midpoints
purify it

DOTS - reloaded

EntwicklerCamp 2014

René Winkelmeyer
midpoints GmbH

Über mich

René Winkelmeyer
Senior Consultant

IBM CHAMPION

midpoints GmbH
http://www.midpoints.de

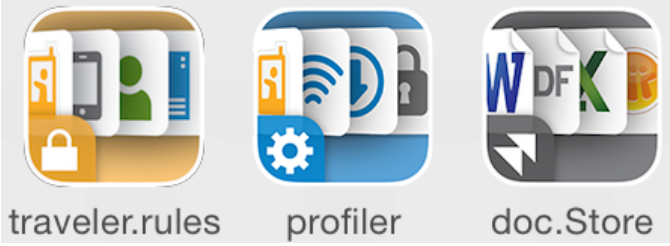
IBM Advanced Business Partner
IBM Design Partner

Services

- Notes / Domino Consulting
- E-Mail Management
- Mobile

Produkte

- IBM Notes Traveler Planung & Deployment
- mobile app development
- Domino basiertes iOS Device Management
- Domino basierte “Dropbox” für Notes and iOS



Available on the App Store

Über mich

René Winkelmeyer
Senior Consultant

IBM CHAMPION

- Skype
  muenzpraeger
- Twitter
  muenzpraeger
- LinkedIn
  muenzpraeger
- Slideshare
  muenzpraeger

- Web
  http://blog.winkelmeyer.com
  http://www.midpoints.de

- Mail
  mail@winkelmeyer.com
  rene.winkelmeyer@midpoints.de

OpenNTF
- File Navigator
- Generic NSF View Widget for IBM Connections

Agenda

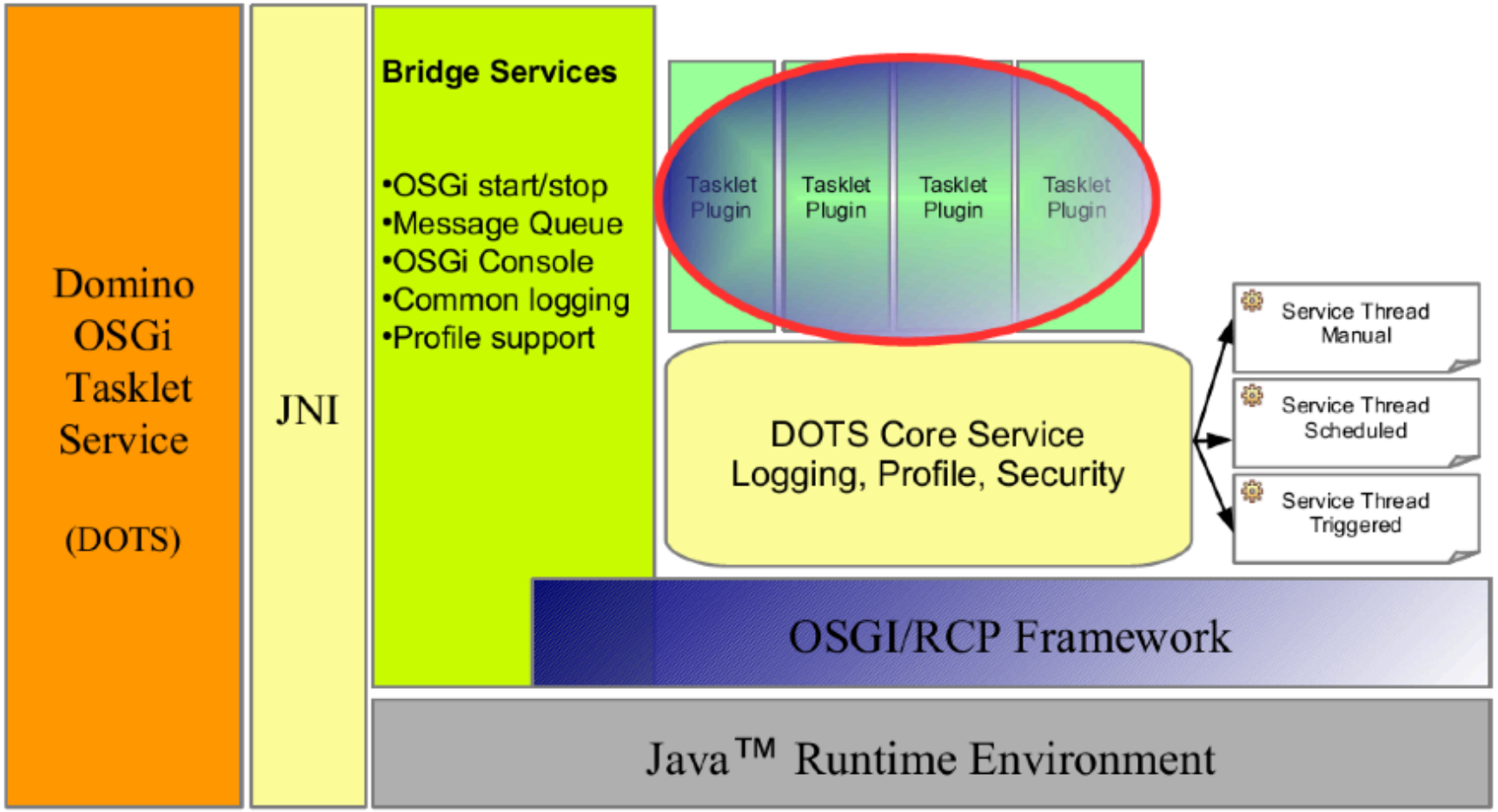
- DOTS – was, wie warum
- Startup und Task Extension Point
- Singleton-Nutzung
- Konfiguration über Annotationen
- Custom Console Commands
- Log4J2
- Links

Agenda

- DOTS – was, wie warum
- Startup und Task Extension Point
- Singleton-Nutzung
- Konfiguration über Annotationen
- Custom Console Commands
- Log4J2
- Links

DOTS

- Der DOTS-Task ist ein C-Task, welcher auf dem Domino-Server eine autarke OSGi-Runtime erzeugt.



DOTS

midpoints
purify it

- DOTS gibt es für 8.5.x Release auf OpenNTF
  - http://www.openntf.org/main.nsf/project.xsp?r=project/OSGI%20Tasklet%20Service%20for%20IBM%20Lotus%20Domino
- Seit Domino 9 Bestandteil des Server-Pakets

DOTS

midpoints
purify it

- Welche Vorteile bietet die Nutzung von DOTS?
  - Granulare Zeit- und Eventsteuerung
  - Server-side permissions
  - Keine Einschränungen à la AgentManager
  - Java, Java, Java (und SCM und Eclipse)

DOTS

- Welche Migrationsszenarien für Java-Agenten gibt es?
  - LotusScript/Formula => keine => neu schreiben!
  - Java => copy & paste
    - Geht – aber bitte in neuen Strukturen denken!

DOTS

- Wie kann ich DOTS-Plug-Ins deployen?
  - Über das Filesystem
    - <dominoprog>/osgi-dots/eclipse/applications/plugins
  - Über eine Updatesite.nsf
    - notes.ini-Parameter

Agenda

- DOTS – was, wie warum
- Startup und Task Extension Point
- Singleton-Nutzung
- Konfiguration über Annotationen
- Custom Console Commands
- Log4J2
- Links

Extension Points

- Über so genannte Extension Points können sich OSGi Plug-Ins an von anderen Plug-Ins bereitstellte Extensions „anhängen“.

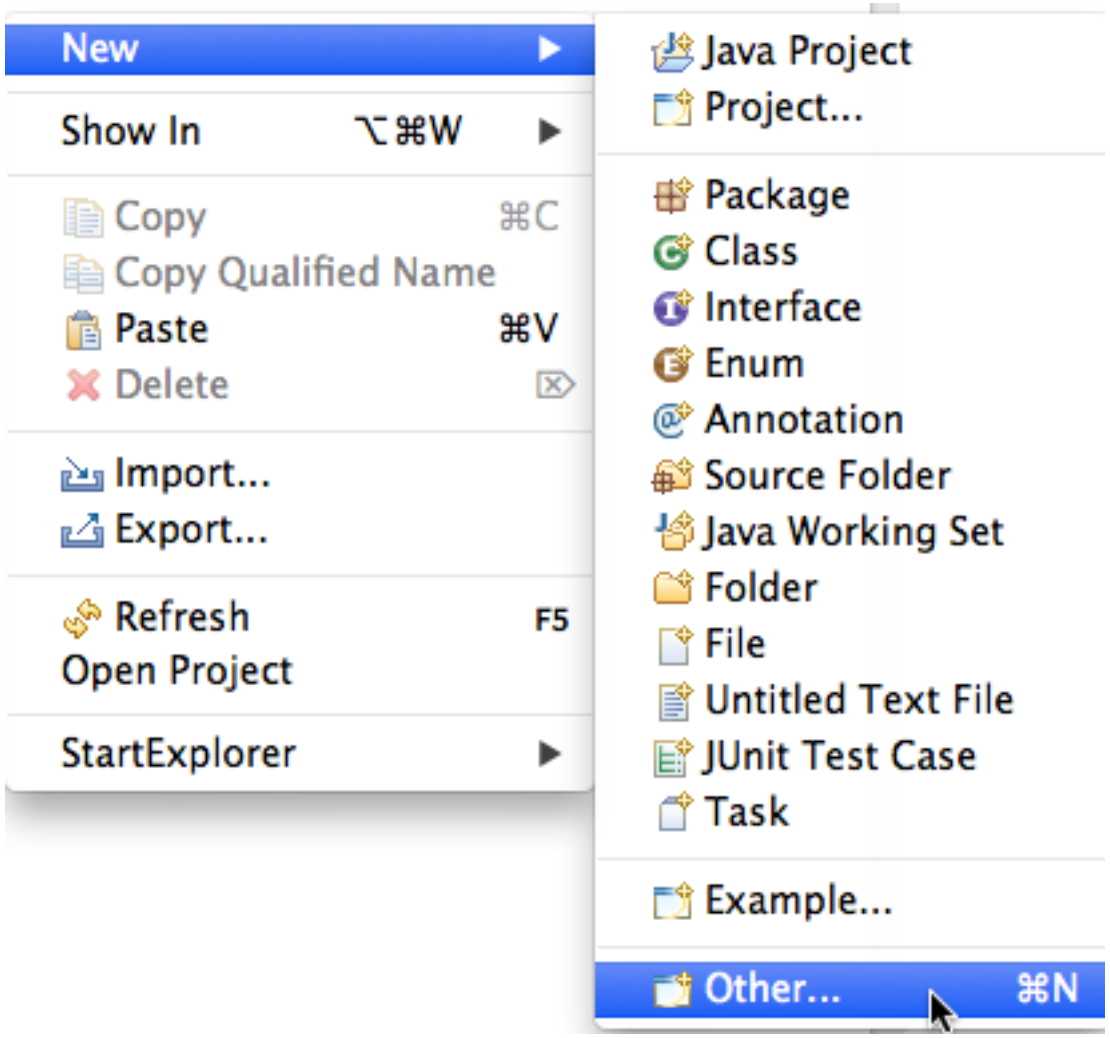
- Die meist benutzten Extension Points für DOTS sind
  - Startup
  - Task

Der Startup Extension Point

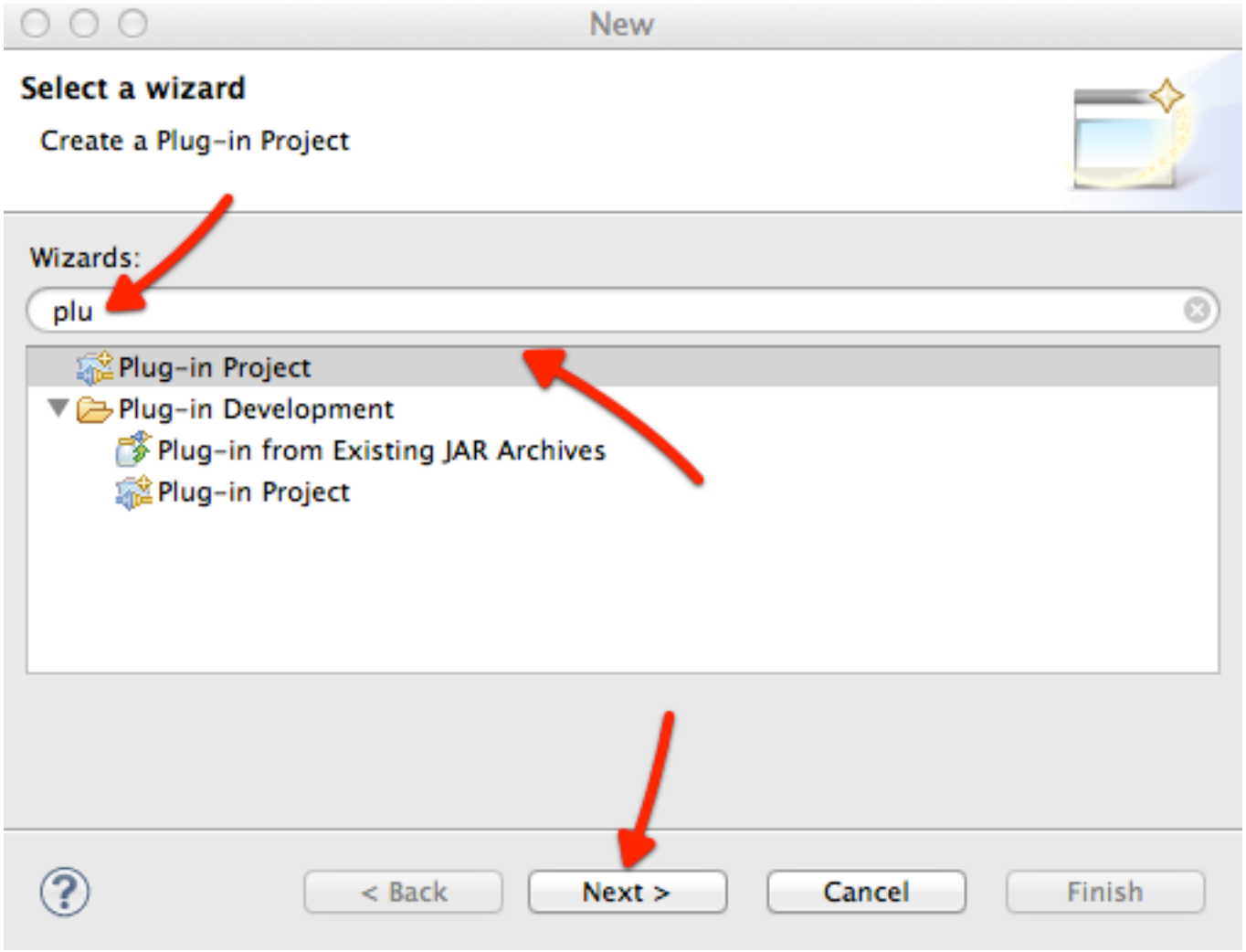
- Der Code, welcher über diesen Extension Point angesprochen wird, läuft noch vor Ausführung der Tasklets.

