

DOTS - reloaded

EntwicklerCamp 2014

René Winkelmeier
midpoints GmbH



Über mich



René Winkelmeyer
Senior Consultant



midpoints GmbH
<http://www.midpoints.de>

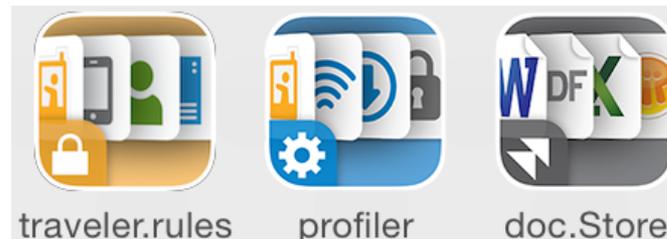
IBM Advanced Business Partner
IBM Design Partner

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



Über mich



René Winkelmeyer
Senior Consultant



- 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

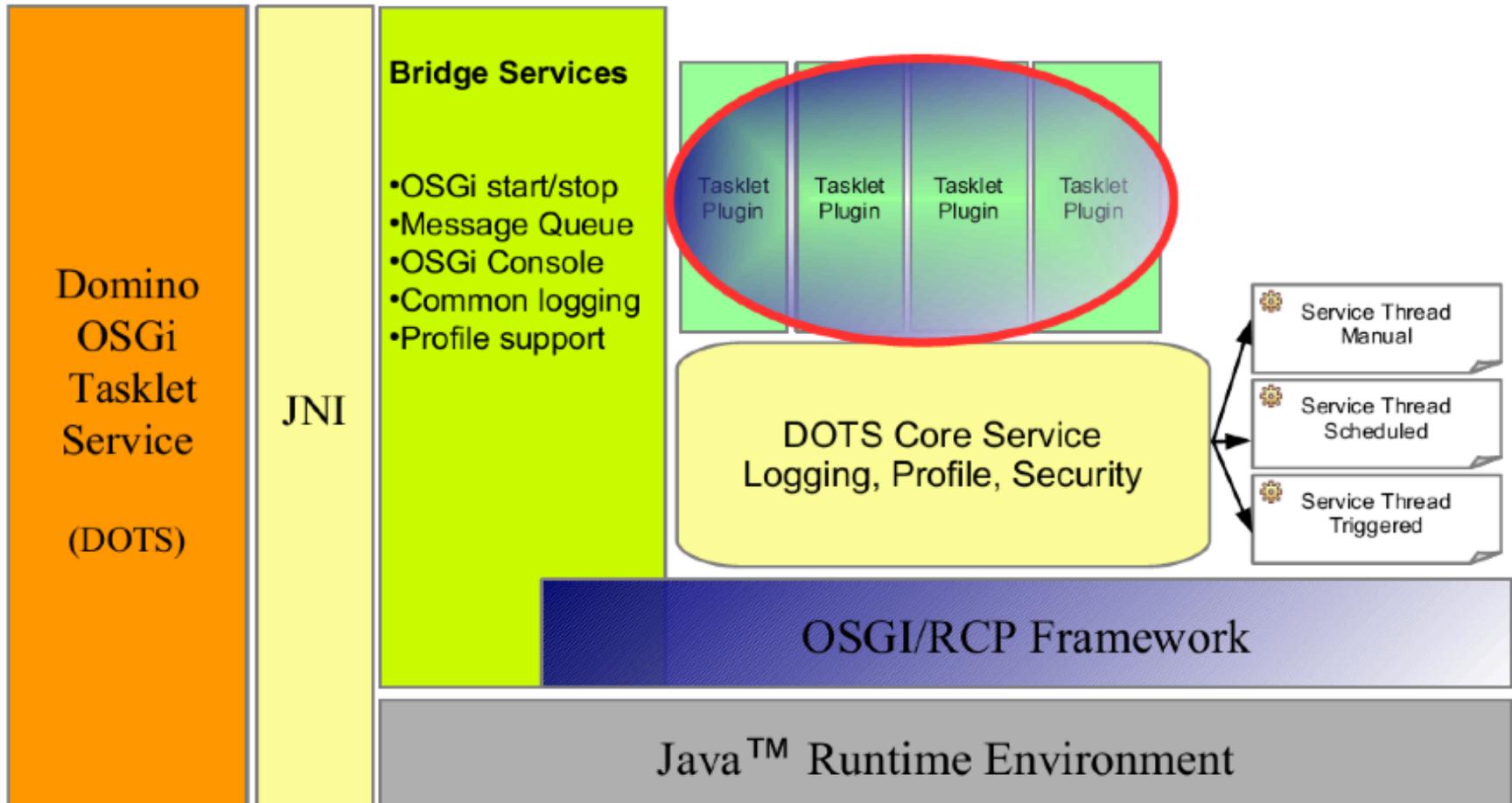
- DOTS – was, wie warum
- Startup und Task Extension Point
- Singleton-Nutzung
- Konfiguration über Annotationen
- Custom Console Commands
- Log4J2
- Links

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 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

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

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

- Wie kann ich DOTS-Plug-Ins deployen?
 - Über das Filesystem
 - <dominoprogram>/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

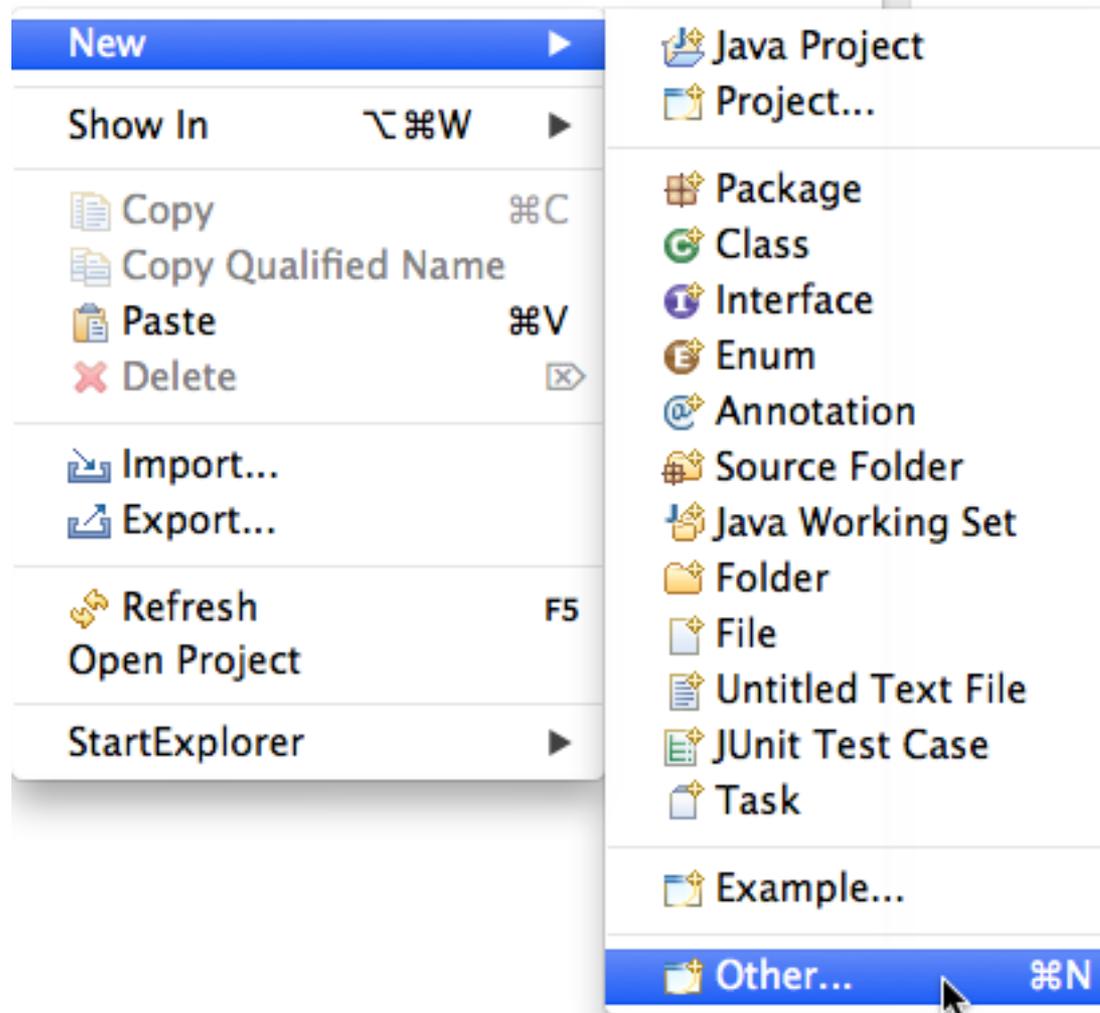
- Über so genannte Extension Points können sich OSGi Plug-Ins an von anderen Plug-Ins bereitgestellte Extensions „anhängen“.
- Die meist benutzten Extension Points für DOTS sind
 - Startup
 - Task

