



発表第3回

1

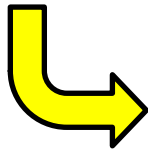
2008MI011 朝倉知也
2008MI079 岩井 大

目次

1. 前回発表の補足
2. OSGi
 - 2.1 OSGi - 概要 -
 - 2.2 OSGi - Bundleの構成 -
 - 2.3 OSGi - Bundleのライフサイクル -
 - 2.4 OSGi - 実装・HelloWorld -
3. 今後の課題
4. 参考文献

1. 前回の発表の補足(1/2)

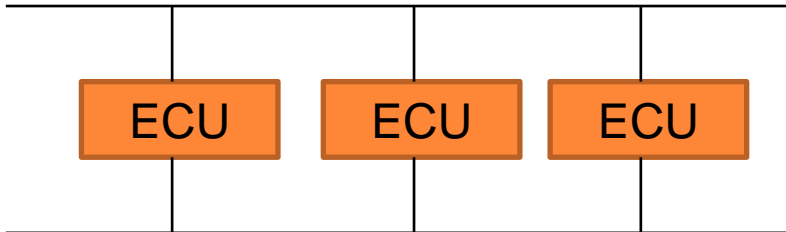
- CAN・・・高速・低速CANを使い分けて幅広い範囲で利用
ケーブルが2本. (利用箇所) ボディ系、パワートレイン系



平衡接続であるため耐ノイズ性を高くしている

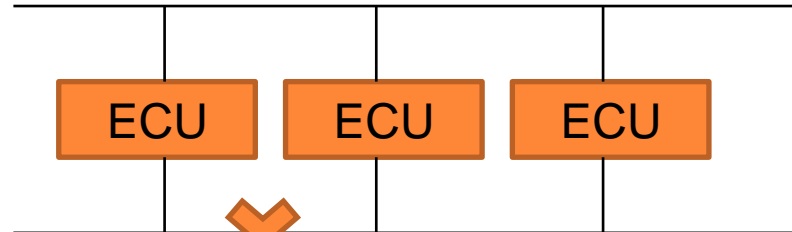
高速CAN

最大1Mbpsの高速通信速度



低速CAN

最大125kbpsの通信速度
1本だけでも通信可能



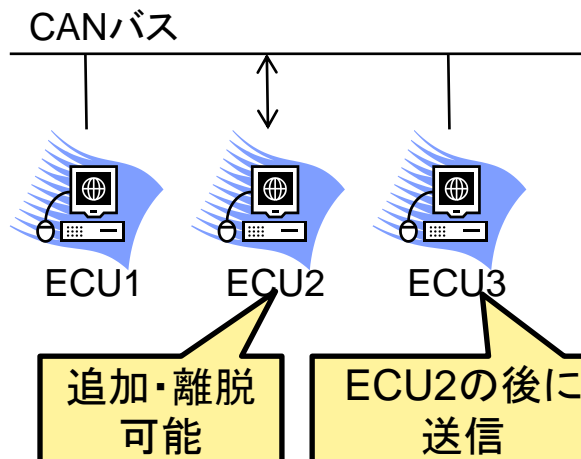
片方が故障しても大丈夫

1. 前回の発表の補足(2/2)