- Er eignet sich im Speziellen für die Initialisierung von globalen Werten, z. B. Log-Level o. ä.

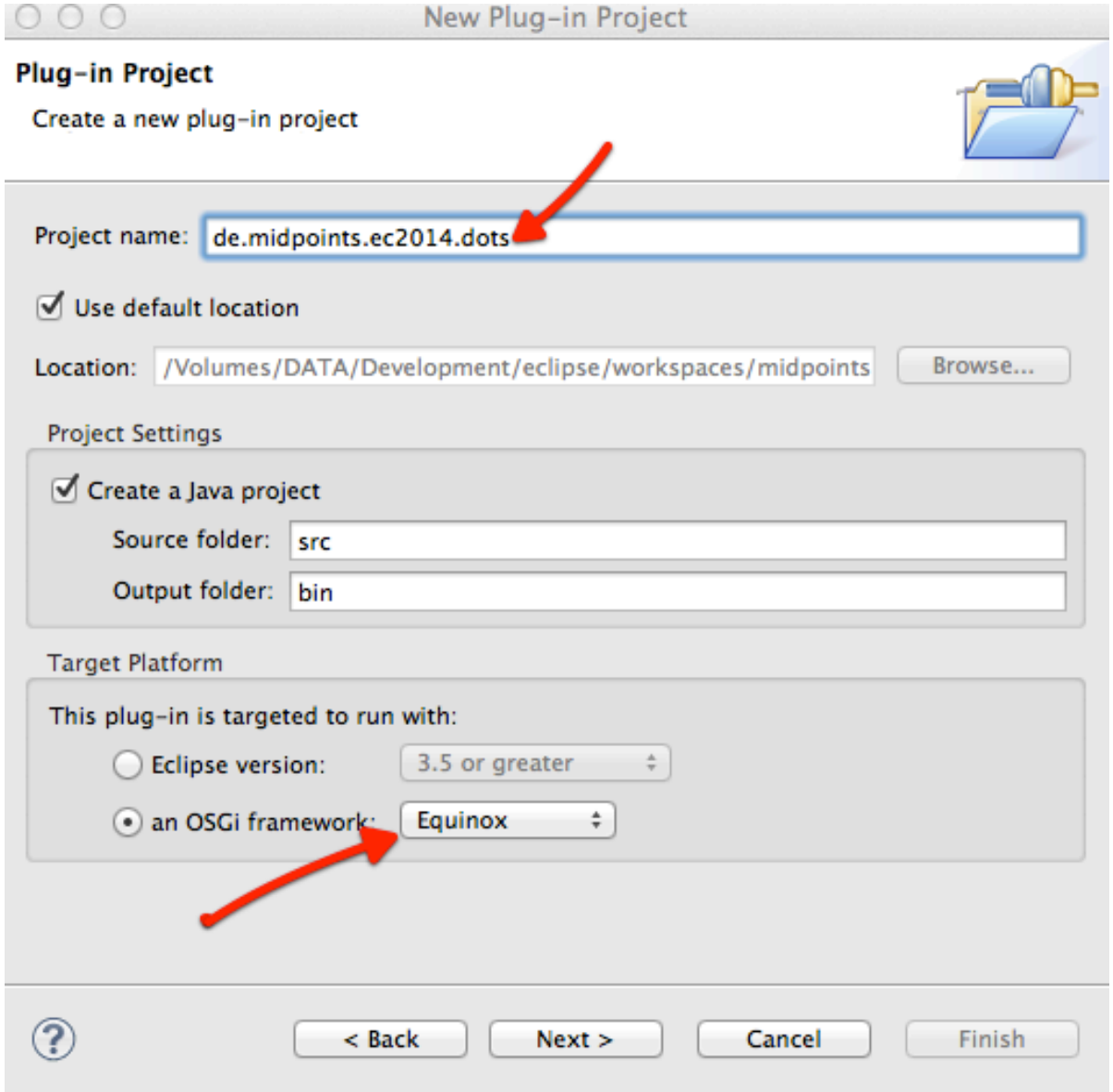
Erstellung des initialen Plug-In



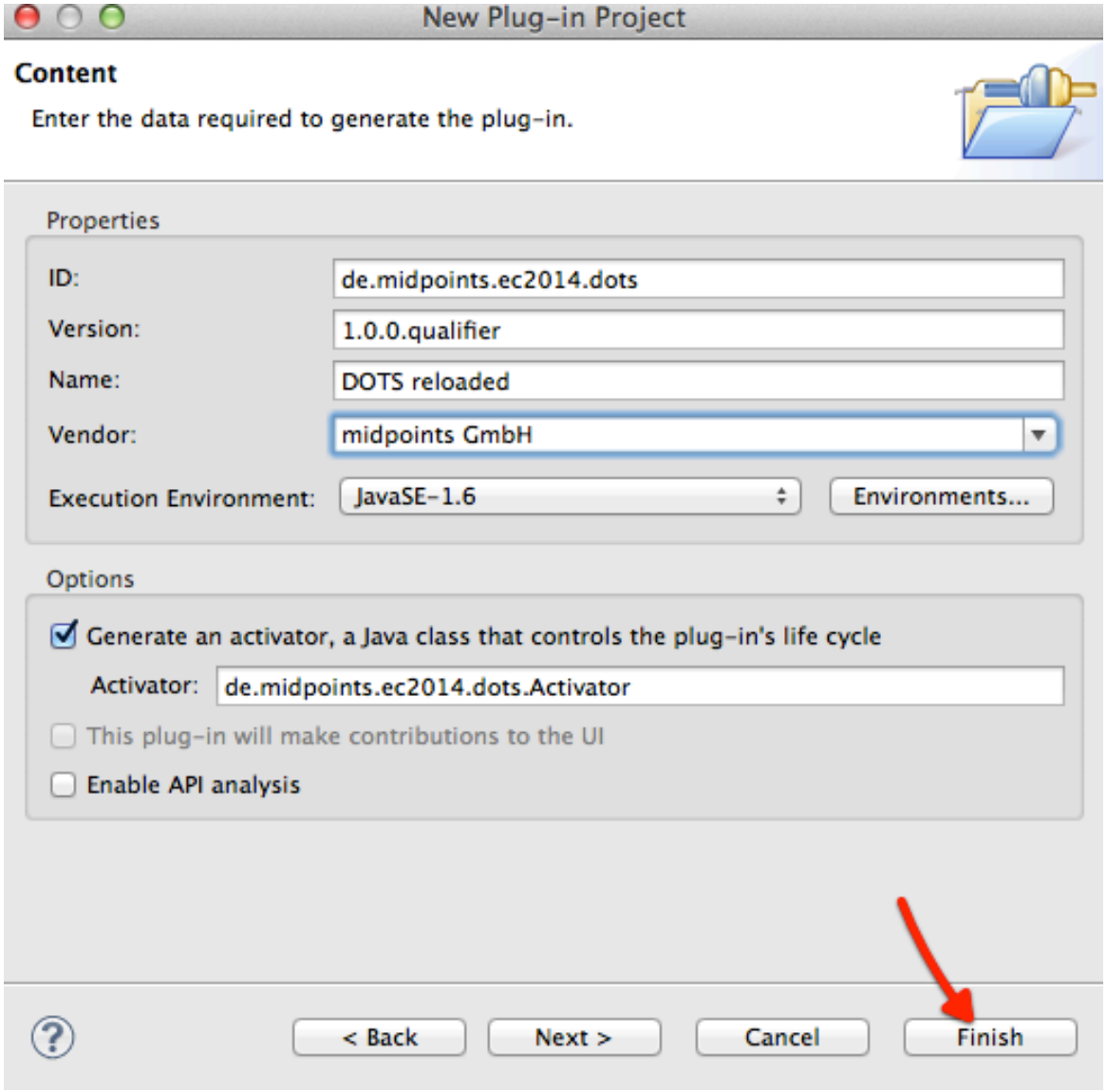
Erstellung des initialen Plug-In



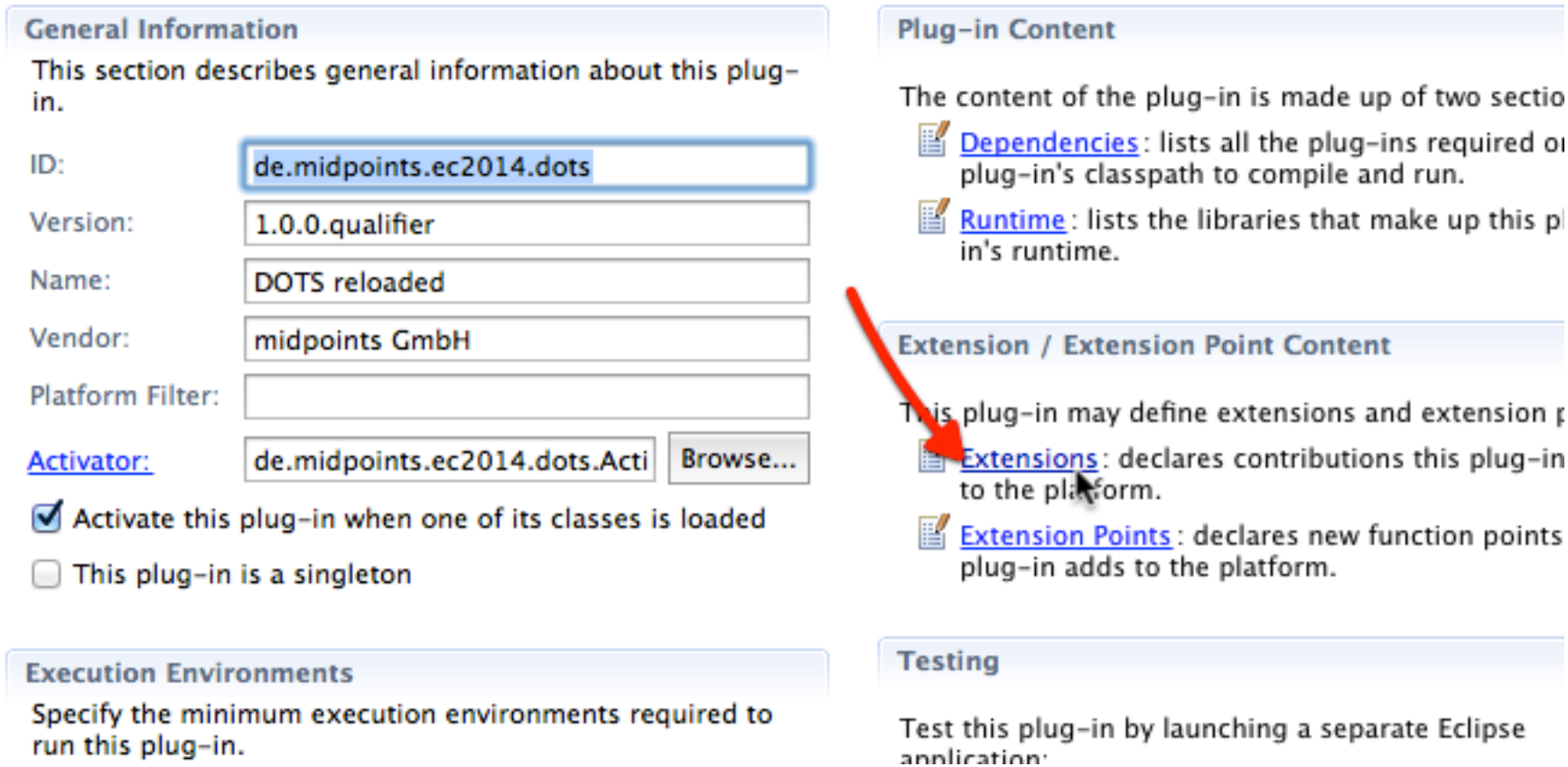
Erstellung des initialen Plug-In



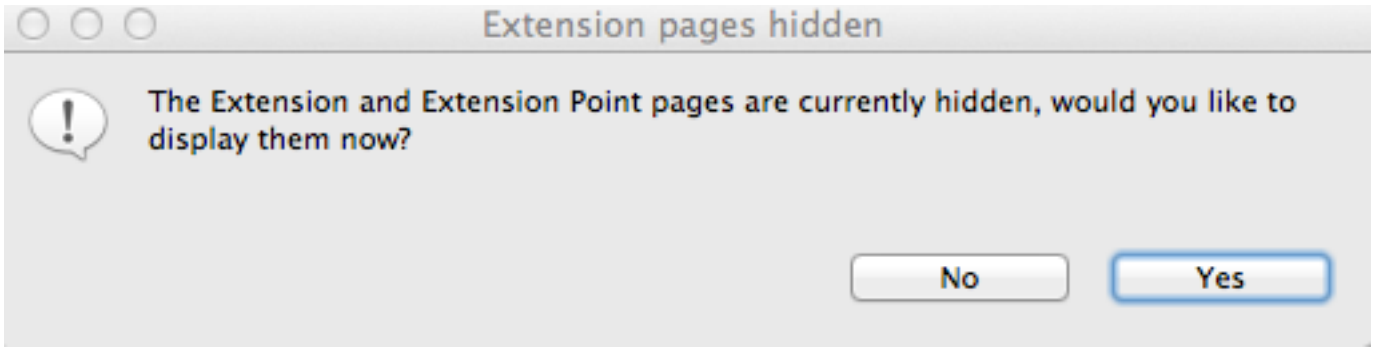
Erstellung des initialen Plug-In



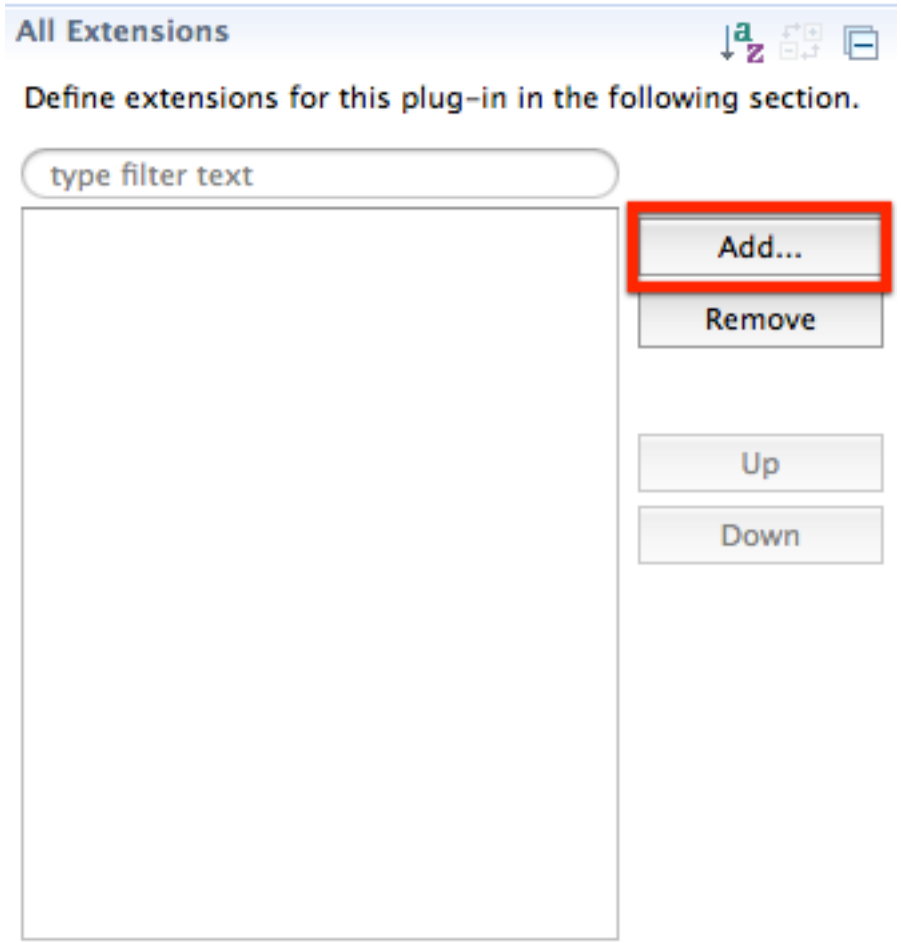
Einbindung des Startup Extension Point



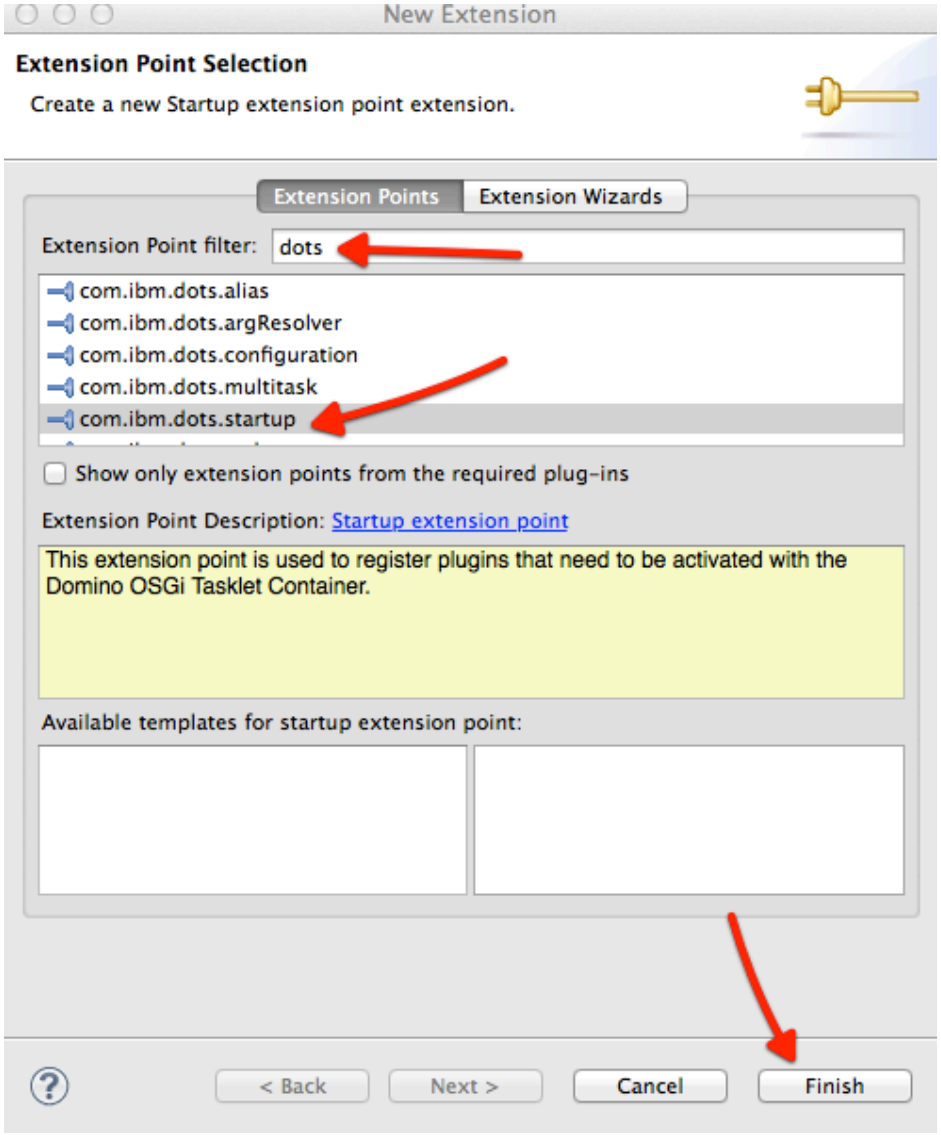
Einbindung des Startup Extension Point



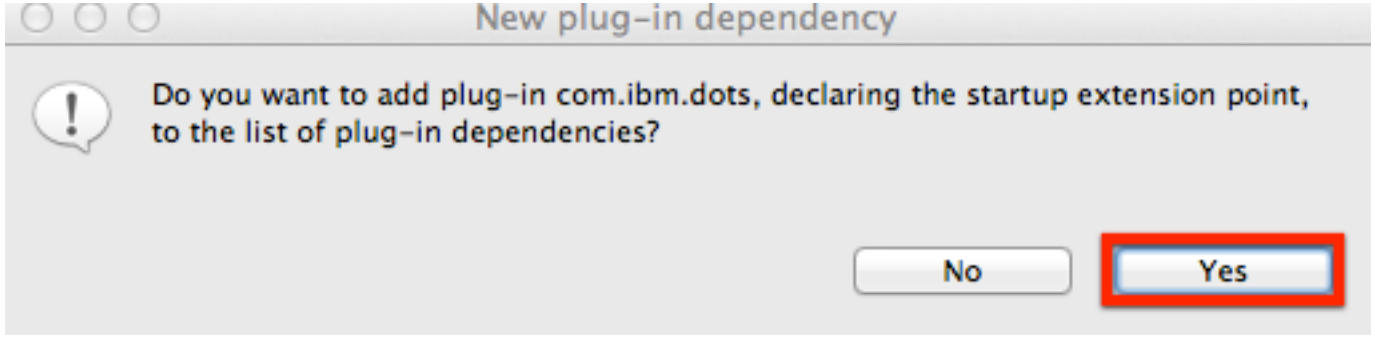
Einbindung des Startup Extension Point