Der Startup Extension Point

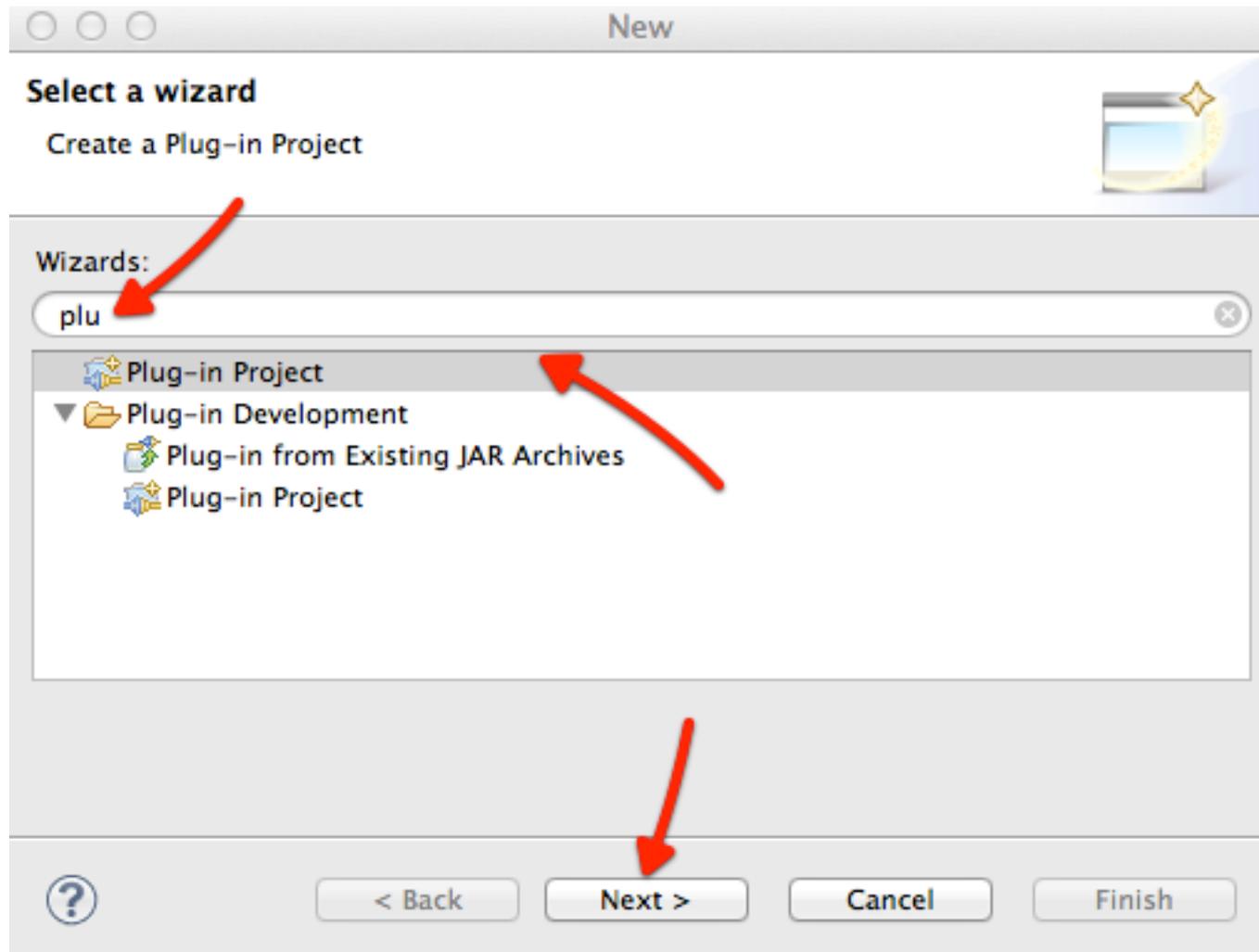
- Der Code, welcher über diesen Extension Point angesprochen wird, läuft noch vor Ausführung der Tasklets.

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

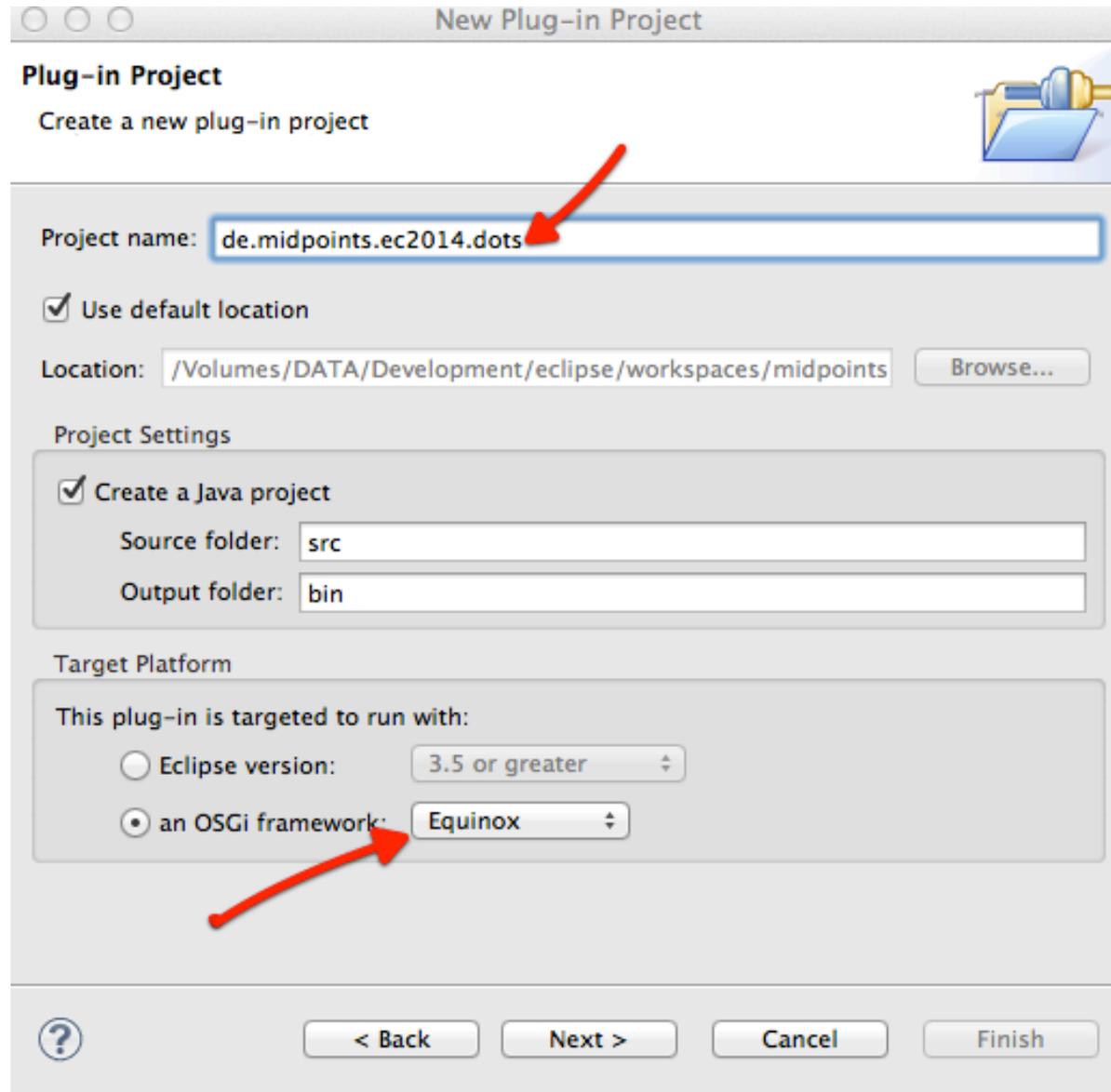
Erstellung des initialen Plug-In



Erstellung des initialen Plug-In



Erstellung des initialen Plug-In



New Plug-in Project

Plug-in Project
Create a new plug-in project

Project name:

Use default location

Location:

Project Settings

Create a Java project

Source folder:

Output folder:

Target Platform

This plug-in is targeted to run with:

Eclipse version:

an OSGi framework:

Erstellung des initialen Plug-In

New Plug-in Project

Content
Enter the data required to generate the plug-in.

Properties

ID:

Version:

Name:

Vendor:

Execution Environment:

Options

Generate an activator, a Java class that controls the plug-in's life cycle
Activator:

This plug-in will make contributions to the UI

Enable API analysis



Einbindung des Startup Extension Point

General Information

This section describes general information about this plug-in.

ID:

Version:

Name:

Vendor:

Platform Filter:

Activator:

Activate this plug-in when one of its classes is loaded

This plug-in is a singleton

Execution Environments

Specify the minimum execution environments required to run this plug-in.

Plug-in Content

The content of the plug-in is made up of two sections:

 [Dependencies](#): lists all the plug-ins required on the plug-in's classpath to compile and run.

 [Runtime](#): lists the libraries that make up this plug-in's runtime.

Extension / Extension Point Content

This plug-in may define extensions and extension points.

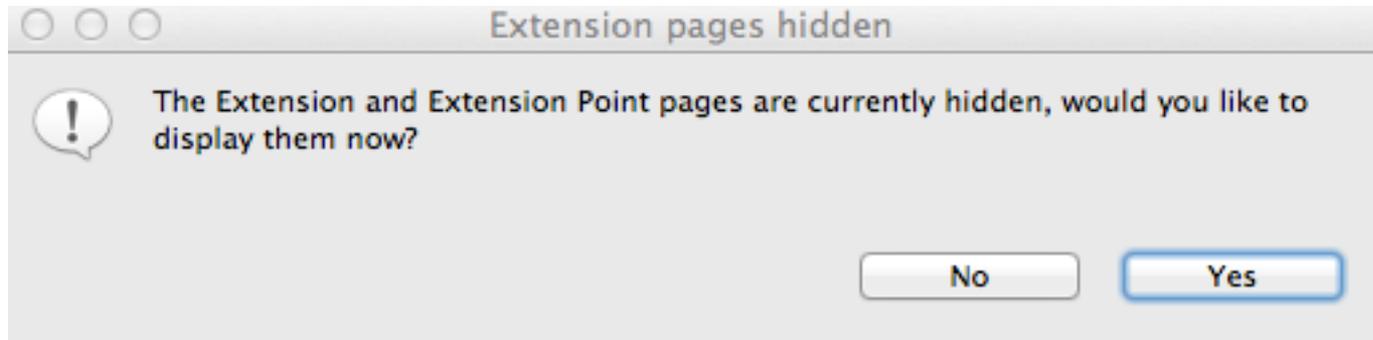
 [Extensions](#): declares contributions this plug-in adds to the platform.