○ CANとLINのプロトコルの違い

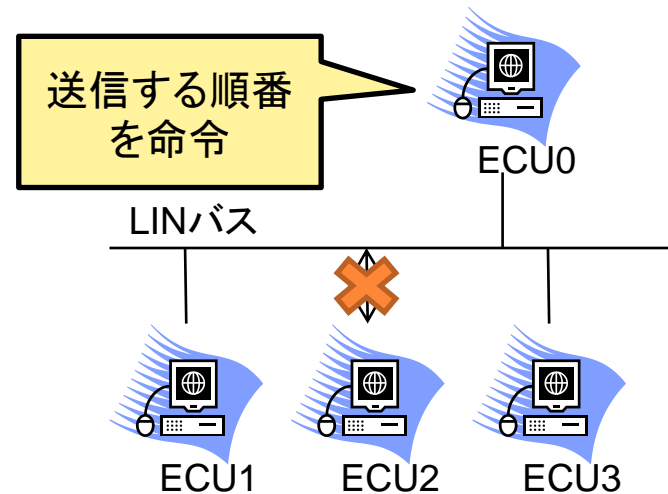
CAN

通信方式はマルチマスタ
各ECUが送信時間を決めることができる
またCSMA/CA方式を採用しており送信の優先順位を決定



LIN

通信方式はシングルマスタ
一つのECUが他を管理
メッセージの衝突はない

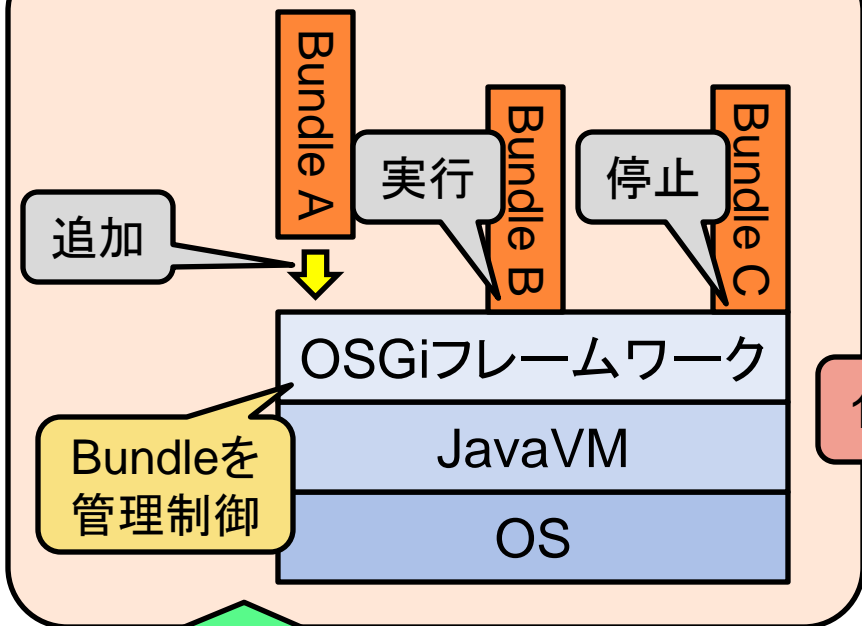


2.1 OSGi - 概要 -

OSGiフレームワーク

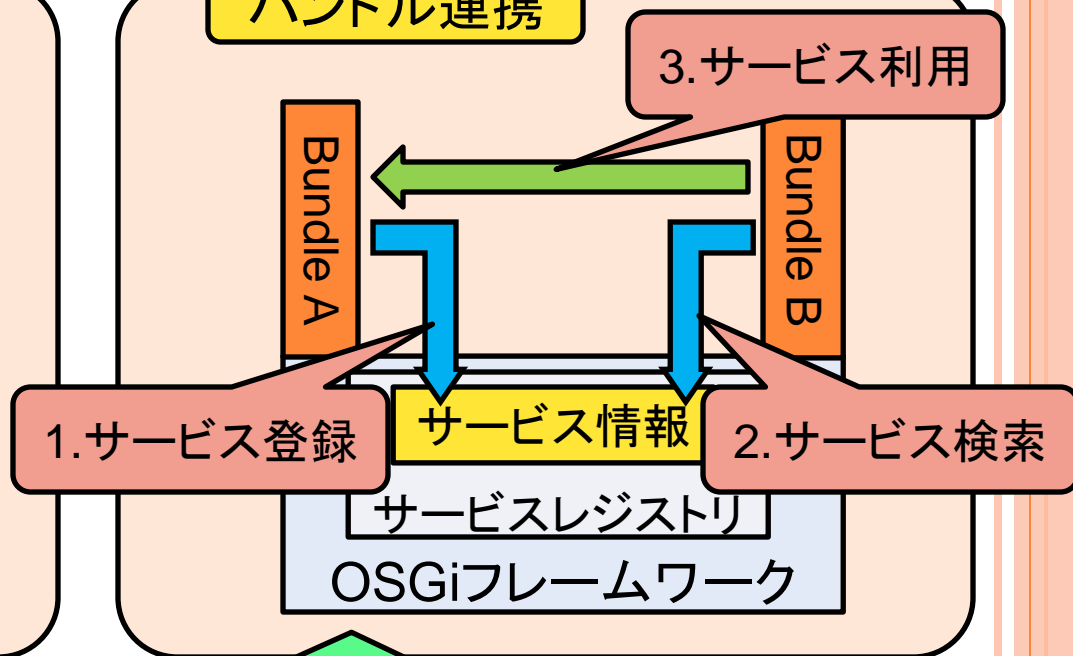
- ・JavaVM上で動作し, Bundle単位で機能を管理
- ・サービスレジストリを使い, Bundle間の連携が可能