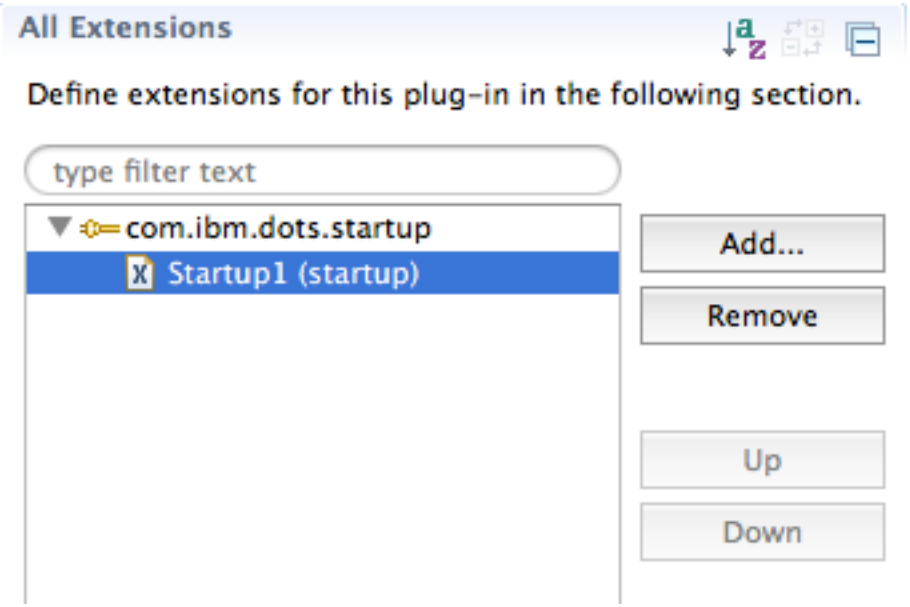
Einbindung des Startup Extension Point



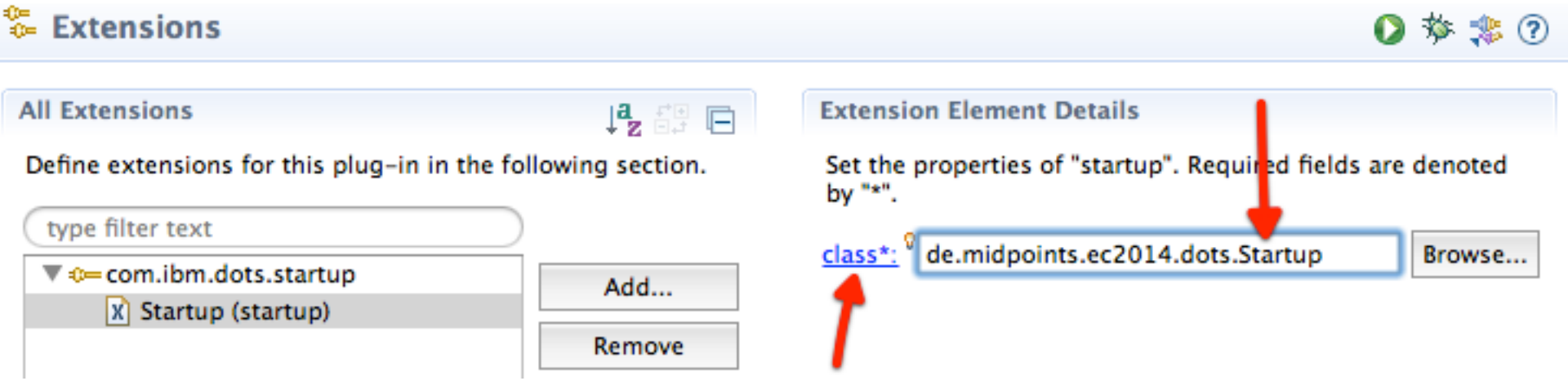
Einbindung des Startup Extension Point



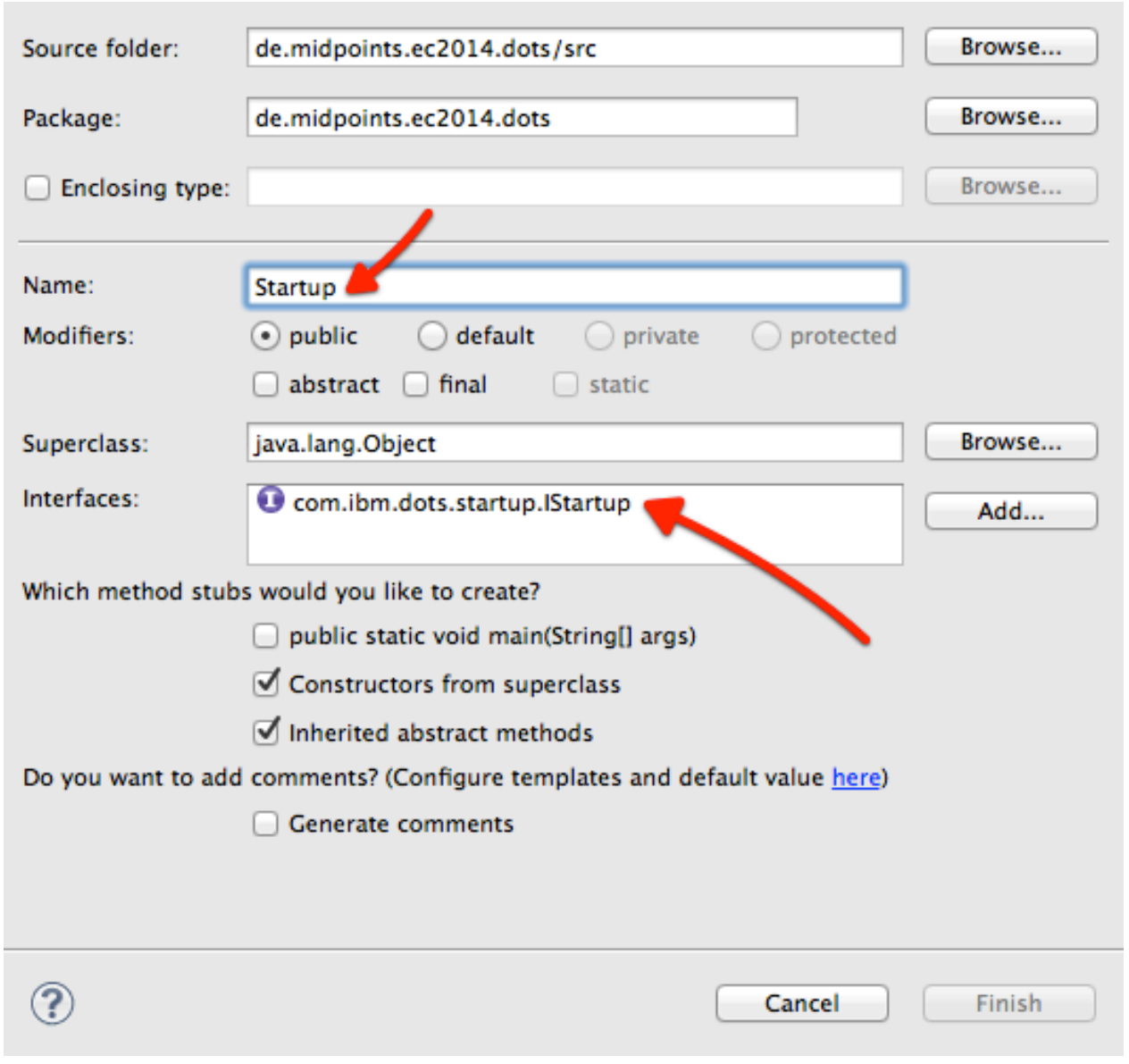
Einbindung des Startup Extension Point



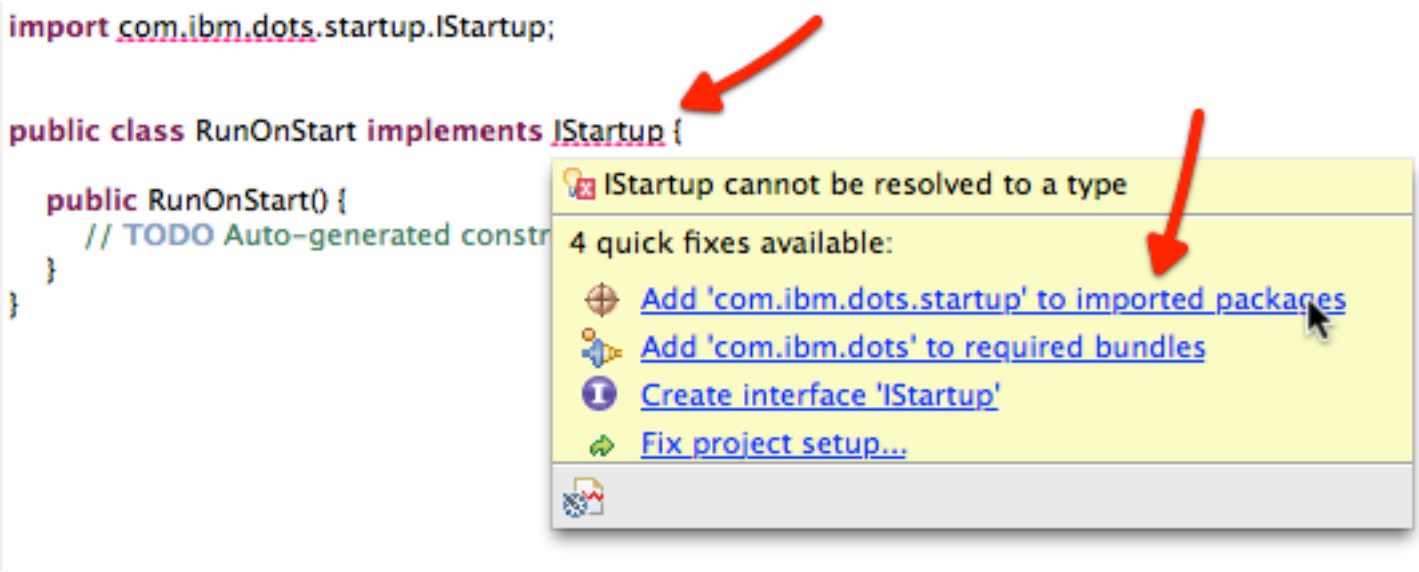
Einbindung des Startup Extension Point



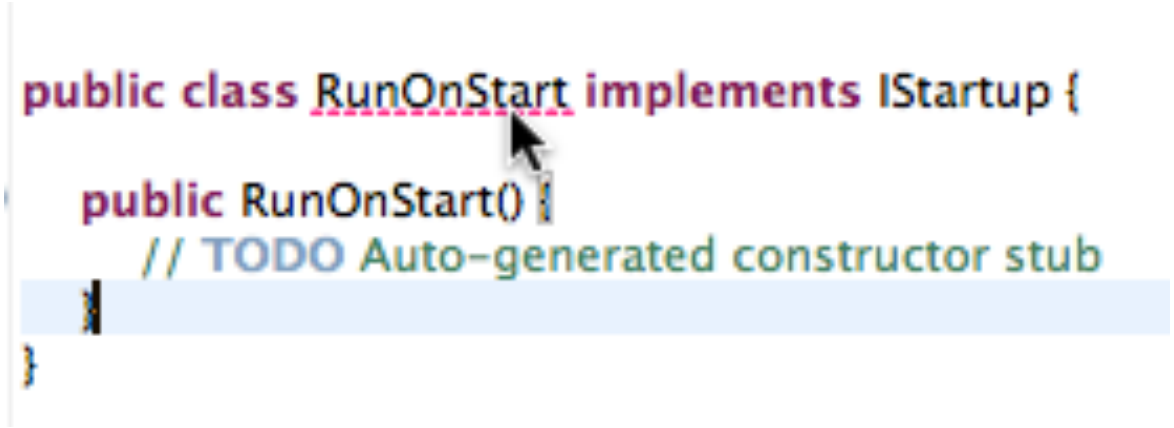
Einbindung des Startup Extension Point



Einbindung des Startup Extension Point



Einbindung des Startup Extension Point



Einbindung des Startup Extension Point

midpoints
purify it

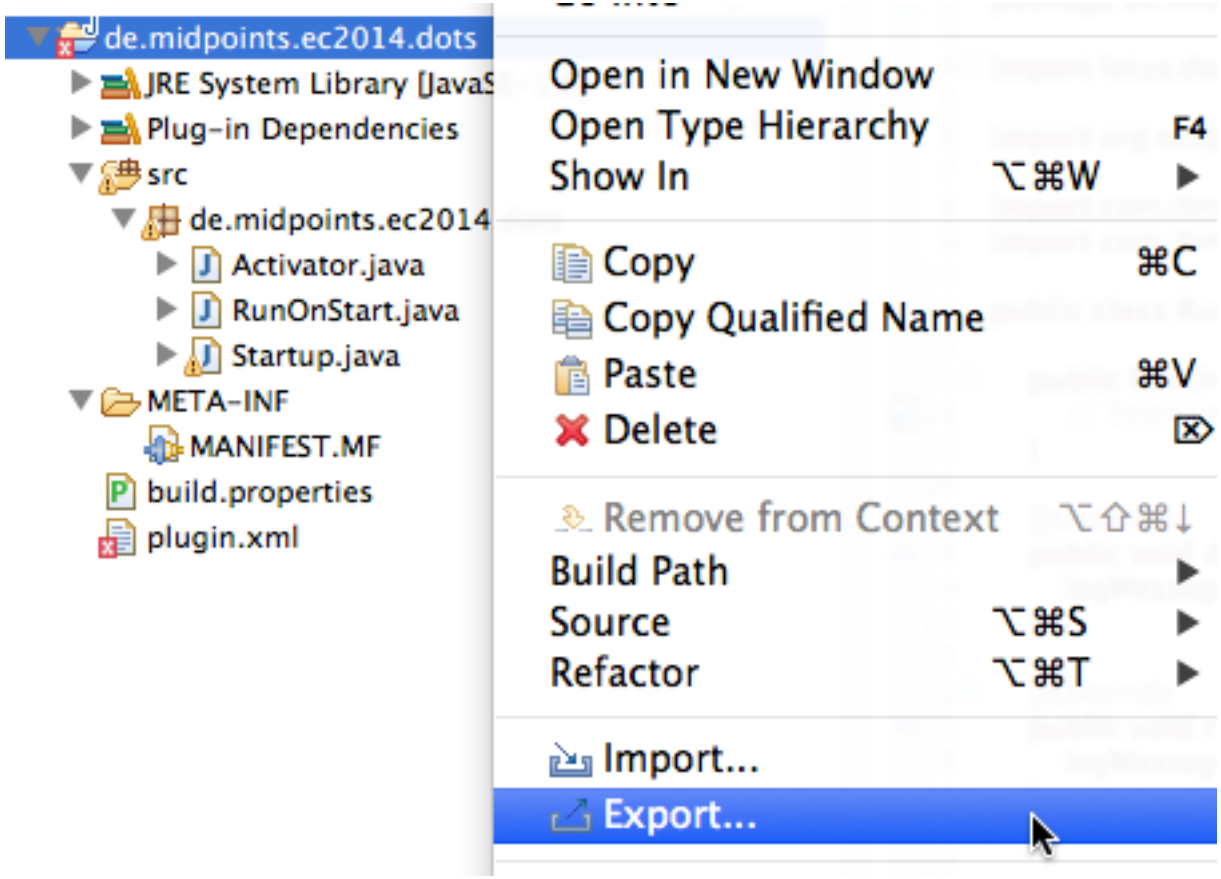
```
package de.midpoints.ec2014.dots;

import com.ibm.dots.startup.IStartup;