 [Extension Points](#): declares new function points this plug-in adds to the platform.

Testing

Test this plug-in by launching a separate Eclipse application:

Einbindung des Startup Extension Point



Einbindung des Startup Extension Point

All Extensions ↓ a z 📄 📄

Define extensions for this plug-in in the following section.

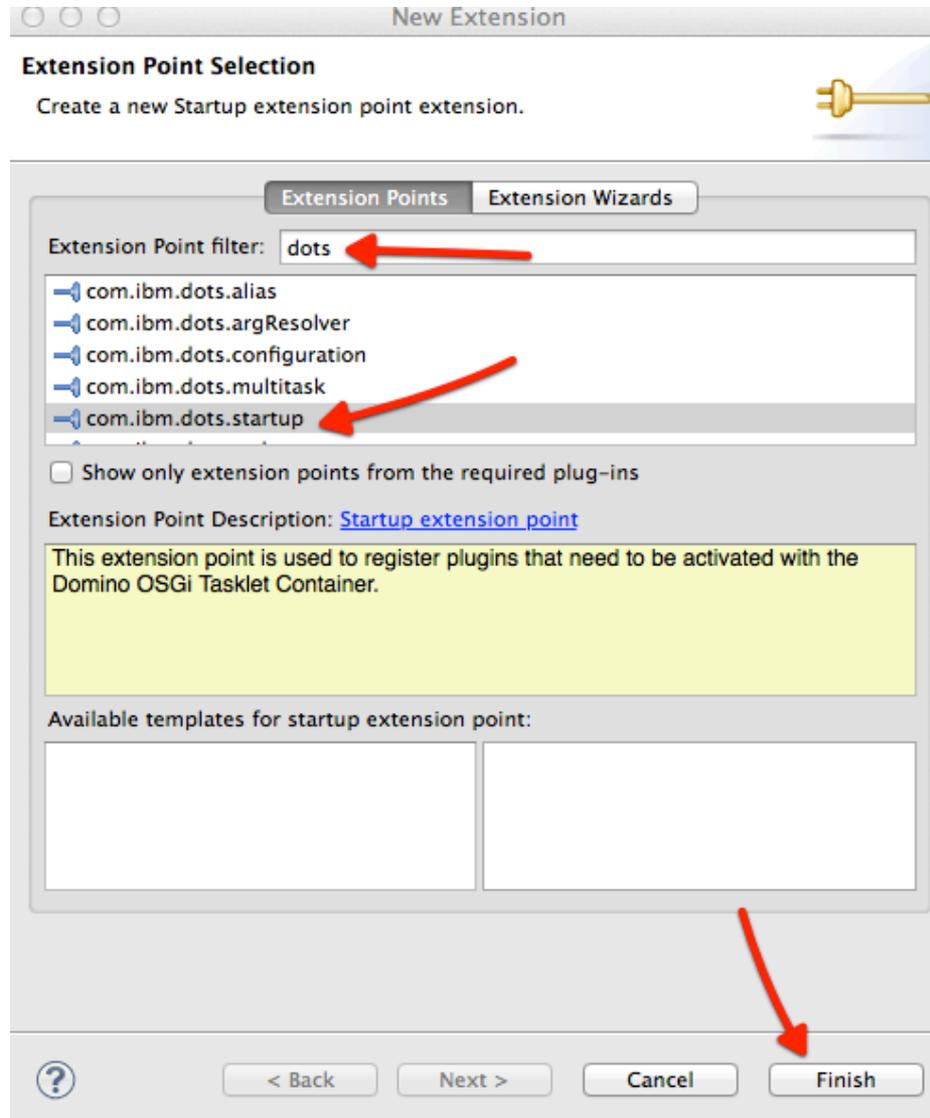
Add...

Remove

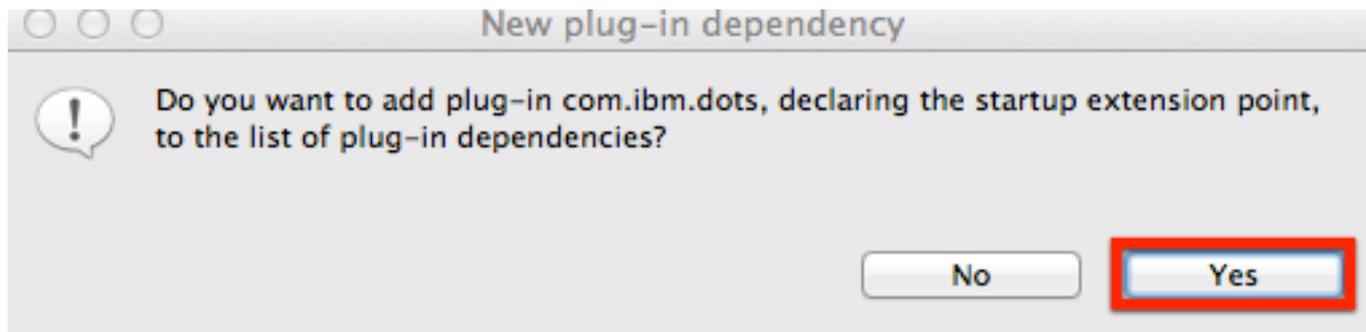
Up

Down

Einbindung des Startup Extension Point



Einbindung des Startup Extension Point



Einbindung des Startup Extension Point

All Extensions



Define extensions for this plug-in in the following section.

type filter text

- ▼ com.ibm.dots.startup
 - ✕ Startup1 (startup)

Add...

Remove

Up

Down

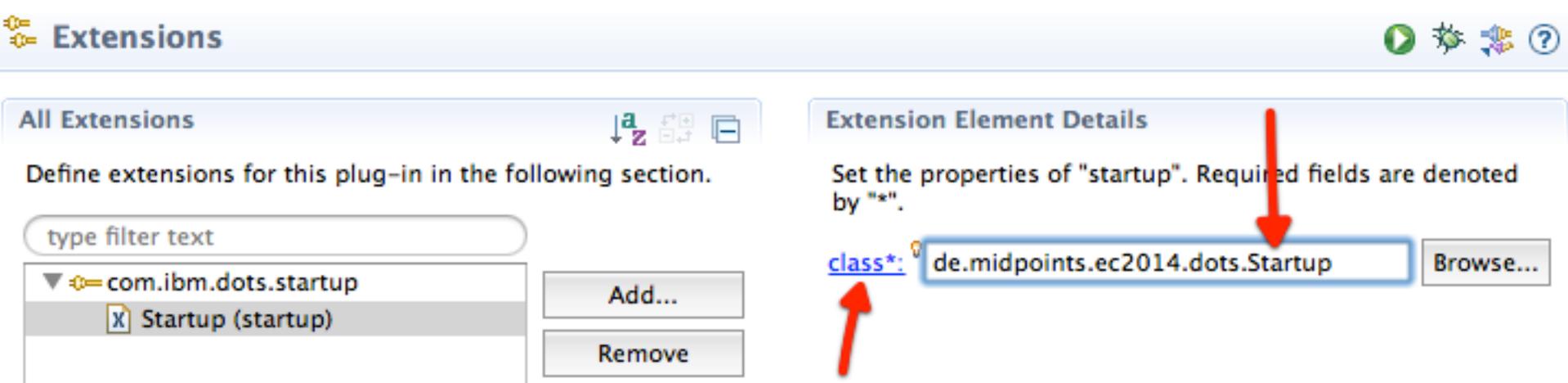
Extension Element Details

Set the properties of "startup". Required fields are denoted by "*".

class*:



Einbindung des Startup Extension Point



The screenshot shows the Eclipse IDE interface. On the left, the 'Extensions' view is open, displaying a tree structure under 'All Extensions'. The 'com.ibm.dots.startup' extension is expanded, showing a sub-extension 'Startup (startup)'. On the right, the 'Extension Element Details' view is open, showing the configuration for the 'startup' extension point. The 'class*' field is highlighted with a red box, and the value 'de.midpoints.ec2014.dots.Startup' is entered. A red arrow points to the 'class*' label, and another red arrow points to the text box containing the class name.

Extensions

All Extensions

Define extensions for this plug-in in the following section.

type filter text

- com.ibm.dots.startup
 - Startup (startup)

Add...
Remove

Extension Element Details

Set the properties of "startup". Required fields are denoted by "*".

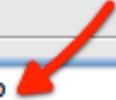
class*: Browse...