OSGiの構成



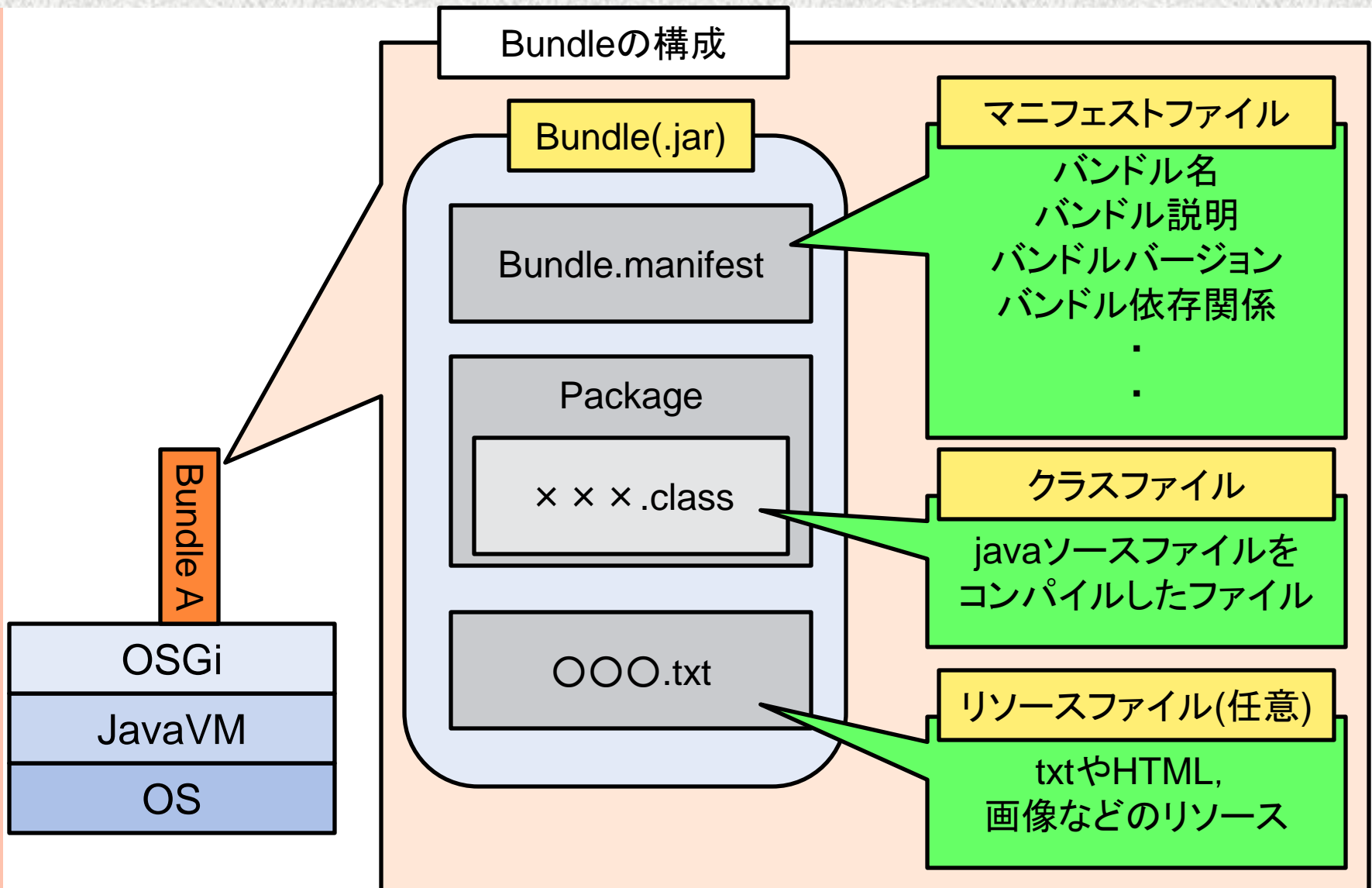
Bundleはプラグ&プレイ方式
ネットワーク経由での管理も可能

バンドル連携



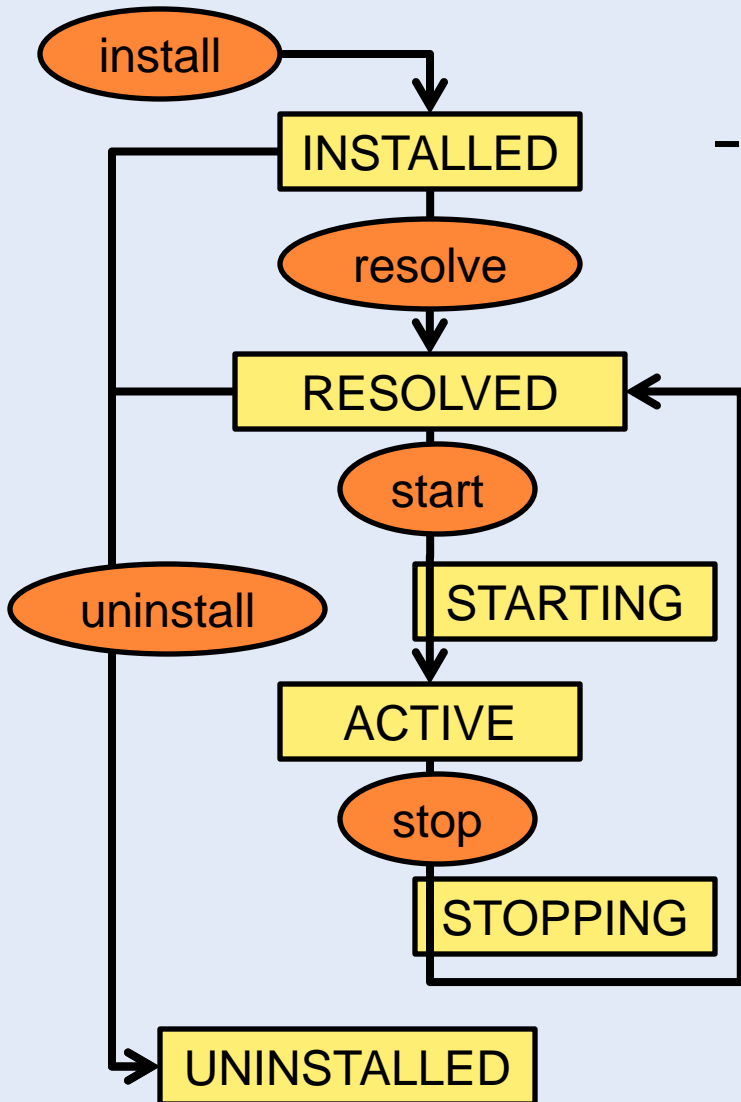
BundleAが登録したサービスを
BundleBが利用できる

2.2 OSGi – Bundleの構成 –

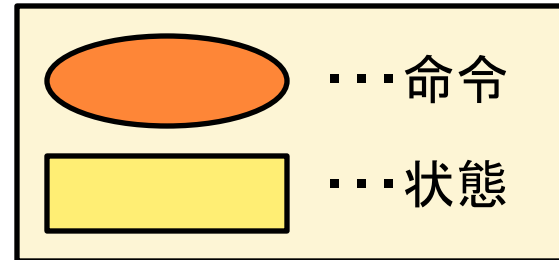


2.3 OSGi – Bundleのライフサイクル –

Bundleのライフサイクル遷移図



遷移図中表記



Bundleの状態と意味