public class RunOnStart implements IStartup {

    public RunOnStart() {
        // TODO Auto-generated constructor stub
    }

    /**
     *
     */
    @Override
    public void earlyStartup() {
    }
}
```

Einbindung des Startup Extension Point

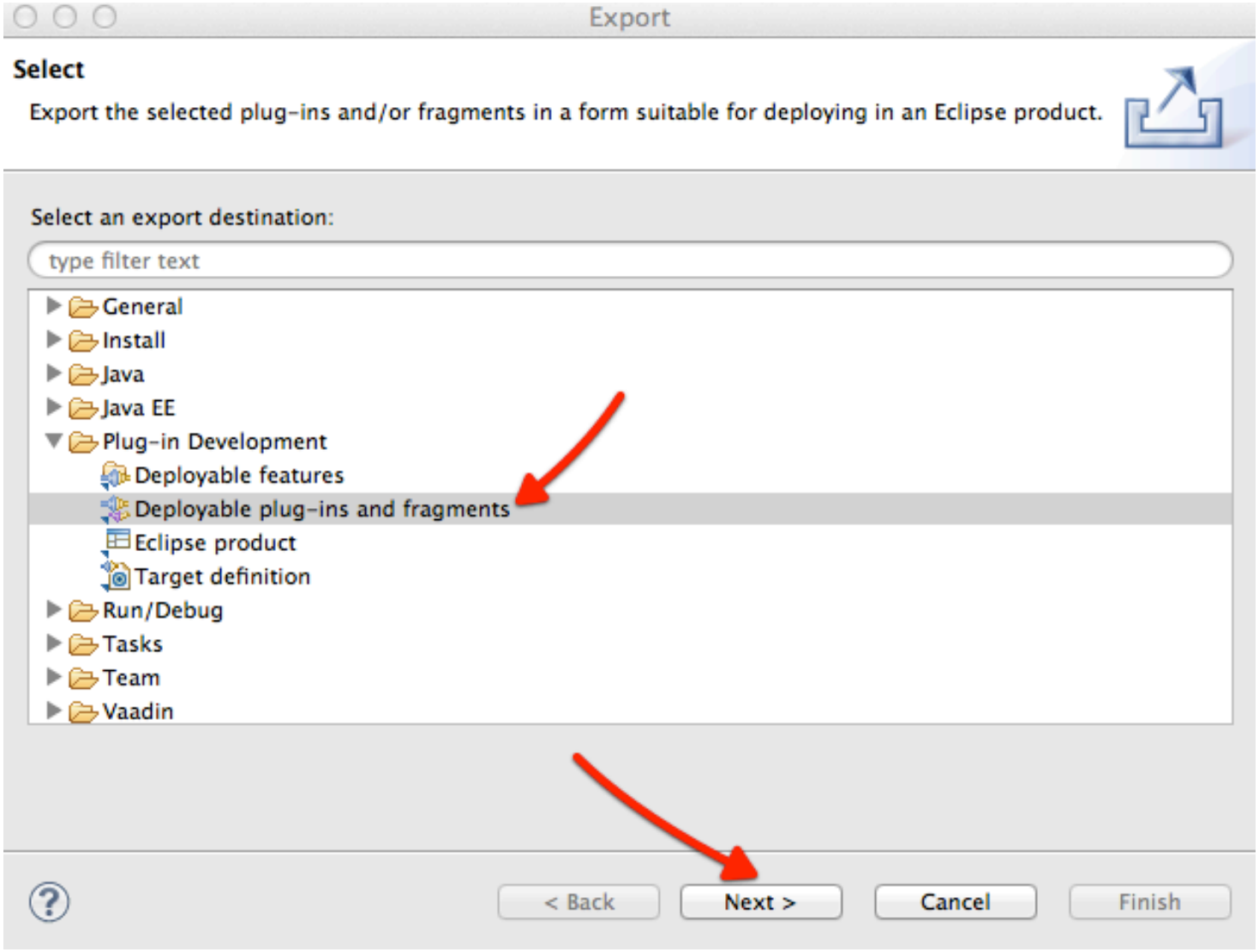
midpoints
purify it

```
package de.midpoints.ec2014.dots;

import com.ibm.dots.startup.IStartup;
import com.ibm.dots.task.ServerTaskManager;

public class RunOnStart implements IStartup {

    public RunOnStart() {
    }

    /**
     *
     */
    @Override
    public void earlyStartup() {
        ServerTaskManager.getInstance().logMessageText("Early Startup started");
    }
}
```