Einbindung des Startup Extension Point

Source folder:

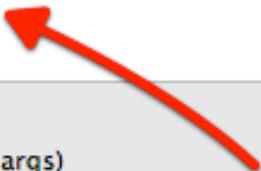
Package:

Enclosing type:

Name: 

Modifiers: public default private protected
 abstract final static

Superclass:

Interfaces: 

Which method stubs would you like to create?

public static void main(String[] args)
 Constructors from superclass
 Inherited abstract methods

Do you want to add comments? (Configure templates and default value [here](#))

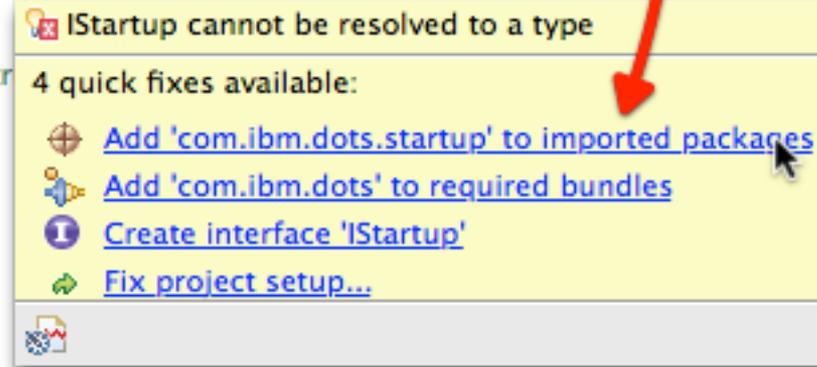
Generate comments

Einbindung des Startup Extension Point

```
import com.ibm.dots.startup.IStartup;
```

```
public class RunOnStart implements IStartup {
```

```
    public RunOnStart() {  
        // TODO Auto-generated constr  
    }  
}
```



IStartup cannot be resolved to a type

4 quick fixes available:

-  [Add 'com.ibm.dots.startup' to imported packages](#)
-  [Add 'com.ibm.dots' to required bundles](#)
-  [Create interface 'IStartup'](#)
-  [Fix project setup...](#)

Einbindung des Startup Extension Point

```
public class RunOnStart implements IStartup {  
    public RunOnStart() {  
        // TODO Auto-generated constructor stub  
    }  
}
```