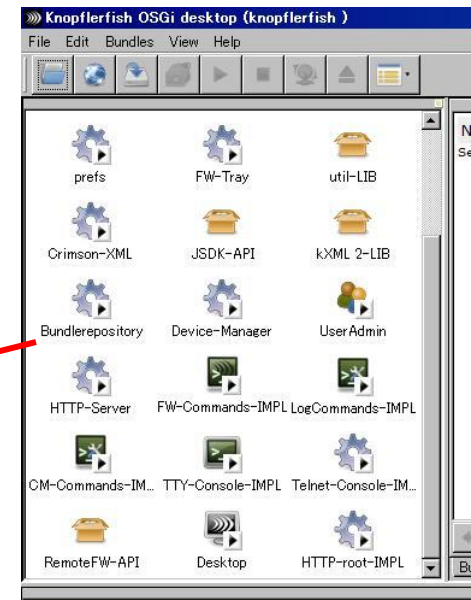
状態	意味
INSTALLED	インストール済
RESOLVED	依存関係解決済
STARTING	起動処理中
ACTIVE	起動中
STOPPING	停止処理中
UNINSTALLED	アンインストール済

2.4 OSGi – 実装・HelloWorld(1/6) –

- 以下の機能を持つBundleを作成する
 - start命令を実行すると「HelloWorld.」を表示
 - stop命令を実行すると「Bye.」を表示
- 作成環境: Eclipse 3.5.0
- 実行環境: Knopflerfish 3.1.0