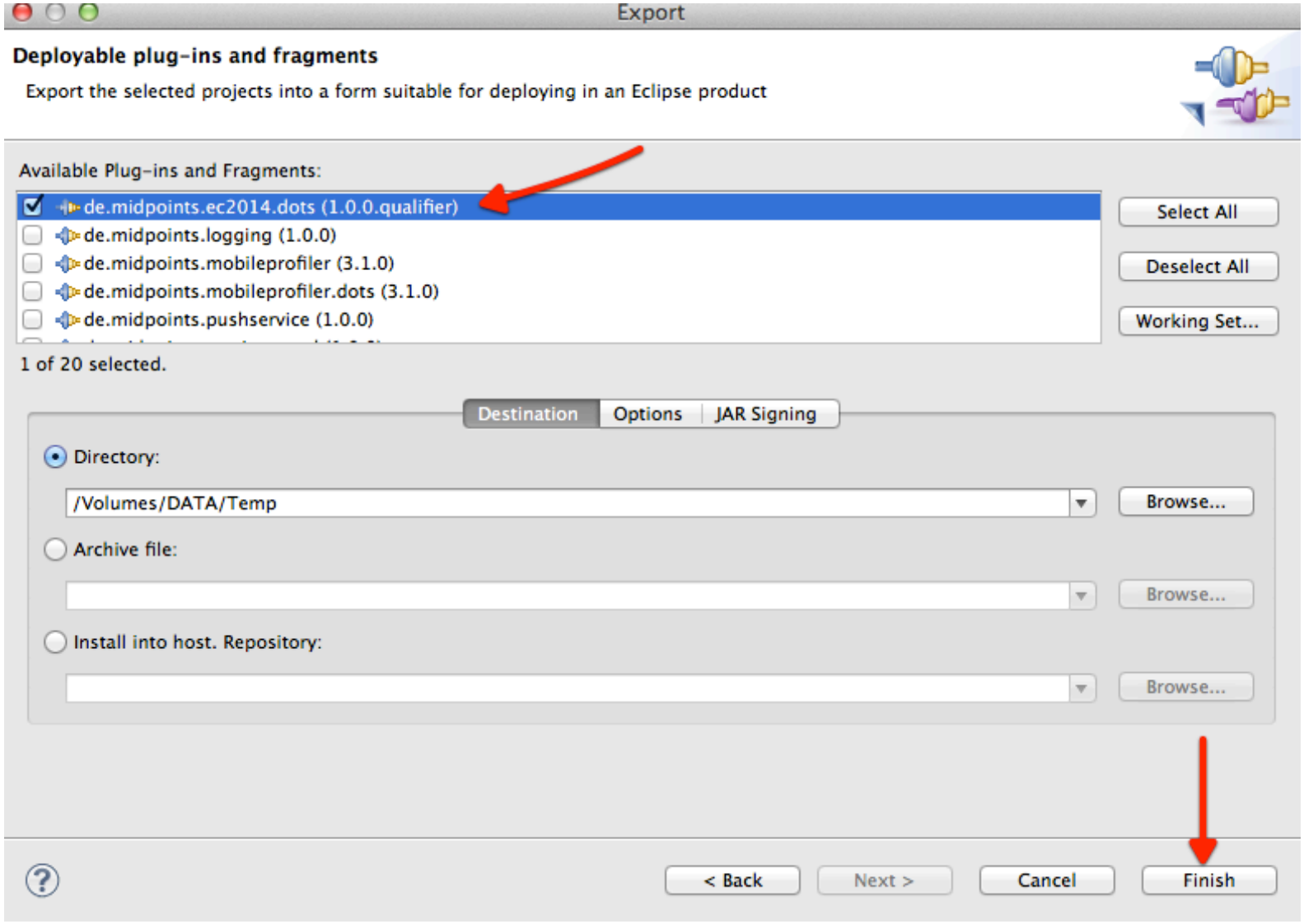
Erstes Test-Deployment



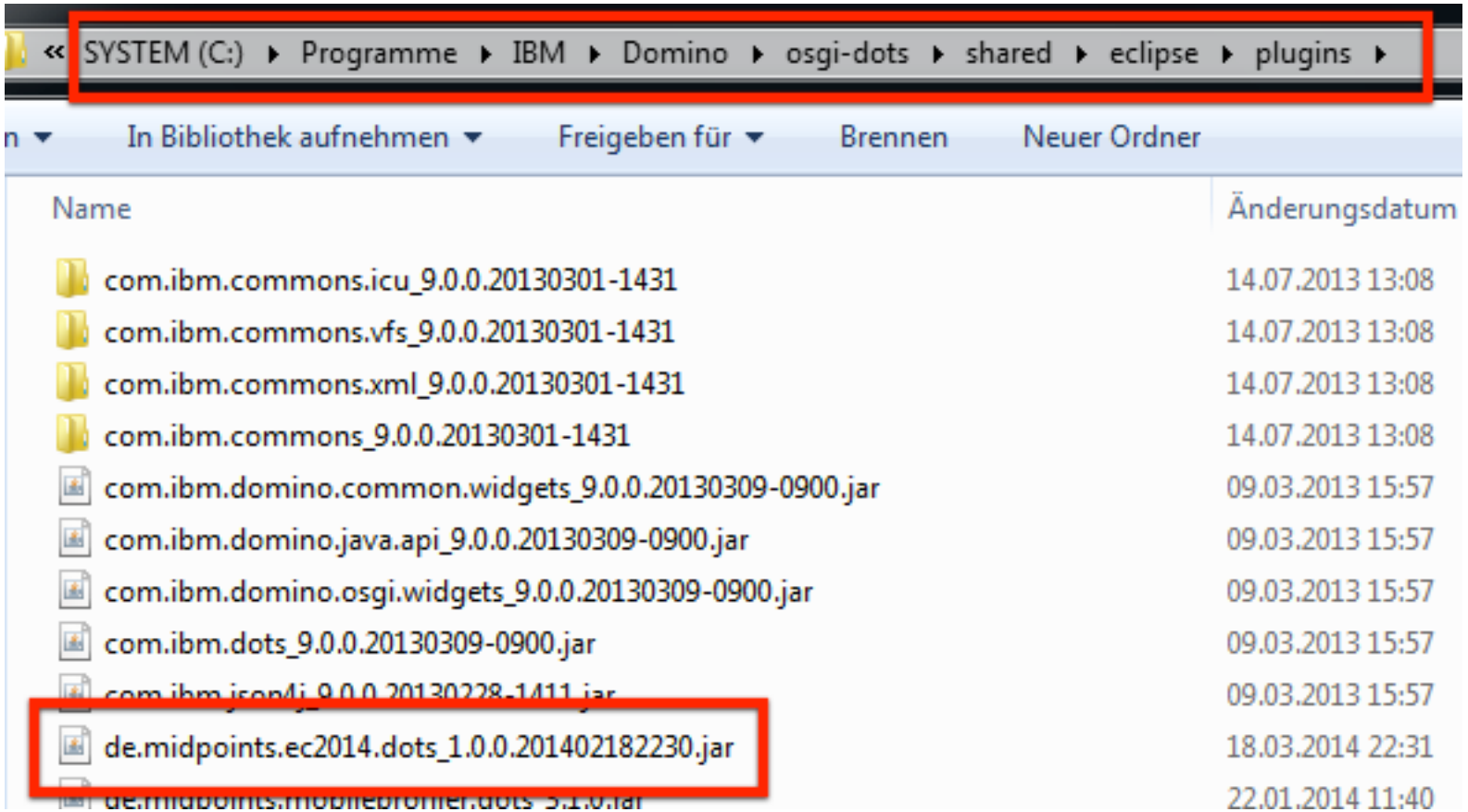
Erstes Test-Deployment



Erstes Test-Deployment

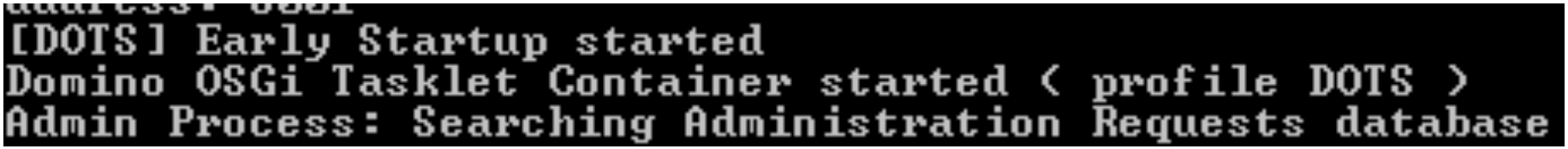


Erstes Test-Deployment

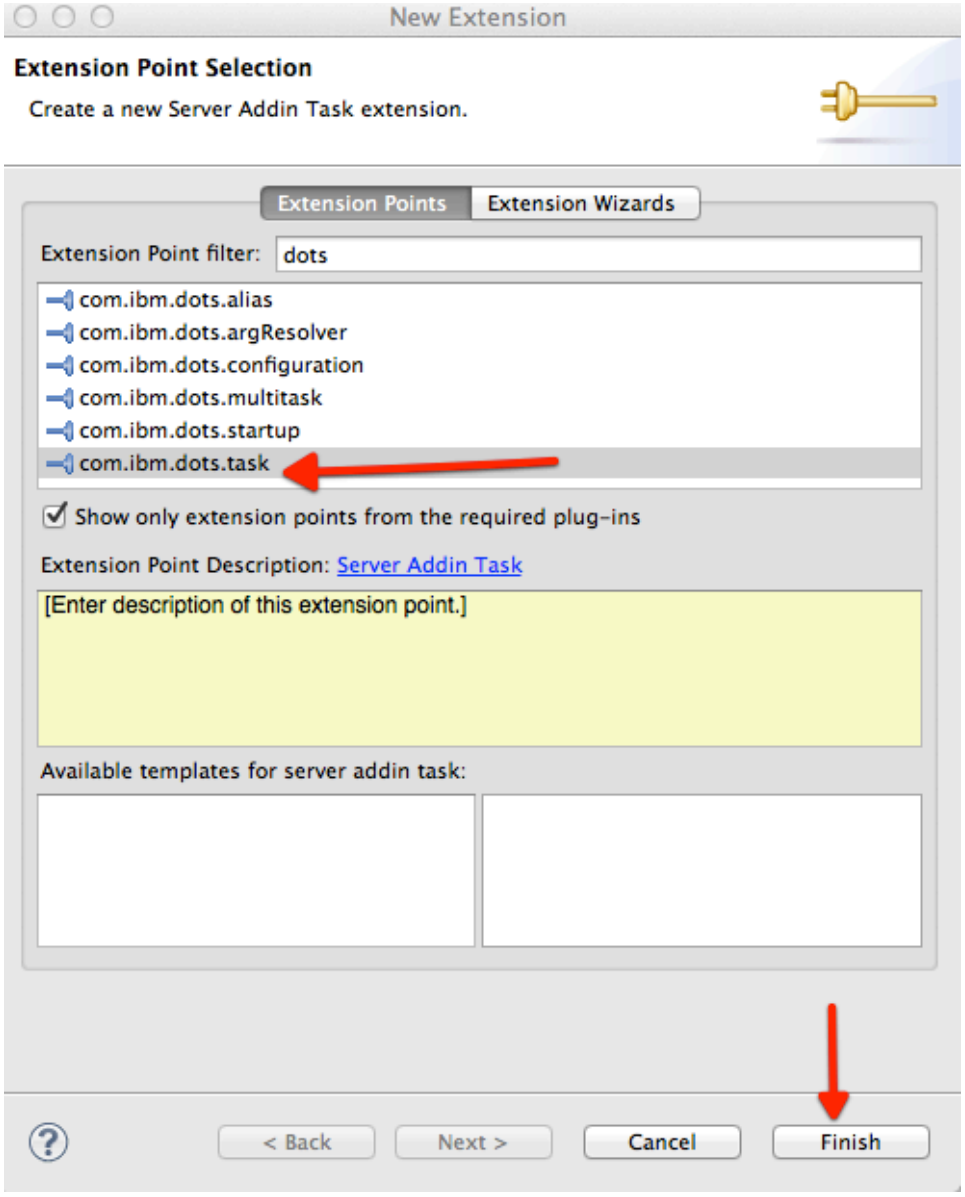


Erstes Test-Deployment

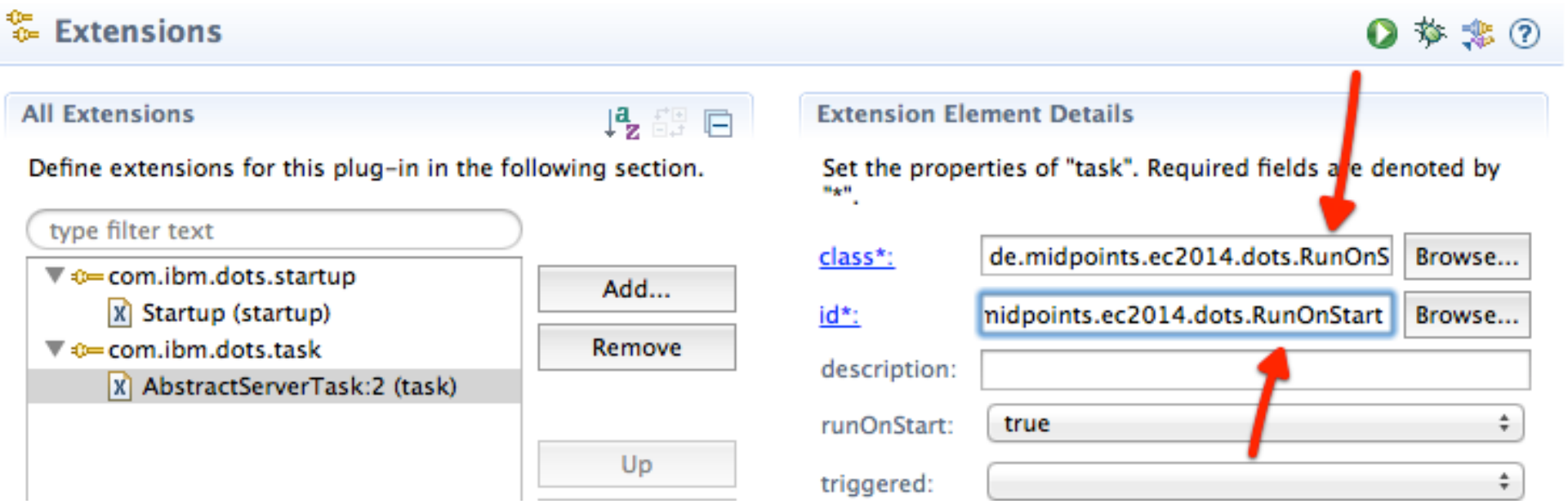
- DOTS-Task mit „load dots“ starten



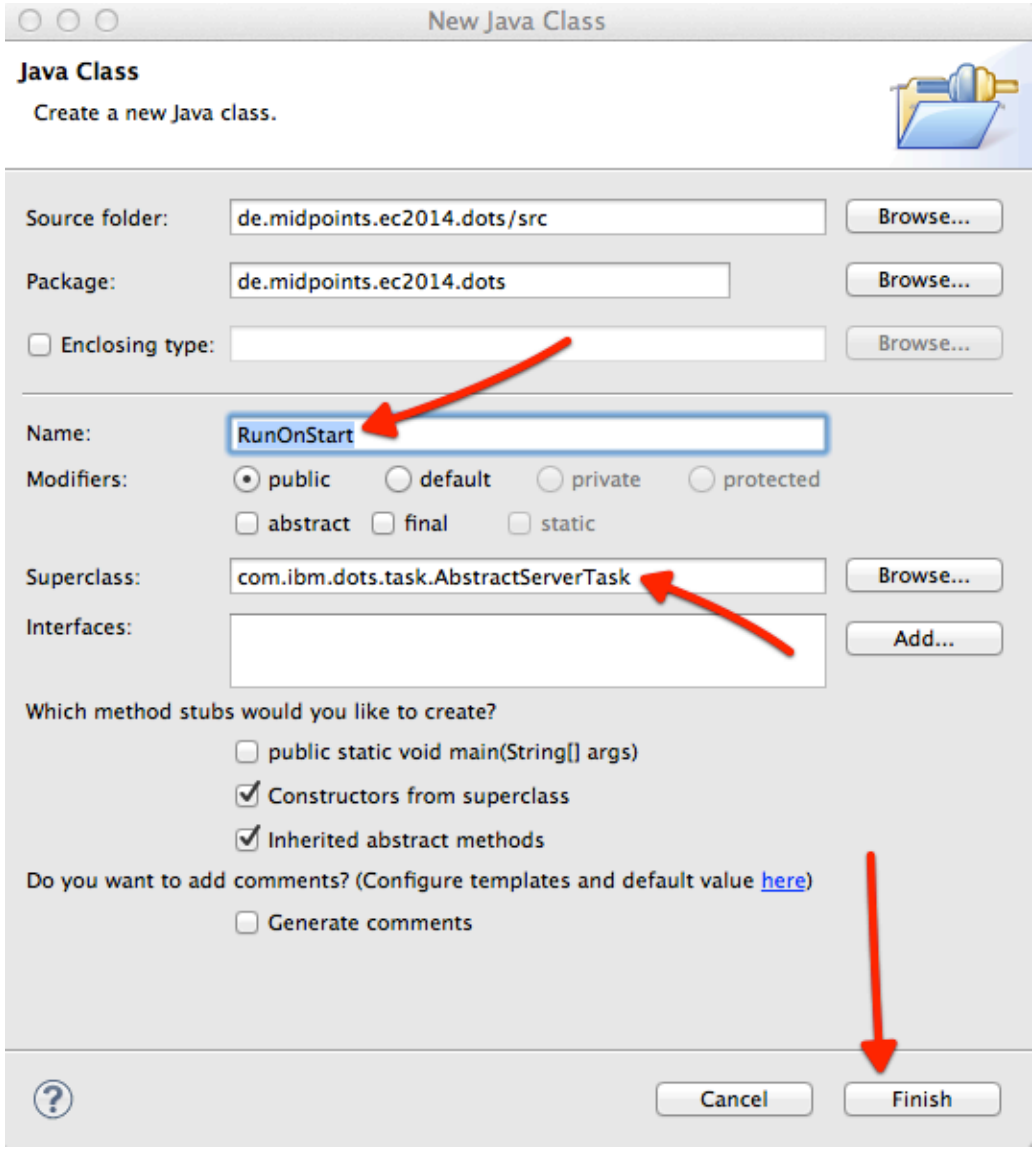
Einbindung des Task Extension Point



Einbindung des Task Extension Point



Einbindung des Task Extension Point



Einbindung des Task Extension Point

midpoints
purify it

```
public class RunOnStart extends AbstractServerTask {

    public RunOnStart() {
        // TODO Auto-generated constructor stub
    }

    @Override
    public void dispose() throws NotesException {
    }

    @Override
    public void run(RunWhen arg0, String[] arg1, IProgressMonitor arg2) throws NotesException {
    }
}
```

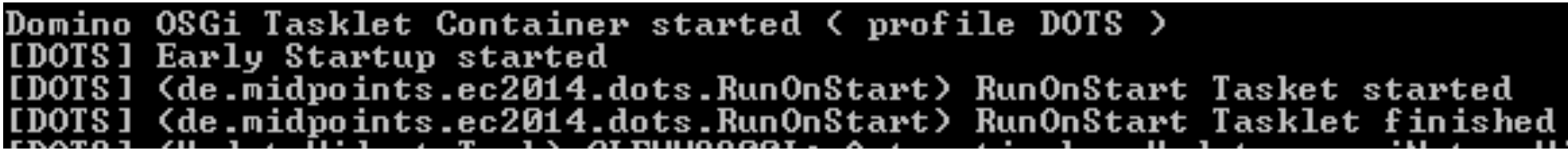
Einbindung des Task Extension Point

midpoints
purify it

```
public class RunOnStart extends AbstractServerTask {

    public RunOnStart() {
        // TODO Auto-generated constructor stub
    }

    @Override
    public void dispose() throws NotesException {
        logMessage("RunOnStart Tasklet finished");
    }

    @Override
    public void run(RunWhen arg0, String[] arg1, IProgressMonitor arg2) throws NotesException {
        logMessage("RunOnStart Tasket started");
    }
}
```

Test des Task Extension Point

- Plug-In erneut exportieren
- DOTS-Task mit „tell dots quit“ beenden
- Plug-In is osgi-dots Verzeichnis kopieren
- DOTS-Task mit „l dots“ starten



Anatomie des Task Extension Point (XML)

- RunOnStart – startet einmalig mit dem DOTS-Task

```
<extension
      point="com.ibm.dots.task">
   <task
         class="de.midpoints.ec2014.dots.RunOnStart"
         id="de.midpoints.ec2014.dots.RunOnStart"
         runOnStart="true">
   </task>
</extension>
```

Anatomie des Task Extension Point (XML)

- Scheduled Task mit definierten Zeiteinheiten

```
<task
    class="de.midpoints.ec2014.dots.ScheduledTask"
    description="Scheduled Task"
    filter="de.midpoints.ec2014.dots.ScheduledTaskFilter"
    id="de.midpoints.ec2014.dots.scheduled">
  <run every="5" unit="minute"/>
  <run every="1" unit="day"/>
</task>
```

Anatomie des Task Extension Point (XML)

- Manueller Task

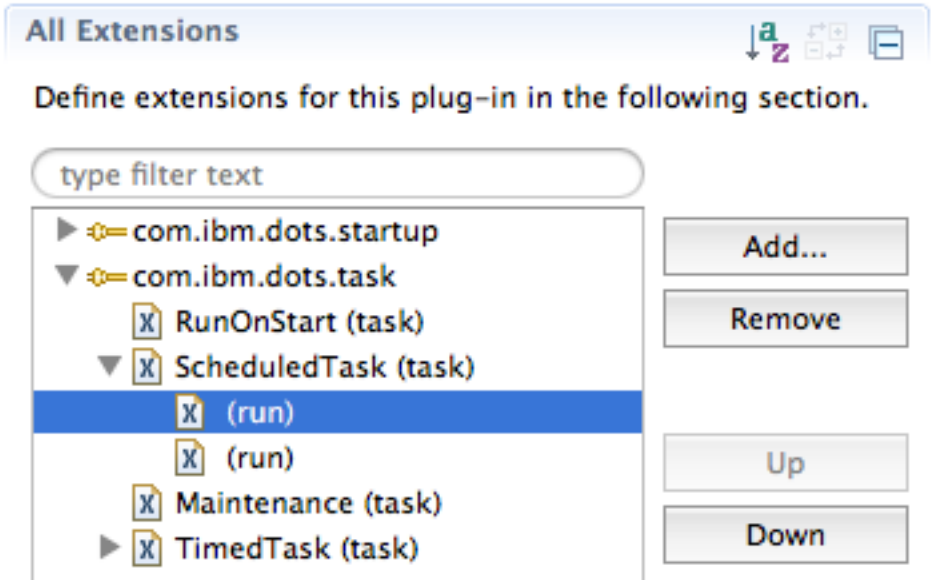
```
<task id="de.midpoints.ec2014.dots.Maintenance"
 description="Runs Maintenance actions"
 class="de.midpoints.ec2014.dots.Maintenance"
 runOnStart="false">
</task>
```

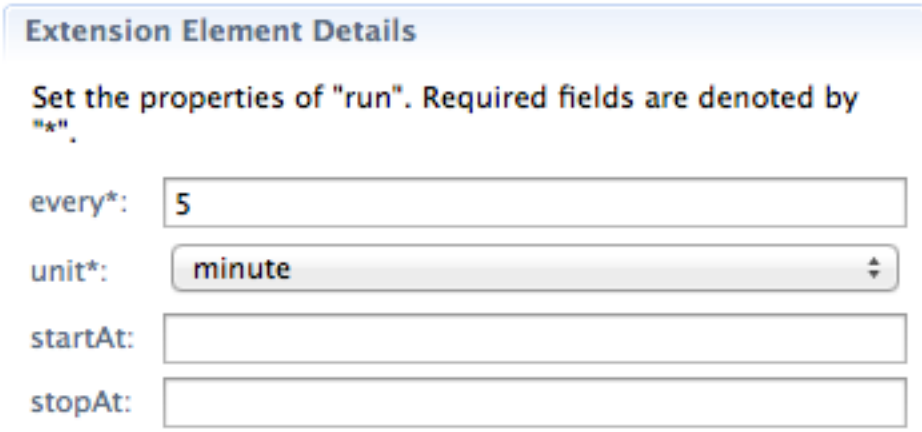
Anatomie des Task Extension Point (XML)

- Periodischer Task mit definiertem Zeitraum

```
<task
    class="de.midpoints.ec2014.dots.TimedTask"
    description="Scheduled between specific times of the day"
    id="de.midpoints.ec2014.dots.timedTask"
    runOnStart="false"
    triggered="false">
  <run
      every="2"
      startAt="2:00 PM"
      stopAt="3:00 PM"
      unit="minute">
  </run>
</task>
```

Anatomie des Task Extension Point (GUI)





Wissenswerte DOTS-Commands

- *tell dots tasklist*
  - Übersicht der Tasklets
- *tell dots taskstatus*
  - Aktueller Run-Status der Tasklets
- *tell dots ss <name>*
  - Filtert die Tasklets auf Basis des <name> Parameters

Wissenswerte DOTS-Commands

- *tell dots diag <plugin-id>*
  - Überprüft die vorhandenen Plug-Ins auf (nicht-erfüllte) Dependencies
- *tell dots run <task-id>*
  - Führt das per <task-id> angegebene Tasklet aus (analog ‚tell amr run...)
- *tell dots cancel <task-id>*
  - Unterbricht den aktuellen Lauf des per <task-id> übergebenen Tasklet

Agenda

- DOTS – was, wie warum
- Startup und Task Extension Point
- Singleton-Nutzung
- Konfiguration über Annotationen
- Custom Console Commands
- Log4J2
- Links

Singleton

- Ein Singleton ist eine Art „globaler Container“. Er wird einmalig initialisiert und steht persistent zur Verfügung.

```
public class StatusManager {

    private static final StatusManager   instance   = new StatusManager();

    public static StatusManager getInstance() {
        return instance;
    }
}
```

Singleton

midpoints
purify it

- Wir nutzen einen Singleton um intern den Status der Tasklets zu tracken.