Einbindung des Startup Extension Point

```
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

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

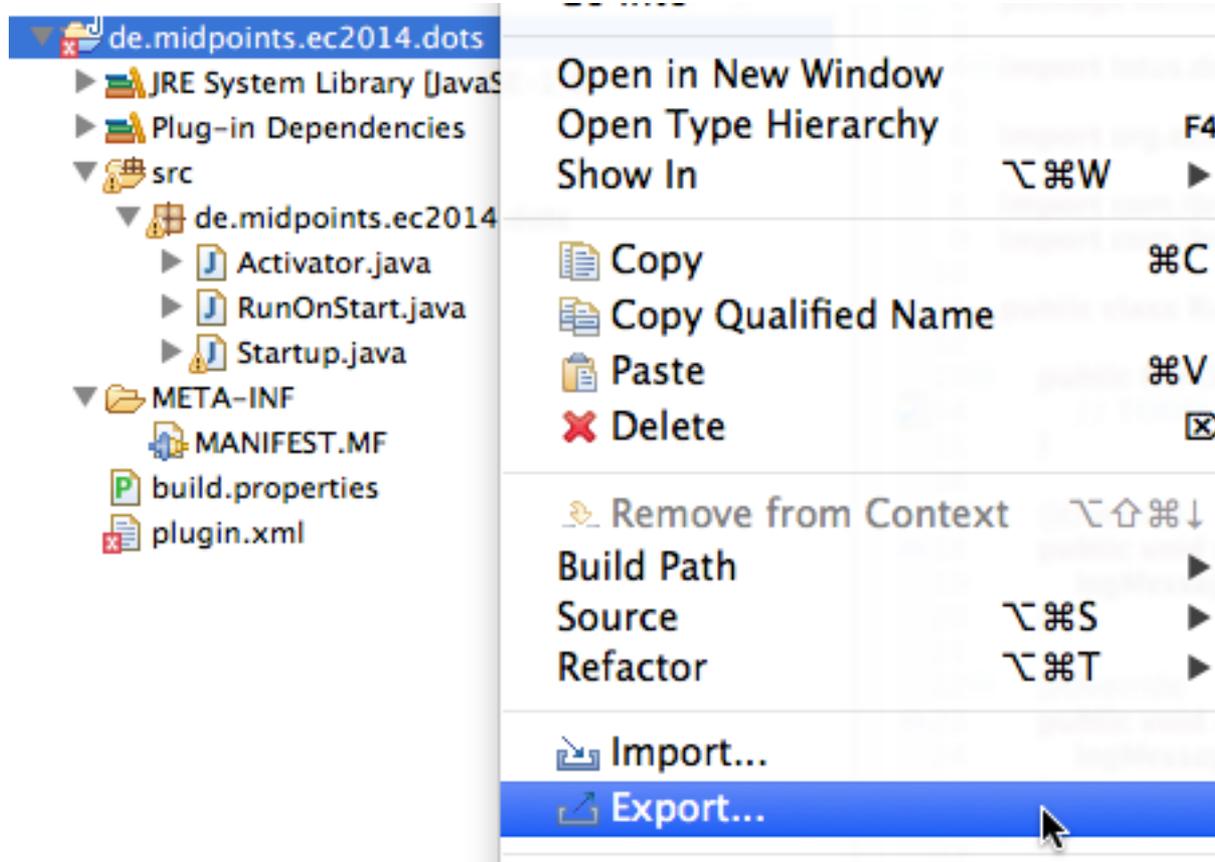
import com.ibm.dots.startup.IStartup;
import com.ibm.dots.task.ServerTaskManager;

public class RunOnStart implements IStartup {

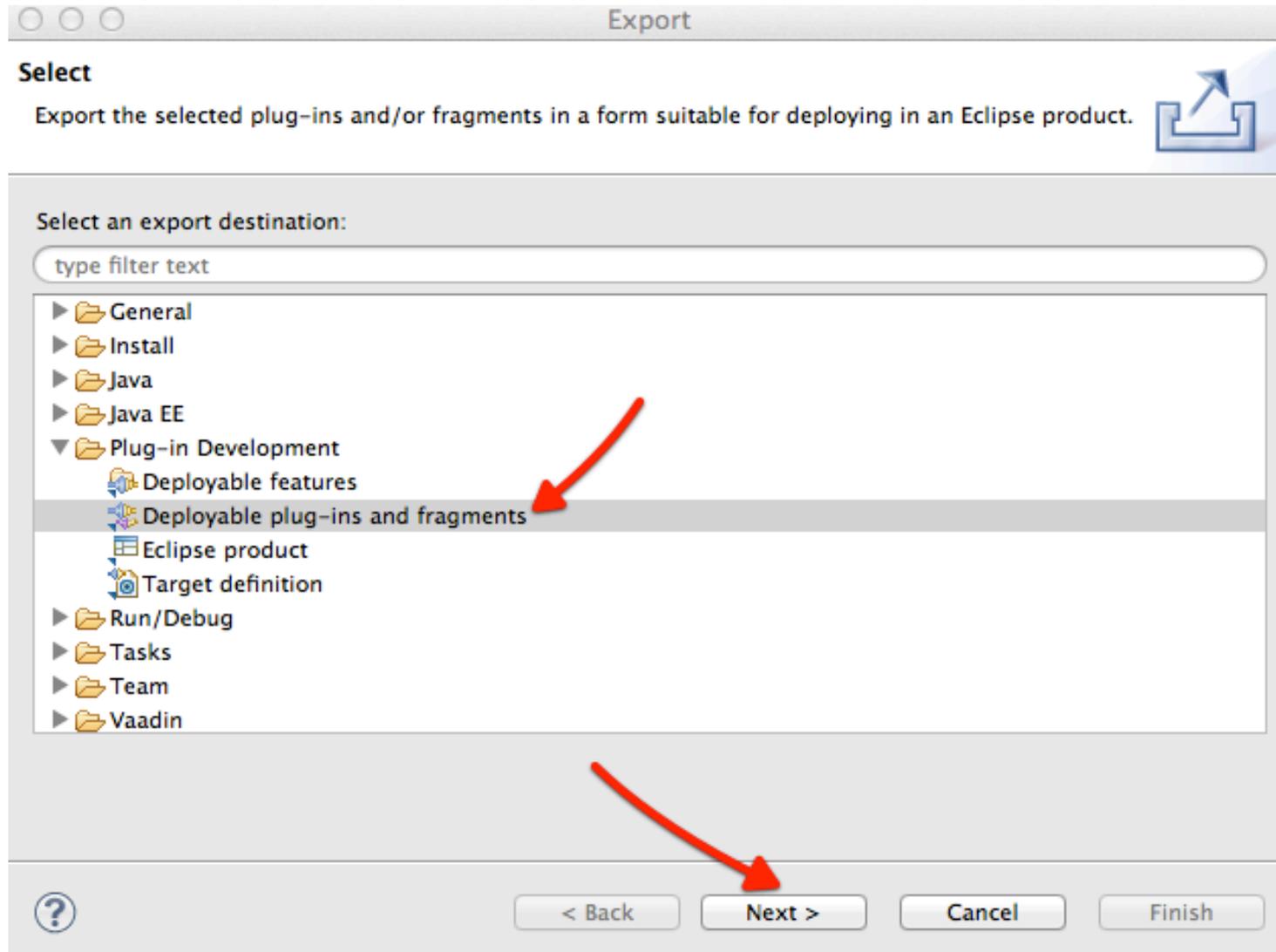