Knopflerfish … オープンソースのOSGiフレームワーク
(ノプラフィッシュ) BundleをGUIで管理可能

Bundleの状態をGUIにより視覚的に
管理可能



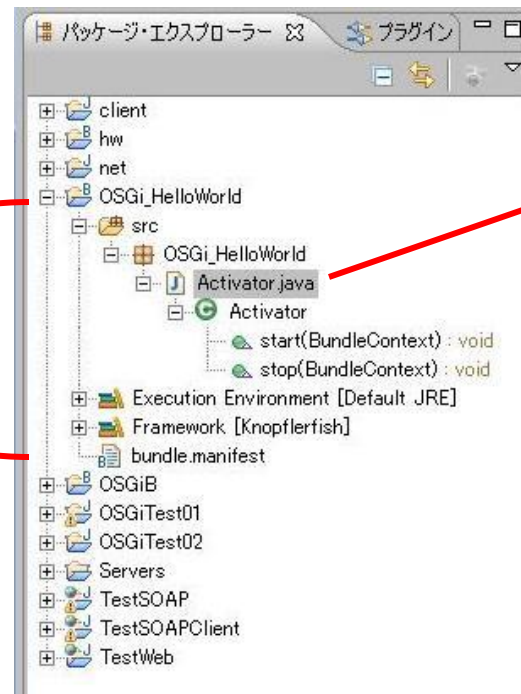
2.4 OSGi – 実装・HelloWorld(2/6) –

1. Eclipseを起動し,

「ファイル → 新規 → その他 → OSGi → BundleProject」
を選択.

プロジェクト名を入力し, Create BundleActivator にチェック,
完了すると新規Bundleプロジェクトが作成される.