```
public class StatusManager {

   private static final StatusManager                    instance    = new StatusManager();
   private final ConcurrentHashMap<String, IProgressMonitor>  hmMonitors  = new ConcurrentHashMap<String, IProgressMonitor>();

   private StatusManager() {
   }

   public static StatusManager getInstance() {
      return instance;
   }

   public void addProgressMonitor(String taskletName, IProgressMonitor monitor) {
      if (!hmMonitors.containsKey(monitor)) {
         hmMonitors.put(taskletName, monitor);
      }
   }

   public boolean isMonitorRunning(String taskletName) {
      return hmMonitors.containsKey(taskletName);
   }

   public void removeProgressMonitor(String taskletName) {
      if (hmMonitors.containsKey(taskletName)) {
         hmMonitors.remove(taskletName);
      }
   }
}
```

Singleton

- Beim Start und beim Beendes eines Tasklets setzen wir den entsprechenden Status im Singleton.

```
public class RunOnStart extends AbstractServerTask {

    public RunOnStart() {
    }

    @Override
    public void dispose() throws NotesException {
        StatusManager.getInstance().removeProgressMonitor("RunOnStart");
        logMessage("RunOnStart Tasklet finished");
    }

    @Override
    public void run(RunWhen runWhen, String[] arguments, IProgressMonitor monitor) throws NotesException {
        logMessage("RunOnStart Tasket started");
        StatusManager.getInstance().addProgressMonitor("RunOnStart", monitor);
        // do stuff
    }
}
```

Singleton – weitere Use-Cases

- Vorhalten Tasklet-übergreifender Konfigurationsinformationen
  - Server und Pfad von Datenbanken
  - Credentials
  - ...
- Nutzung Tasklet-übergreifender Funktionen
  - Logging

Agenda

- DOTS – was, wie warum
- Startup und Task Extension Point
- Singleton-Nutzung
- Konfiguration über Annotationen
- Custom Console Commands
- Log4J2

Java-Annotationen allgemein

midpoints
purify it

- Über Annotationen kann Quellcode mit Metadaten versehen werden.
  - Eingeführt mit Java 5.
- Bekannte Vertreteter sind z. B. @Deprecated, @Override, @SurpressWarnings

Java-Annotationen allgemein

midpoints
purify it

- Es ist die Definition eigener Annotationen möglich. Diese werden zur Compiletime oder zur Runtime ausgewertet.
  - Compiletime => z. B. Anzeige von Warnungen
  - Runtime => Interpretation von Konfigurationen

http://de.wikipedia.org/wiki/Annotation_(Java)

DOTS-Konfiguration mit Annotationen

midpoints
purify it

- Annontationen ersetzen (teils) die Konfiguration über die plugin.xml.
- Developer sollen damit die Möglichkeit haben die Konfiguration „besser“ vornehmen zu können.
- In der plugin.xml ist nur noch eine Basis-Konfiguration zu hinterlegen.

DOTS-Konfiguration mit Annotationen

midpoints
purify it

- Mögliche Annotationen sind:
  - @Run
  - @RunEvery
  - @RunOnStart
  - @Triggered
  - @HungPossibleAfter

DOTS-Konfiguration mit Annotationen

```
<extension
      point="com.ibm.dots.task">
   <task
         class="de.midpoints.ec2014.dots.Annotations"
         id="de.midpoints.ec2014.dots.Annotations">
   </task>
</extension>
```

DOTS-Konfiguration mit Annotationen

midpoints
purify it

```
@RunOnStart
public void runOnStart(IProgressMonitor monitor) {
    logMessage("Annotated onStart method");
}
```

DOTS-Konfiguration mit Annotationen

midpoints
purify it

```
@Run(id = "manual")
public void runManual(String[] args, IProgressMonitor monitor) {
    logMessage("Annotated run method with id=manual");
}
```

DOTS-Konfiguration mit Annotationen

midpoints
purify it

```
@RunOnStart
@RunEvery(every = 5, unit = RunUnit.second)
@HungPossibleAfter(timeInMinutes = 1)
public void runEvery60seconds(IProgressMonitor monitor) {
    logMessage("Called from annotated method every 60 seconds");
}
```

Agenda

- DOTS – was, wie warum
- Startup und Task Extension Point
- Singleton-Nutzung
- Konfiguration über Annotationen
- Custom Console Commands
- Log4J2
- Links

Custom Console Commands

midpoints
purify it

- Durch die Nutzung von Standard-Eclipse Komponenten kann der DOTS-Task um eigene OSGi Consolen Commands erweitert werden.
  - Dies ist besonders nützlich, wenn man definierte Funktionen ohne direkte Nutzung der Tasklets bereitstellen möchte.