    public RunOnStart() {
    }

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

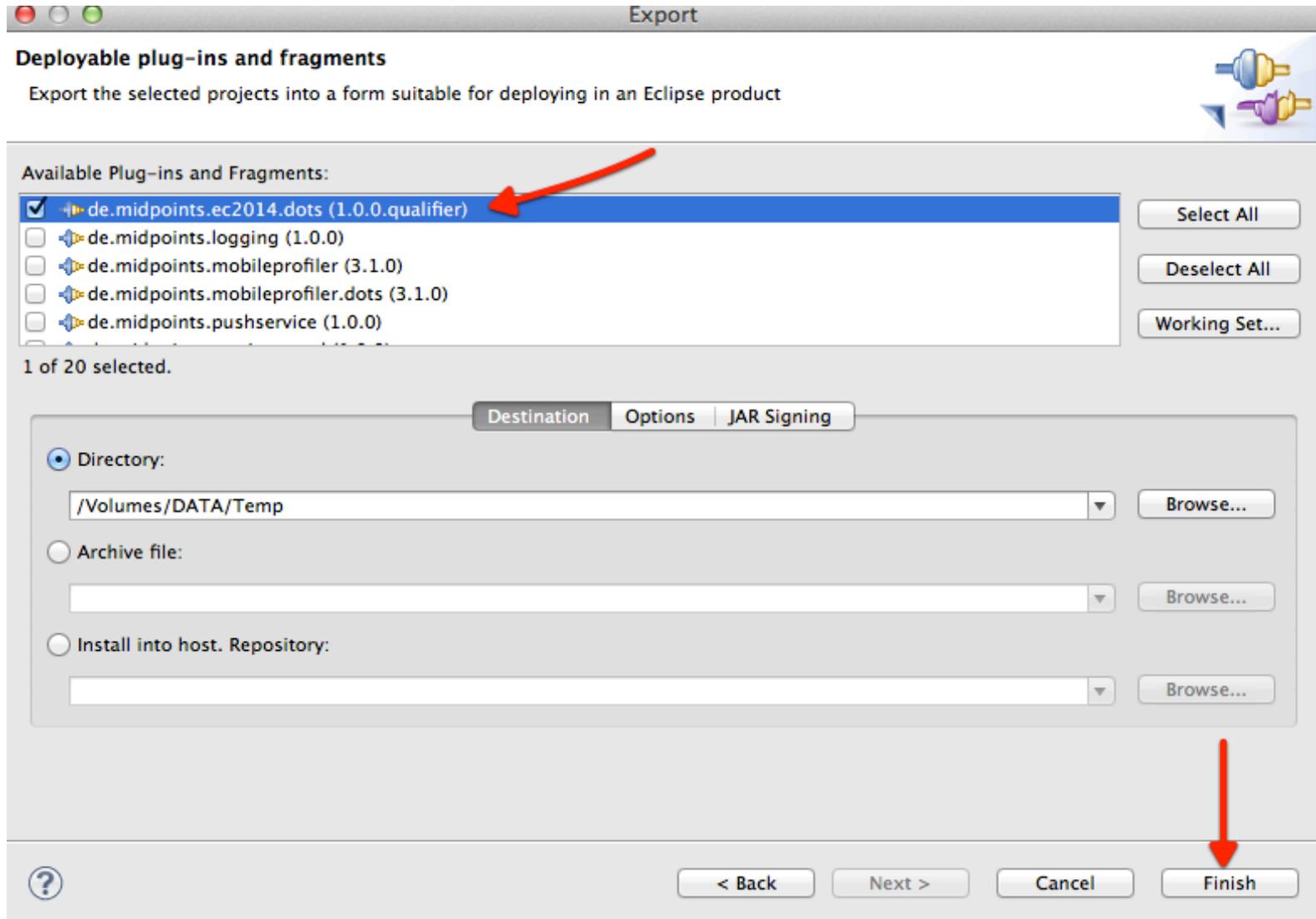
Erstes Test-Deployment



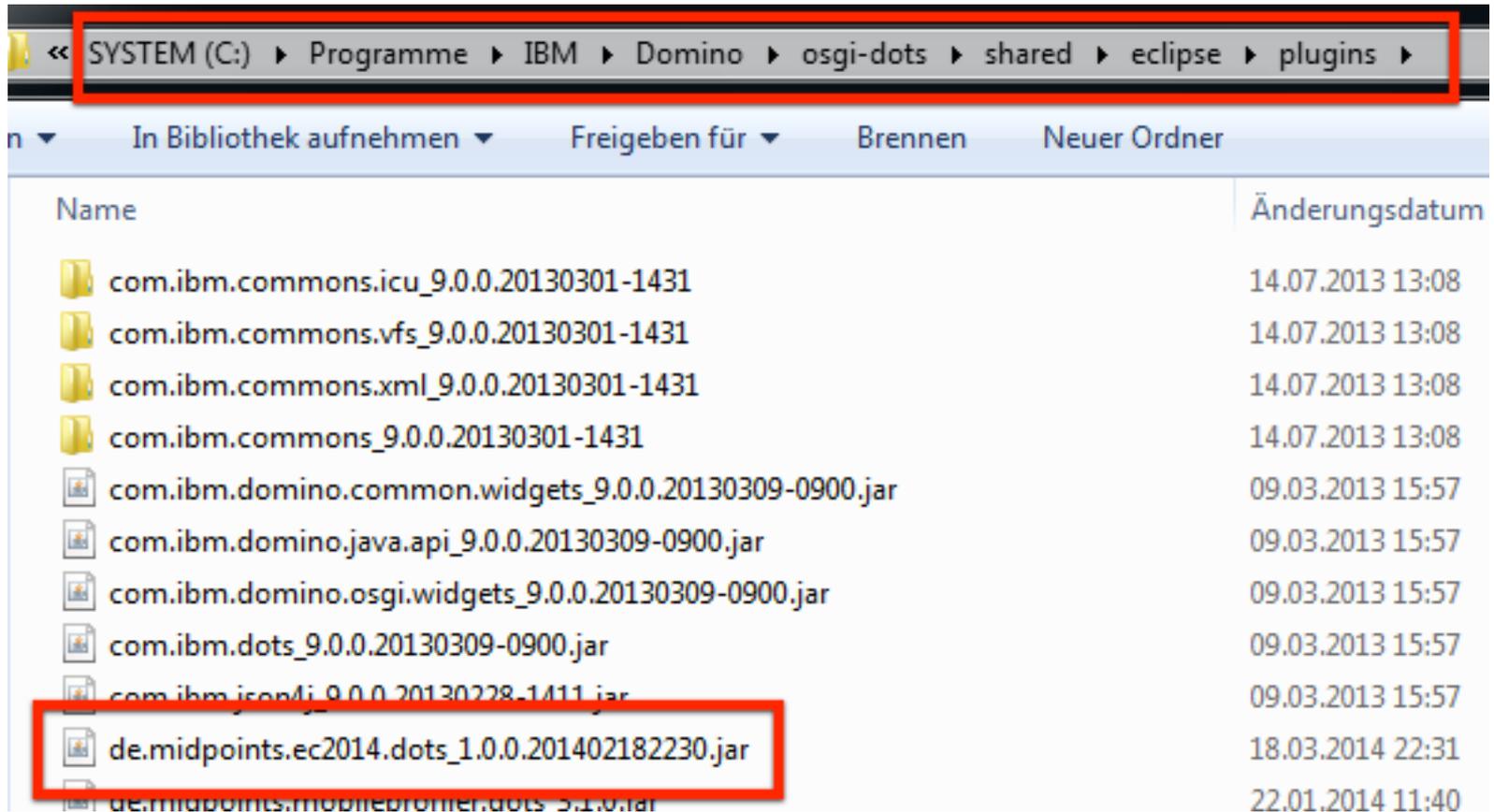
Erstes Test-Deployment



Erstes Test-Deployment



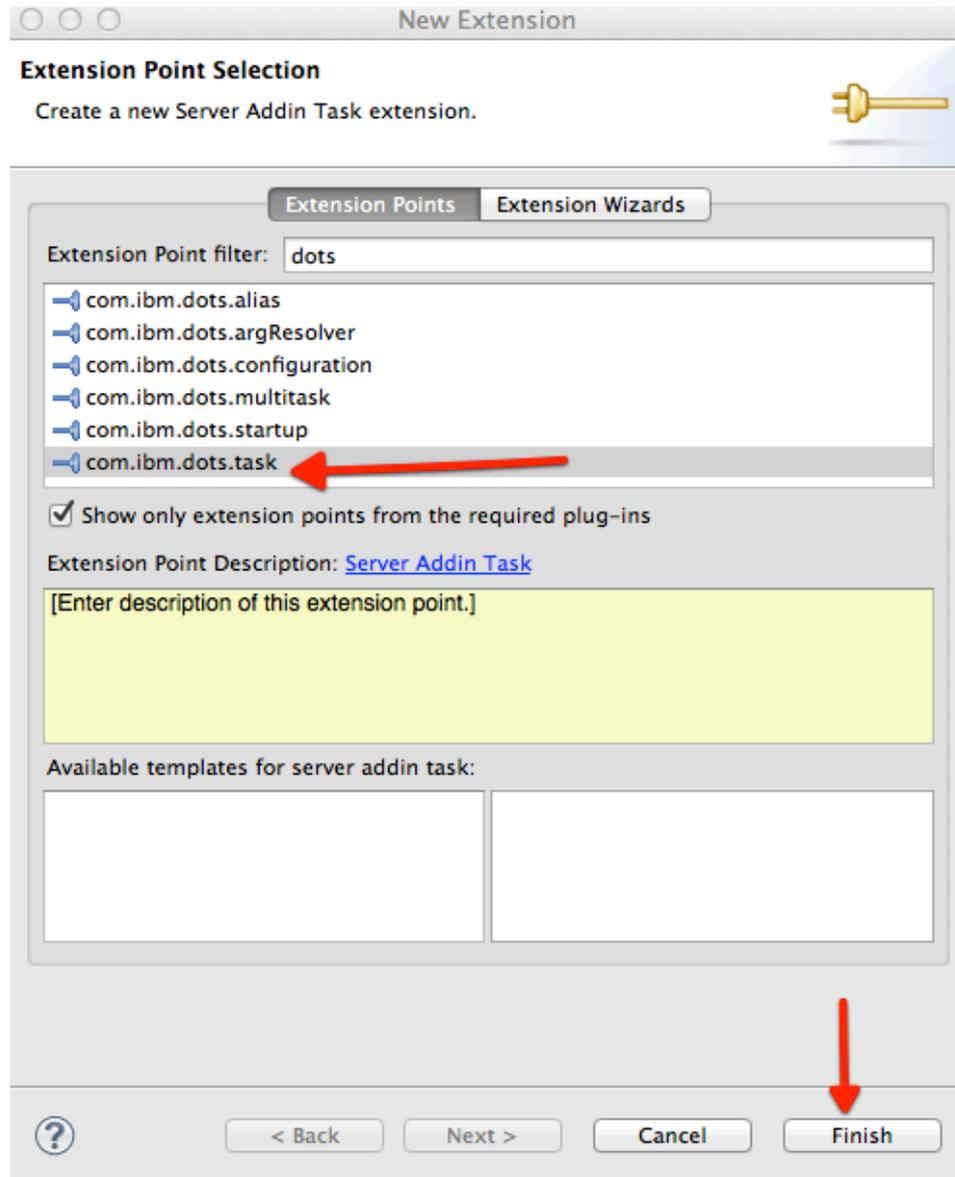
Erstes Test-Deployment



- DOTS-Task mit „load dots“ starten