「OSGi_HelloWorld」プロジェクトを作成
すると, 自動的にこれらのファイルが
作成される



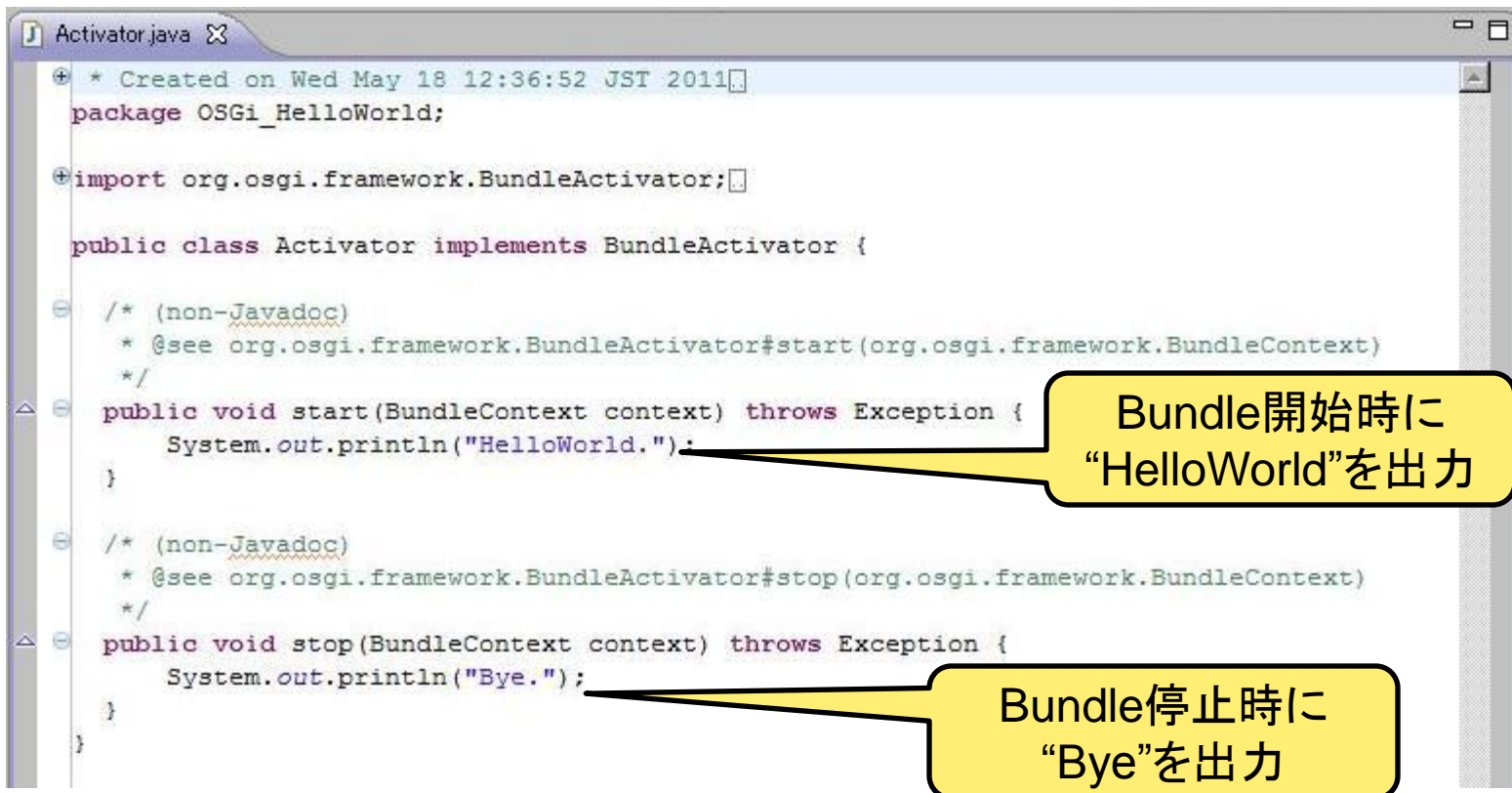
Bundle開始時
に実行される
クラスが記述

2.4 OSGi – 実装・HelloWorld(3/6) –

2. 作成されたActivator.java - Activatorクラスを編集する.