- Hierzu bedienen wir und der *CommandProvider*-Klasse aus dem Paket *org.eclipse.osgi.framework.console*.

Custom Console Commands

- Die Commands werden dabei in einer eigenen Klasse implementiert. Dies ist dann im Activator zu registrieren.

Commands-Klasse erstellen

midpoints
purify it

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

public class Commands implements CommandProvider {
```

Commands-Klasse erstellen

midpoints
purify it

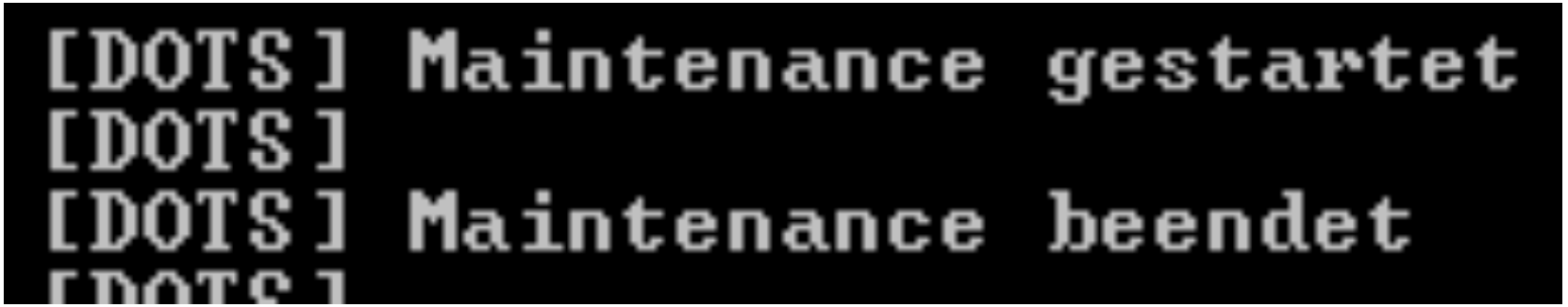
```
public void _ec2014(CommandInterpreter ci) {
    String command = ci.nextArgument();
    if (command == null) {
        ci.print("kein Parameter wurde erfasst");
        ci.println(getHelp());
    } else if (command.equals("maintenance")) {
        ci.print("Maintenance gestartet");
        // do maintenance here
        ci.print("Maintenance beendet");
    } else {
        ci.print("ein unbekannter Parameter wurde erfasst");
        ci.println(getHelp());
    }
}

/**
 * @return
 */
@Override
public String getHelp() {
    return "dies ist die Hilfe\nkann auch mehrzeilig sein";
}
```

Commands-Klasse im Activator registrieren

midpoints
purify it

```
@Override
public void start(BundleContext bundleContext) throws Exception {
    Activator.context = bundleContext;
    context.registerService(CommandProvider.class.getName(), new Commands(), null);
}
```

Custom Console Commands



Agenda

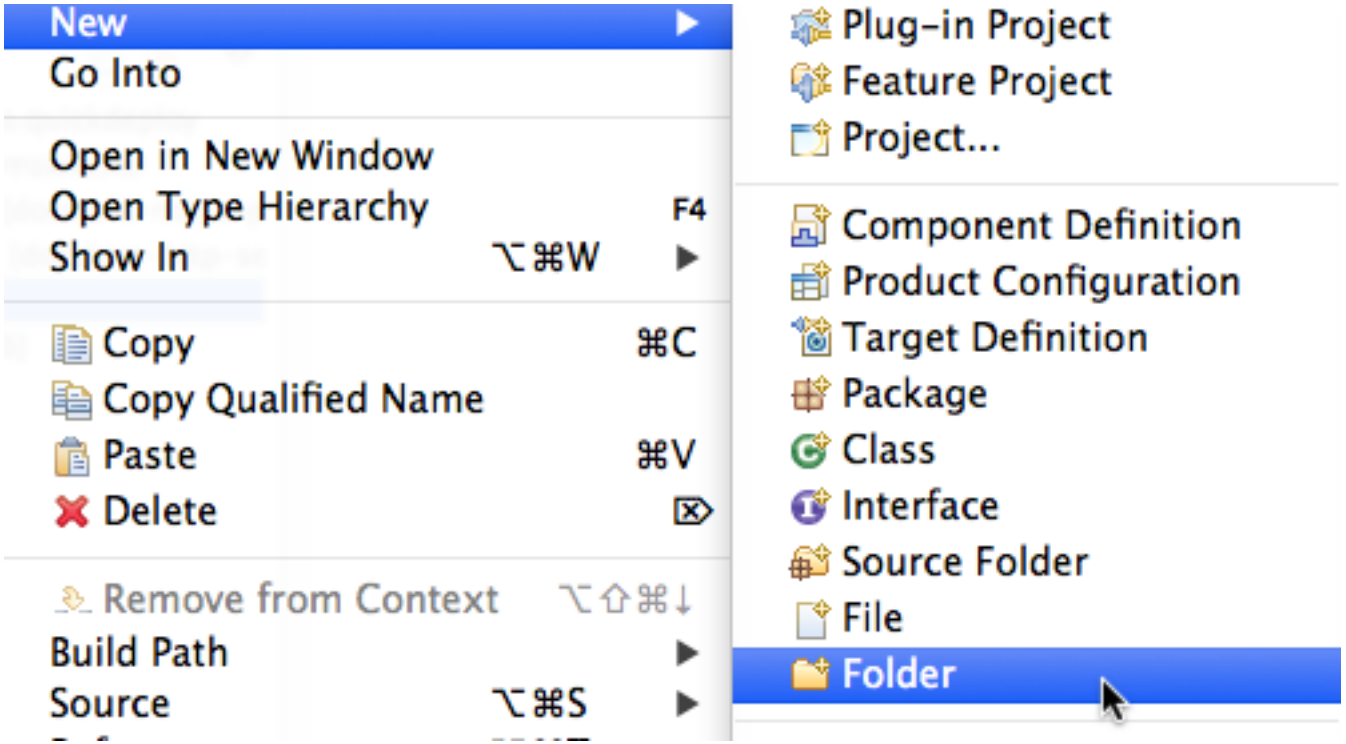
- DOTS – was, wie warum
- Startup und Task Extension Point
- Singleton-Nutzung
- Konfiguration über Annotationen
- Custom Console Commands
- Log4J2
- Links

Log4J2

- Log4J2 ist der Nachfolger des weit verbreiteten Java-Logging Frameworks Log4J.
  - http://logging.apache.org/log4j/2.x/
- Das Framework ist hoch-flexibel und skaliert extrem gut.
- Die Konfiguration kann über XML-Dateien oder Java-Code erfolgen.

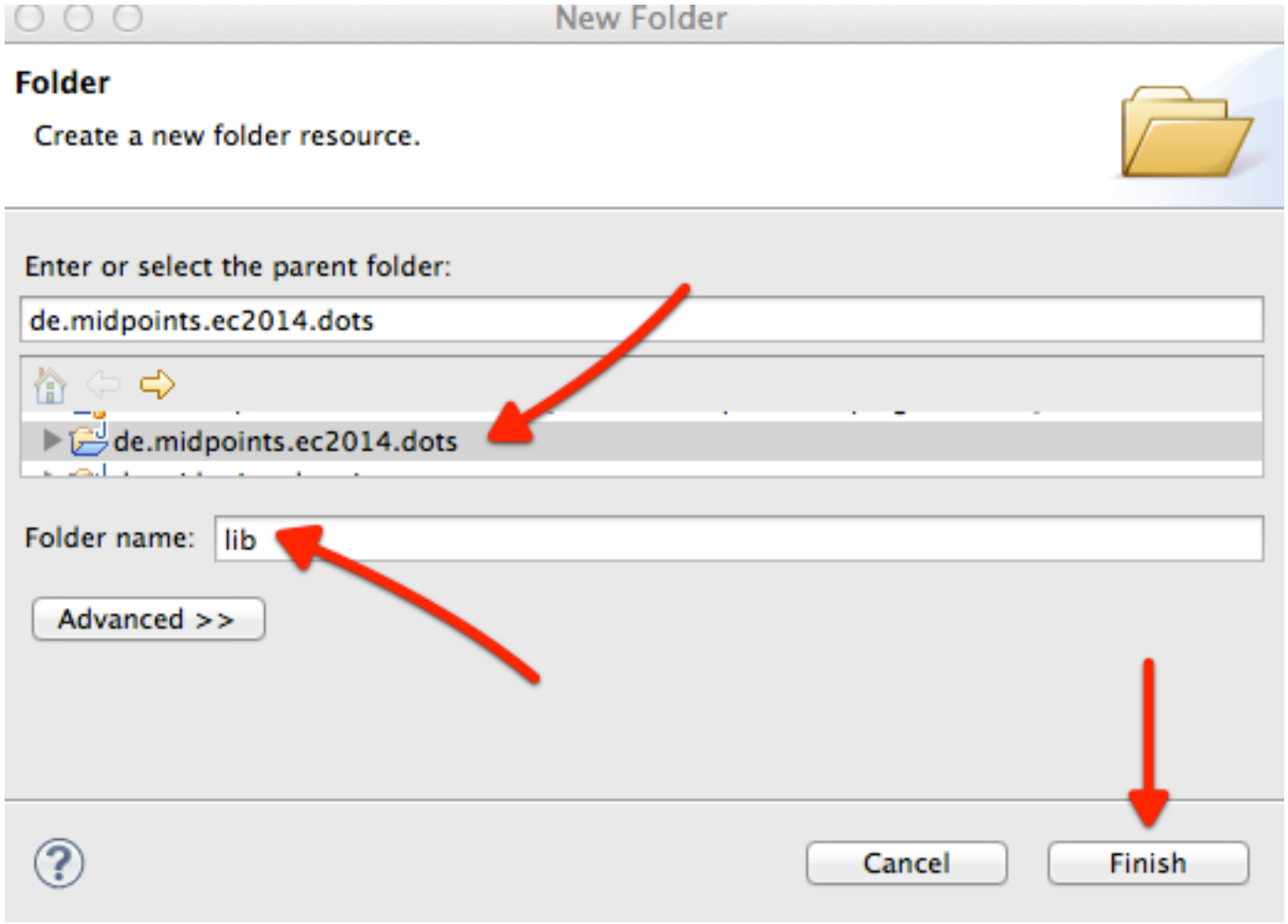
Log4J2-Implementierung in DOTS

- Erstellung eines Folder im Project



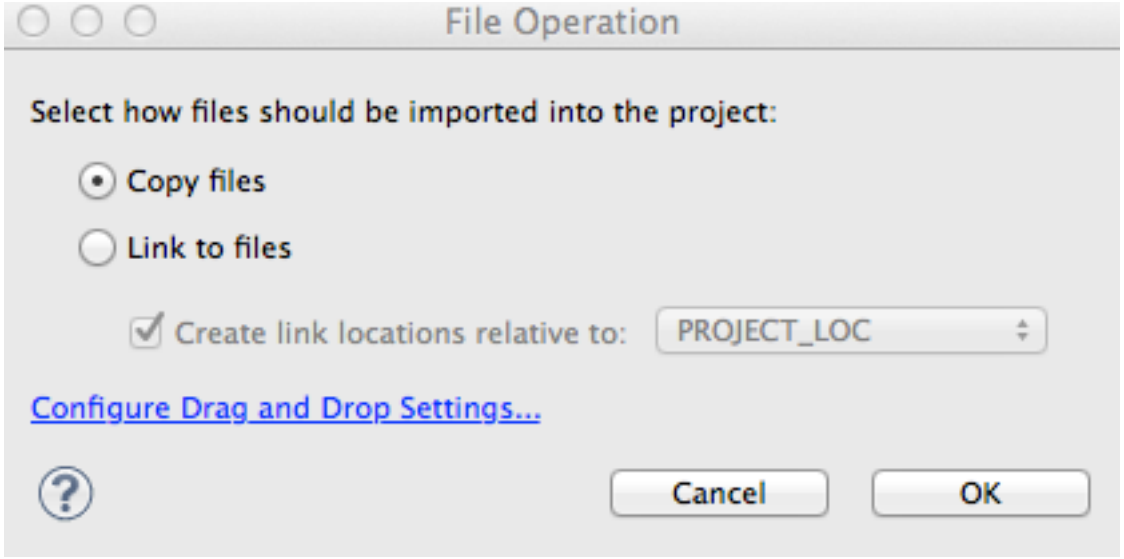
Log4J2-Implementierung in DOTS

- Erstellung eines Folder im Projekt



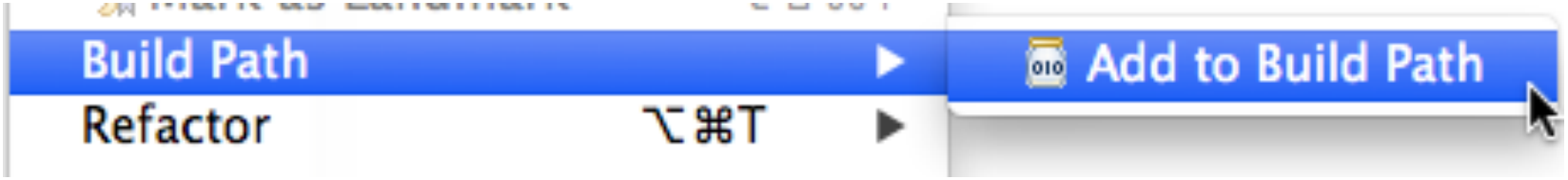
Log4J2-Implementierung in DOTS

- Hinzufügen der Libraries („api“ und „core“ sind erforderlich, beta9 vertestet, rc1 noch nicht) zum Folder per Drag‘n‘Drop.



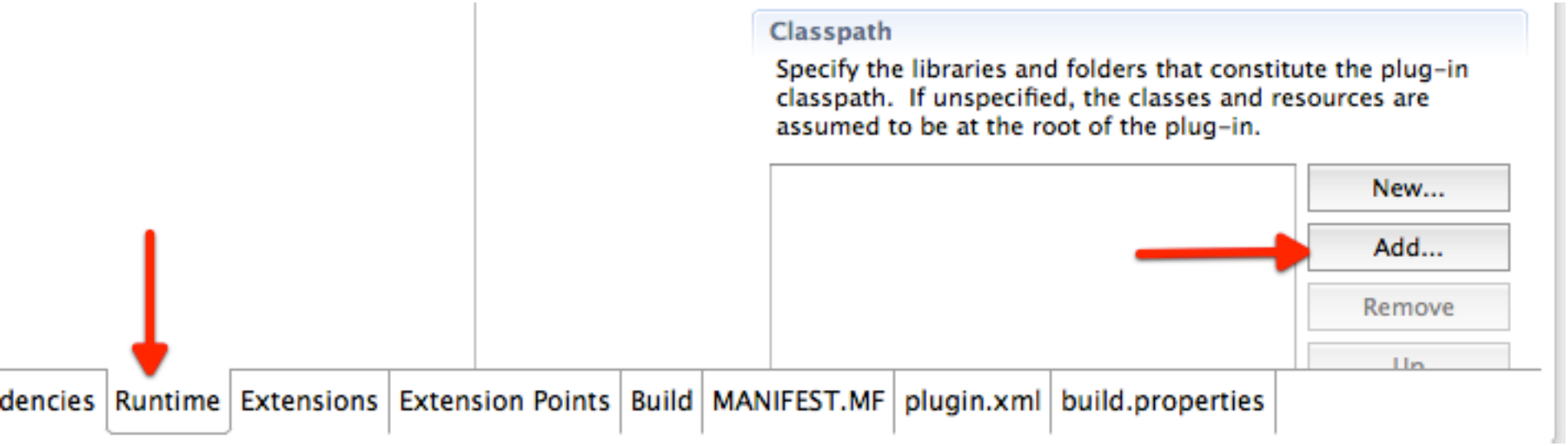
Log4J2-Implementierung in DOTS

- Hinzufügen der Libraries zum Build Path



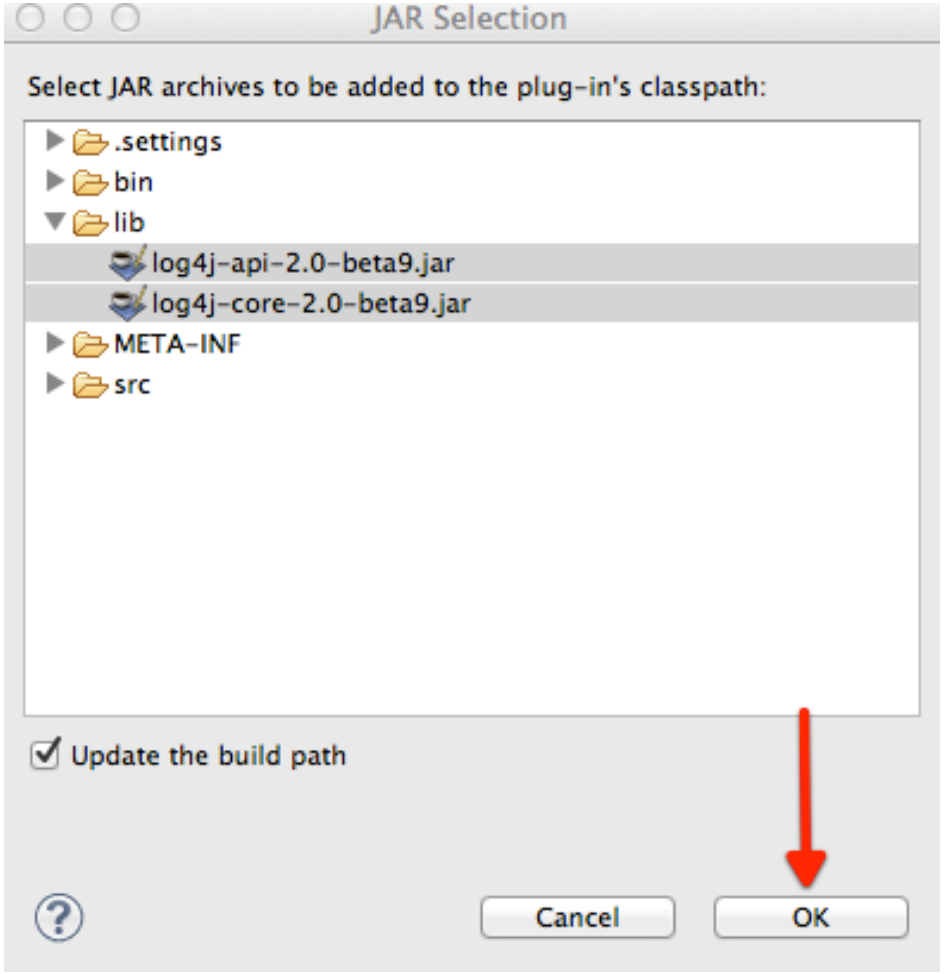
Log4J2-Implementierung in DOTS

- Hinzufügen der Libraries zum Classpath in der MANIFEST.MF



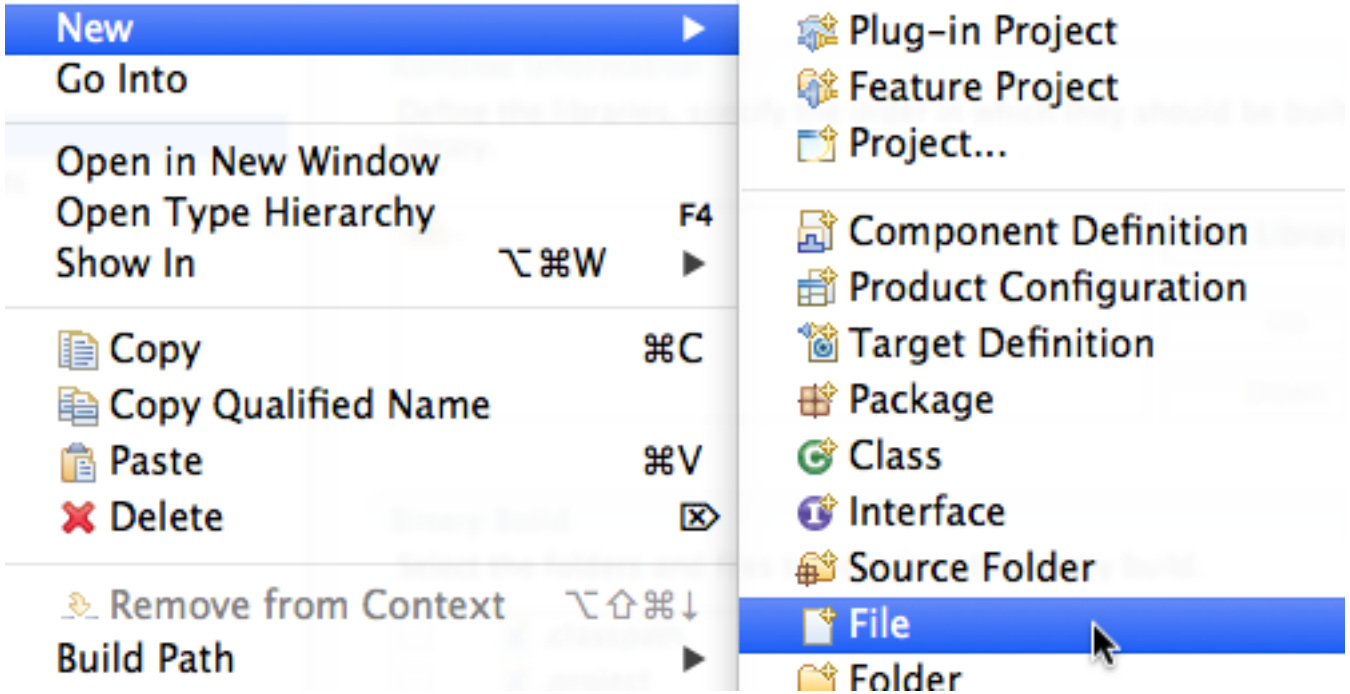
Log4J2-Implementierung in DOTS

- Hinzufügen der Libraries zum Classpath in der MANIFEST.MF



Log4J2-Implementierung in DOTS

- Wir erstellen eine neue Datei „log4j2.xml“ für die Log-Konfiguration.



Log4J2-Implementierung in DOTS

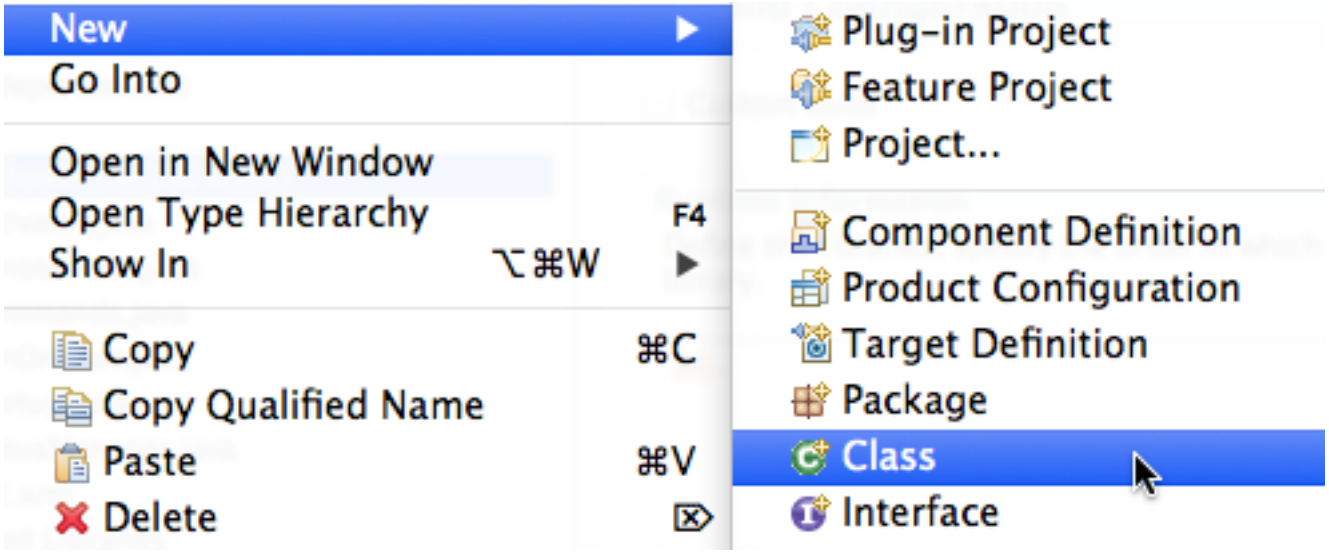
midpoints purify it

- Definition der Logging-Parameter in der XML-Datei

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration status="warn">
   <appenders>
      <Console name="Console" target="SYSTEM_OUT">
         <PatternLayout pattern="%d{yyyy-dd-MM HH:mm:ss.SSS} [%t] %-5level %logger{36} - %msg%n" />
      </Console>
      <RollingFile name="ec2014file"
         fileName="${sys:mplog.domino.data}/ec2014/ec.log"
         filePattern="${sys:mplog.domino.data}/ec2014/ec-%i.log.gz">
         <PatternLayout pattern="%d{yyyy-dd-MM HH:mm:ss.SSS} %-5level - %msg%n" />
         <SizeBasedTriggeringPolicy size="5 MB" />
         <DefaultRolloverStrategy max="10" />
      </RollingFile>
   </appenders>
   <loggers>
      <logger name="ec2014" level="debug" additivity="false">
         <appender-ref ref="ec2014file" />
      </logger>
      <root level="warn">
         <appender-ref ref="Console" />
      </root>
   </loggers>
</configuration>
```