```
address 0001  
[DOTS] Early Startup started  
Domino OSGi Tasklet Container started < profile DOTS >  
Admin Process: Searching Administration Requests database
```

Einbindung des Task Extension Point



Einbindung des Task Extension Point

Extensions

All Extensions

Define extensions for this plug-in in the following section.

type filter text

- com.ibm.dots.startup
 - Startup (startup)
- com.ibm.dots.task
 - AbstractServerTask:2 (task)

Add...
Remove
Up

Extension Element Details

Set the properties of "task". Required fields are denoted by "*".

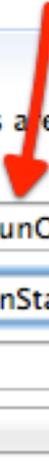
class*: de.midpoints.ec2014.dots.RunOnS Browse...

id*: midpoints.ec2014.dots.RunOnStart Browse...

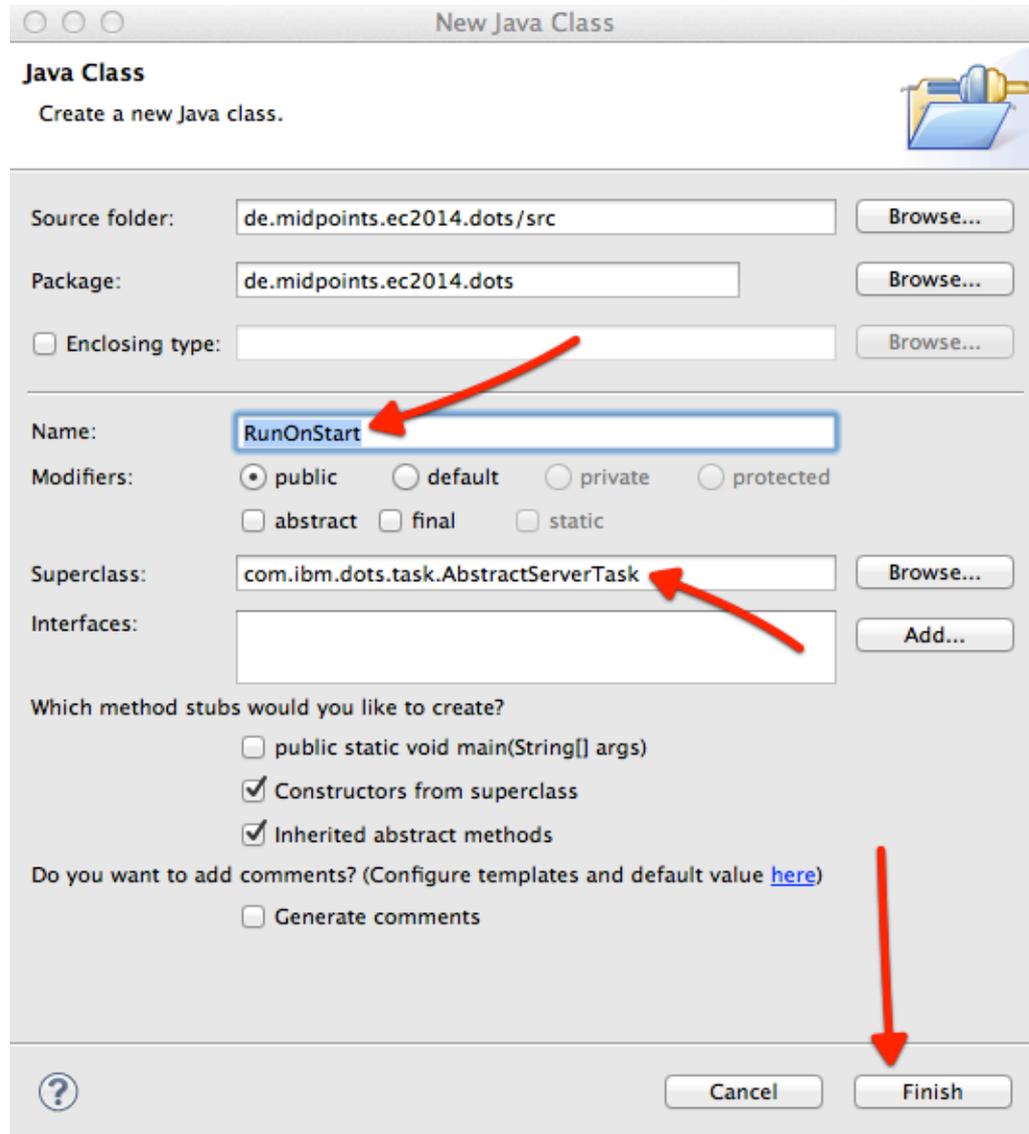
description:

runOnStart: true

triggered:



Einbindung des Task Extension Point



Java Class
Create a new Java class.

Source folder:

Package:

Enclosing type:

Name:

Modifiers: public default private protected
 abstract final static

Superclass:

Interfaces:

Which method stubs would you like to create?

- public static void main(String[] args)
- Constructors from superclass
- Inherited abstract methods

Do you want to add comments? (Configure templates and default value [here](#))

- Generate comments

Einbindung des Task Extension Point

```
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

```
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 Tasklet started");  
    }  
}
```

Test des Task Extension Point

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

```
Domino OSGi Tasklet Container started < profile DOTS >
[DOTS] Early Startup started
[DOTS] <de.midpoints.ec2014.dots.RunOnStart> RunOnStart Tasket started
[DOTS] <de.midpoints.ec2014.dots.RunOnStart> RunOnStart Tasklet finished
[DOTS] <de.midpoints.ec2014.dots.RunOnStart> RunOnStart Tasket finished
```

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)

All Extensions ↓ a z 🔄 📄

Define extensions for this plug-in in the following section.

type filter text

- ▶ com.ibm.dots.startup
- ▼ com.ibm.dots.task
 - RunOnStart (task)
 - ▼ ScheduledTask (task)
 - (run)
 - (run)
 - Maintenance (task)
 - ▶ TimedTask (task)

Extension Element Details

Set the properties of "run". Required fields are denoted by "*".

every*:

unit*:

startAt:

stopAt:

Wissenswertes 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

Wissenswertes 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

- 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 Beenden 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 Tasklet 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

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

Java-Annotationen allgemein

- 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_\(Java\)](http://de.wikipedia.org/wiki/Annotation_(Java))

DOTS-Konfiguration mit Annotationen

- Annotationen 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

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

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

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

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

DOTS-Konfiguration mit Annotationen

```
@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

- 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 uns 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



```
import org.eclipse.osgi.framework.console.CommandInterpreter;  
import org.eclipse.osgi.framework.console.CommandProvider;  
  
public class Commands implements CommandProvider {
```

Commands-Klasse erstellen

```
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

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

Custom Console Commands



```
[DOTS] Maintenance gestartet  
[DOTS]  
[DOTS] Maintenance beendet  
[DOTS]
```

Agenda

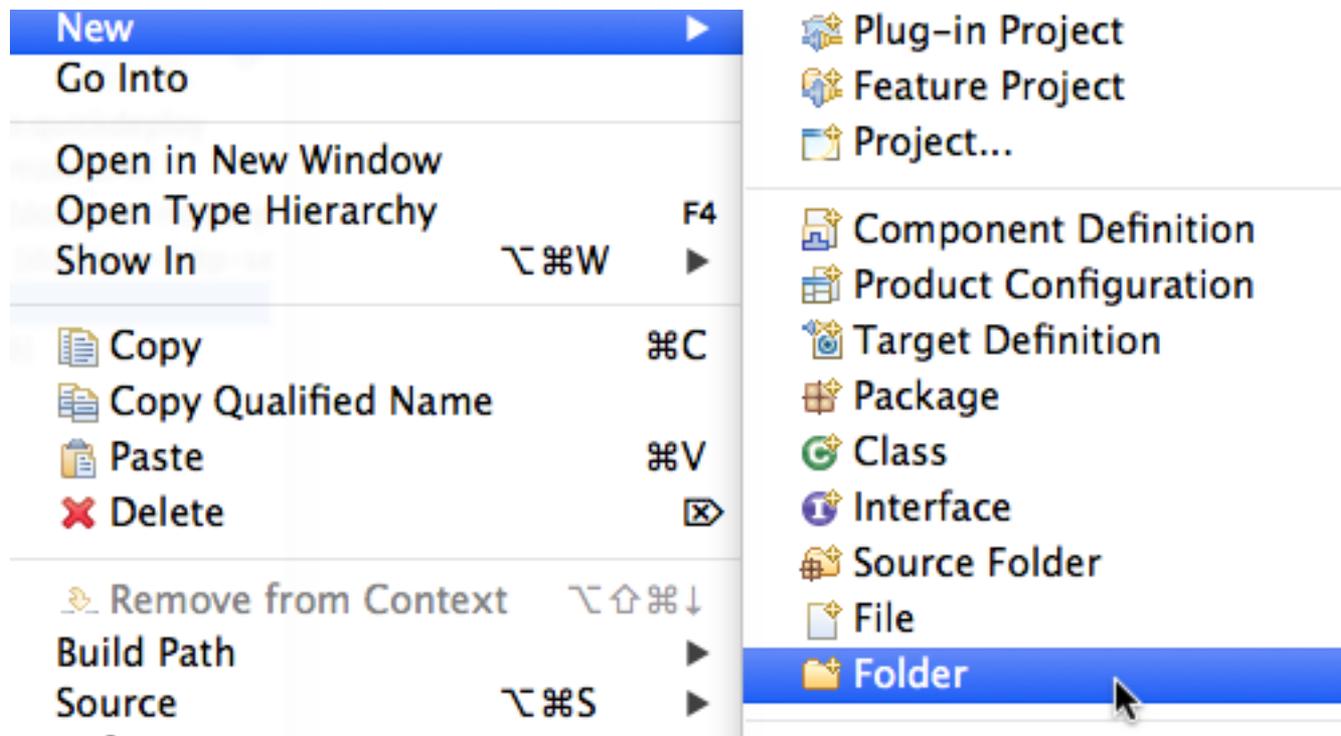
- DOTS – was, wie warum
- Startup und Task Extension Point
- Singleton-Nutzung
- Konfiguration über Annotationen
- Custom Console Commands
- Log4J2
- Links

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.

