiguration konfigurieren, damit Equinox eine HTTP-Verbindung zum Knopflerfish-Repository aufbauen kann. Geben Sie in diesem Fall auf der Seite **Arguments** der Launch-Konfiguration unter **VM arguments** die folgenden System-Properties an:

```
-DproxySet=true -DproxyHost=<<proxyHost> -DproxyPort=<<proxyPort>.
```

10.5.2 Arbeiten mit dem Knopflerfish-Desktop

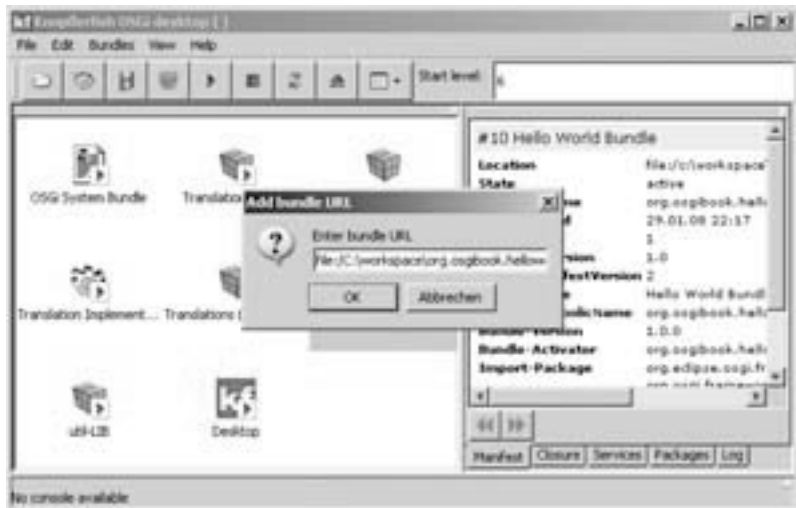
Der Knopflerfish-Desktop wird mit dem Start des Desktop-Bundles geöffnet und zeigt eine grafische Übersicht des OSGi Frameworks an. Sämtliche Bundle-Lebenszyklus-Operationen (Installieren, Starten, Stoppen, Aktualisieren und Deinstallieren) können über den Desktop vorgenommen werden. Zusätzlich werden Bundle- und Service-Details angezeigt.

Um ein Bundle über den Knopflerfish-Desktop zu installieren, öffnen Sie bitte über **File -> Add bundle URL...** (oder über die Tastenkombination Strg+u) den »Add bundle URL«-Dialog. In diesem Dialog können Sie die URL des zu ladenden Bundles eintragen, also bspw. `file:/c:/workspace/org.osgibook.helloworld` (vgl. Abb. 10–6).

Nach Betätigen des OK-Buttons wird das angegebene Bundle im Framework installiert. Über die Schaltflächen in der Toobar können Sie die Bundles jederzeit stoppen, aktualisieren usw. Weitere Informationen zum Knopflerfish-Desktop finden Sie unter [KNODES].

Abb. 10–6

Installation eines Bundles
im Knopflerfish-Desktop



10.6 Zusammenfassung

In diesem Kapitel haben Sie gelernt, wie Sie Management Agents nutzen, um den Zustand des OSGi Frameworks zu administrieren:

- Ein *Management Agent* implementiert eine Benutzungsschnittstelle zum OSGi Framework, über die das Framework »von außen« administriert werden kann.
- Ein Management Agent ist selber in einem oder mehreren Bundles implementiert, die innerhalb des Frameworks installiert sind (den sog. *Management Bundles*).
- Die OSGi-Spezifikation spezifiziert eine API, über die Management Bundles das Framework programmatisch verwalten und manipulieren können. Management Bundles, die ausschließlich die spezifizierten Schnittstellen der OSGi Service Platform nutzen, können in beliebigen Framework-Implementierungen eingesetzt werden
- Die *Equinox-Konsole* implementiert eine einfache, kommandozeilenorientierte Benutzungsschnittstelle zur Verwaltung des Equinox OSGi Frameworks.
- Die Equinox-Konsole kann um eigene Kommandos erweitert werden, indem eine Klasse vom Typ `org.eclipse.osgi.framework.console.CommandProvider` implementiert und diese unter dem `CommandProvider`-Interface an der OSGi Service Registry angemeldet wird.