Log4J2-Implementierung in DOTS

- Erstellung der neuen Klasse „Log“ zum Handling des Logging.



Log4J2-Implementierung in DOTS

midpoints
purify it

- Implementierung der Klasse „Log“ als Singleton.

```
public class Log {

   public final static Log instance   = new Log();

   private Log() {
   }

   public static Log getInstance() {
      return instance;
   }

   public void logWarning(String warning) {
      LogManager.getLogger("ec2014").warn(warning);
   }

   public void logError(String error) {
      LogManager.getLogger("ec2014").error(error);
   }

   public void logDebug(String debug) {
      LogManager.getLogger("ec2014").debug(debug);
   }
}
```

Log4J2-Implementierung in DOTS

midpoints
purify it

- Erweiterung des Startup-Tasklet um das Setzen der Java-Sytem-Property „mplog.domino.data“

```
public void earlyStartup() {
   NotesThread.sinitThread();
   try {
      Session session = NotesFactory.createSession();
      System.setProperty("mplog.domino.data", session.getEnvironmentString("Directory", true));
      ServerTaskManager.getInstance().logMessageText("Early Startup started");
   } catch (Exception e) {
      ServerTaskManager.getInstance().logMessageText("Error on initialization.");
      e.printStackTrace();
   } finally {
      NotesThread.stermThread();
   }
}
```

Log4J2-Implementierung in DOTS

midpoints
purify it

- Beispiel-Implementierung des Logging im RunOnStart-Tasklet.

```
@Override
public void run(RunWhen runWhen, String[] arguments, IProgressMonitor monitor) throws NotesException {
    logMessage("RunOnStart Tasket started");
    StatusManager.getInstance().addProgressMonitor("RunOnStart", monitor);
    Log.getInstance().logDebug("RunOnStart Tasklet started");
    Log.getInstance().logWarning("Warning gelogged");
}
```

Agenda

- DOTS – was, wie warum
- Startup und Task Extension Point
- Singleton-Nutzung
- Konfiguration über Annotationen
- Custom Console Commands
- Log4J2
- Links

Links

- Domino OSGi Tasklet Service auf OpenNTF
  - http://www.openntf.org/main.nsf/project.xsp?r=project/OSGI%20Tasklet%20Service%20for%20IBM%20Lotus%20Domino
  - readme.pdf im Download beachten, enthält vielfältige Informationen
- AD105 – Domino OSGi unleashed
  - http://public.dhe.ibm.com/software/dw/ru/download/D-Savustjan-LCTY.pdf
  - Lotusphere 2011

Links

- SHOW112 – Domino OSGi DevelopmentDomino OSGi Tasklet Service auf OpenNTF
  - http://www.slideshare.net/fiorep/domino-osgi-development
  - Lotusphere 2012
- BP207 – Meet the Java Application Server You Already Own
  - http://www.slideshare.net/sbasegmez/bp207-download
  - Connect 2013

Links

- DOTS Debugging (Domino Debug Plug-In)
  - http://www.openntf.org/internal/home.nsf/project.xsp?name=IBM+Lotus+Domino+Debug+Plugin

Vielen Dank!

midpoints
purify it