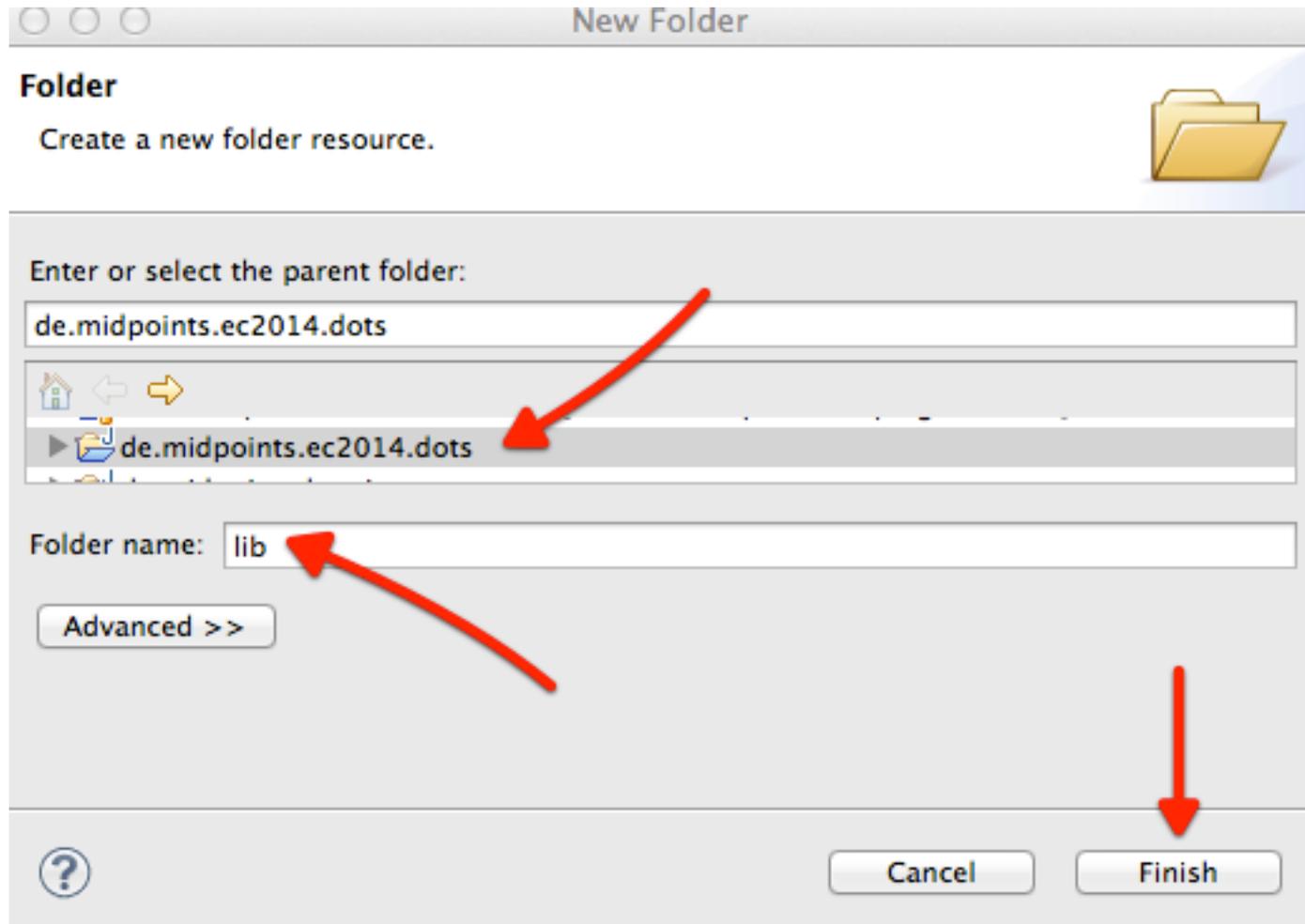
Log4J2-Implementierung in DOTS

- Erstellung eines Folder im Project



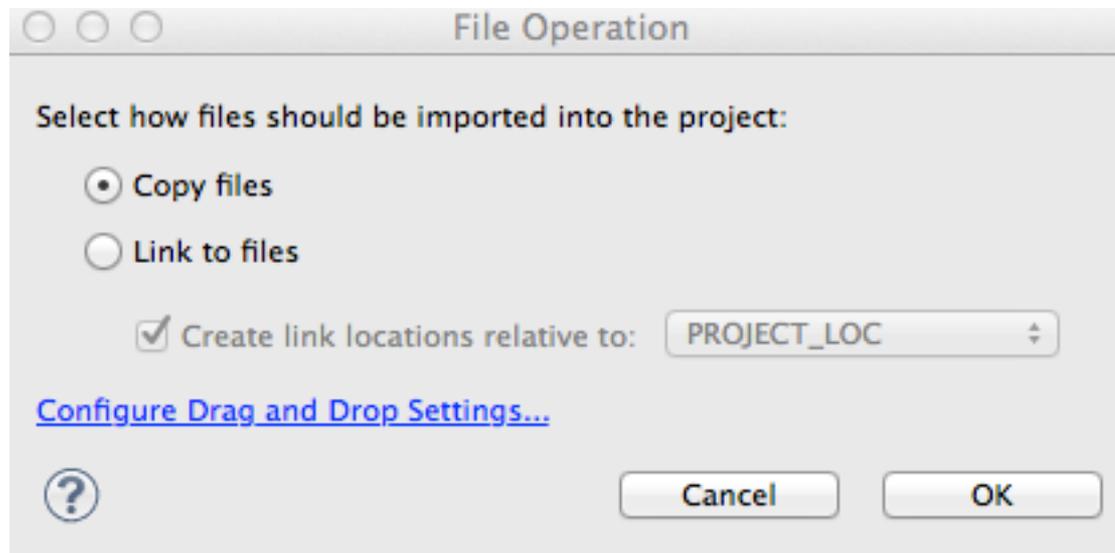
Log4J2-Implementierung in DOTS

- Erstellung eines Folder im Projekt



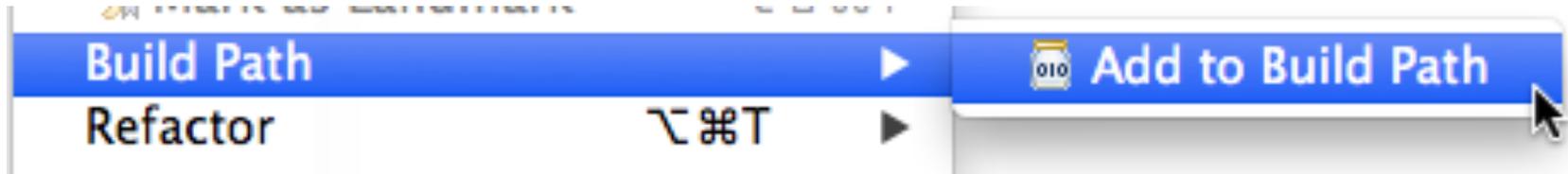
Log4J2-Implementierung in DOTS

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



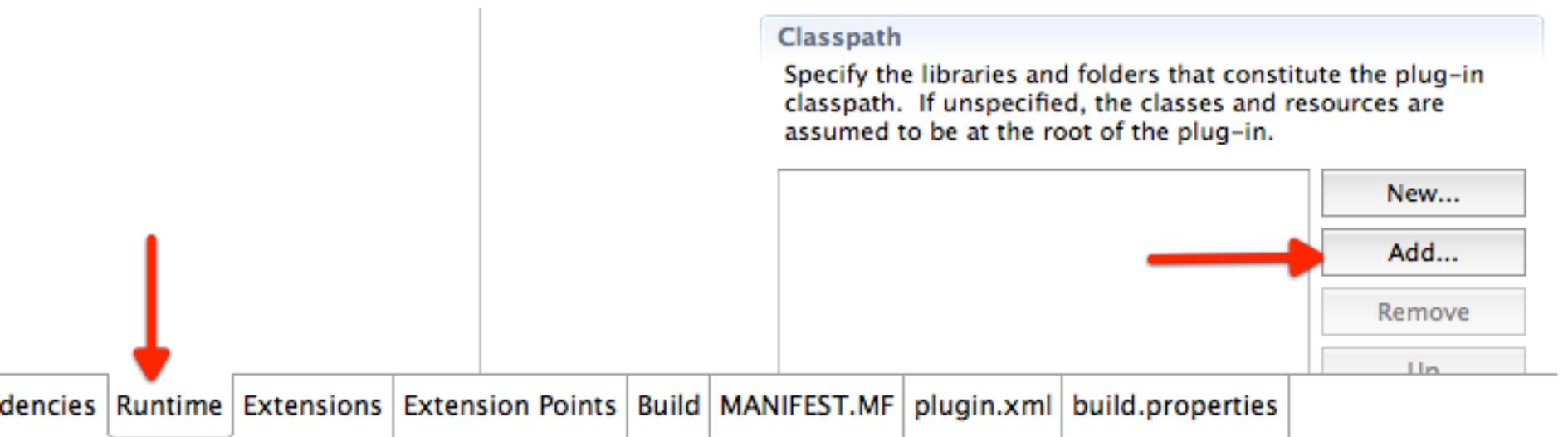
Log4J2-Implementierung in DOTS

- Hinzufügen der Libraries zum Build Path



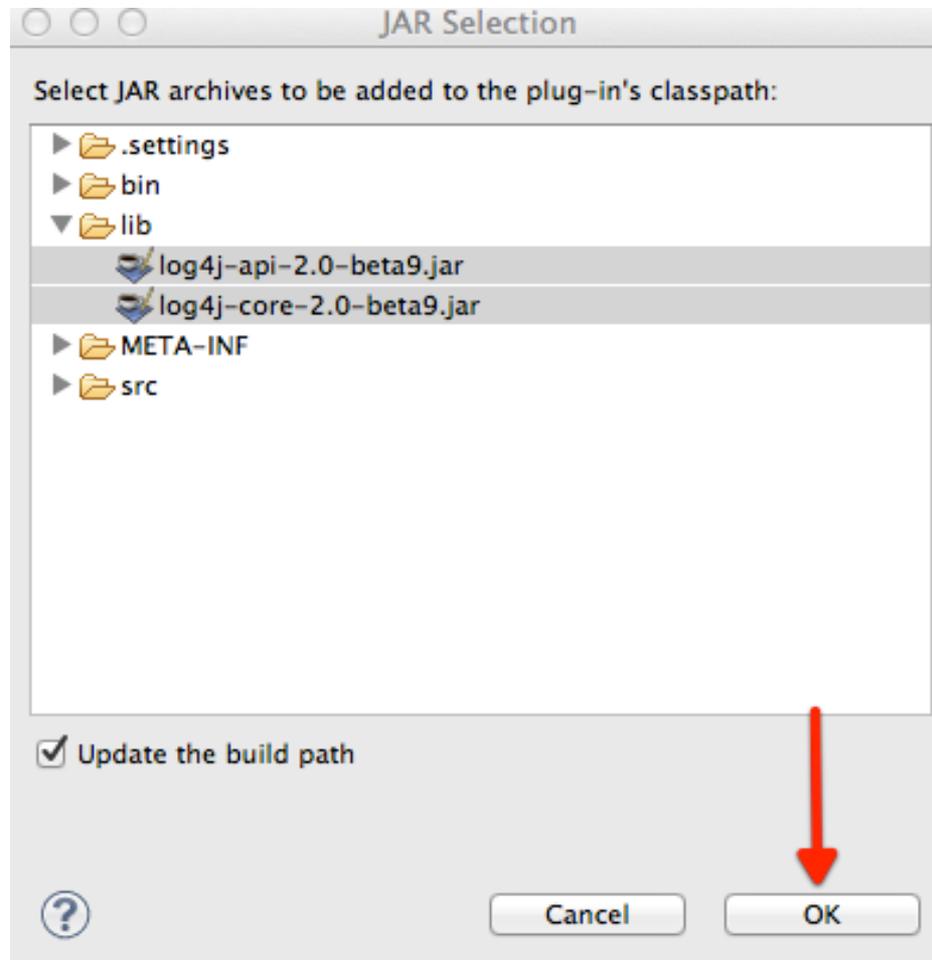
Log4J2-Implementierung in DOTS

- Hinzufügen der Libraries zum Classpath in der MANIFEST.MF



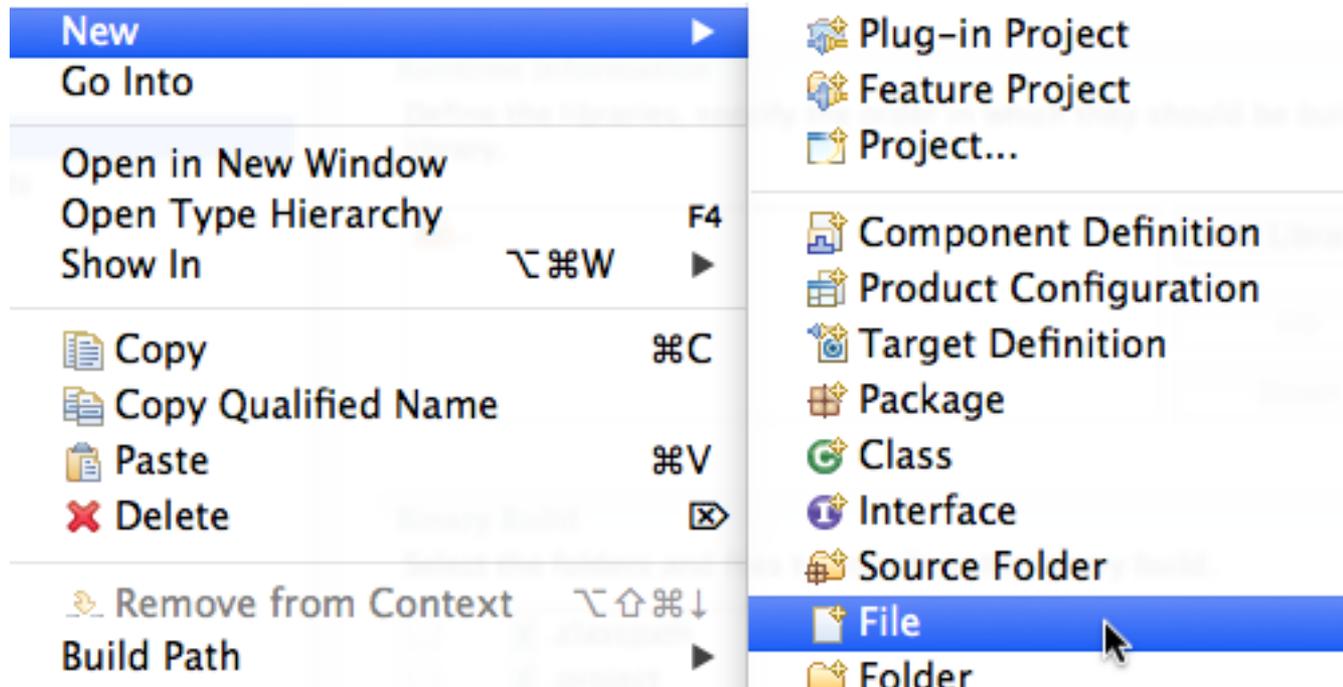
Log4J2-Implementierung in DOTS

- Hinzufügen der Libraries zum Classpath in der MANIFEST.MF



Log4J2-Implementierung in DOTS

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



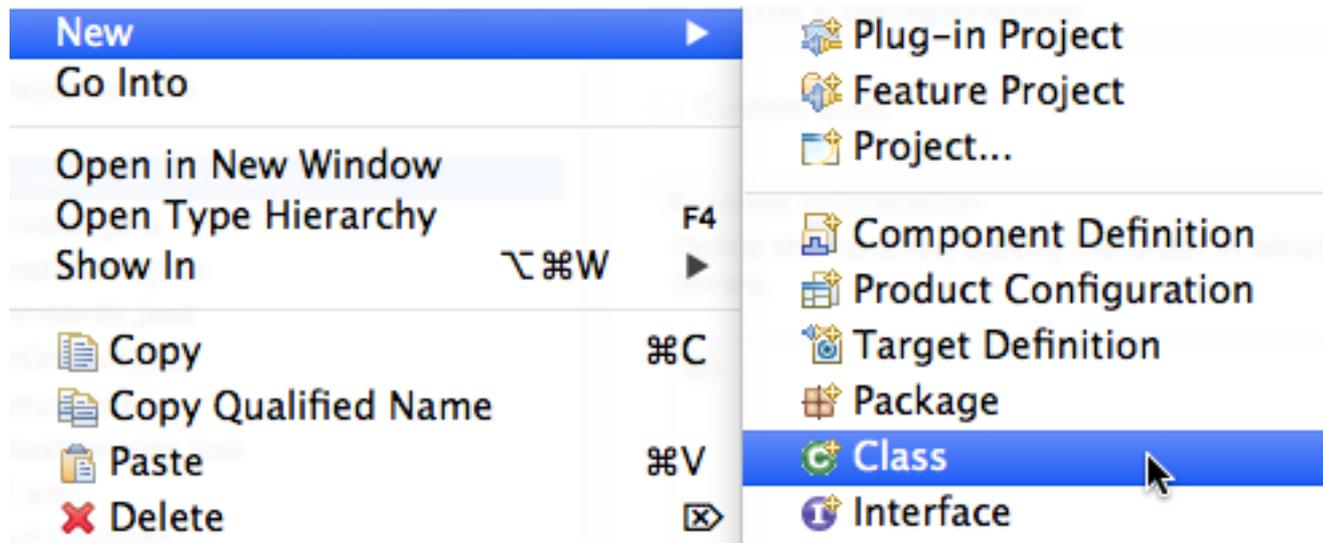
Log4J2-Implementierung in DOTS

- 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

- 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

- Erweiterung des Startup-Tasklet um das Setzen der Java-System-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

- 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 Development
Domino 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

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

Vielen Dank!