startメソッド … Bundle開始時に実行

stopメソッド … Bundle停止時に実行



```
Activator.java
* Created on Wed May 18 12:36:52 JST 2011
package OSGi_HelloWorld;

import org.osgi.framework.BundleActivator;

public class Activator implements BundleActivator {

    /* (non-Javadoc)
     * @see org.osgi.framework.BundleActivator#start(org.osgi.framework.BundleContext)
     */
    public void start(BundleContext context) throws Exception {
        System.out.println("HelloWorld.");
    }

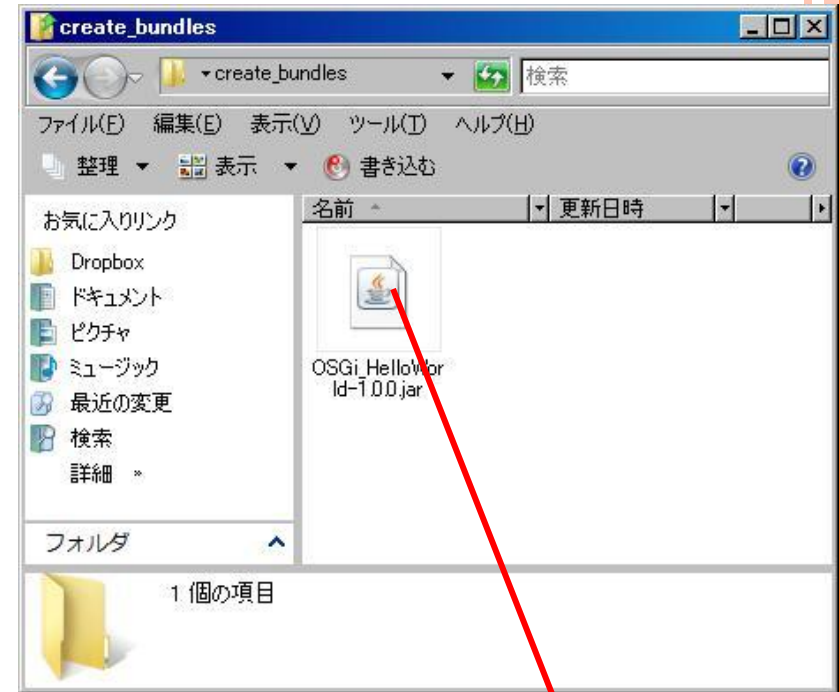
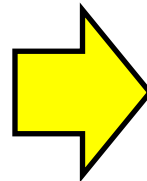
    /* (non-Javadoc)
     * @see org.osgi.framework.BundleActivator#stop(org.osgi.framework.BundleContext)
     */
    public void stop(BundleContext context) throws Exception {
        System.out.println("Bye.");
    }
}
```

Bundle開始時に
“HelloWorld”を出力

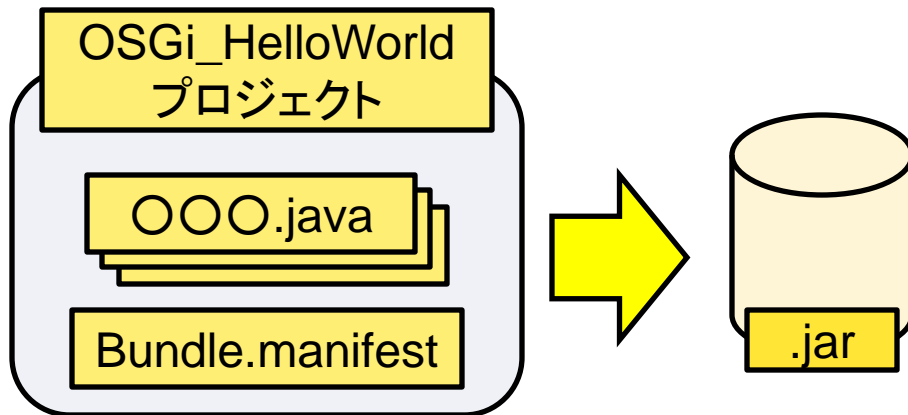
Bundle停止時に
“Bye”を出力

2.4 OSGi – 実装・HelloWorld(4/6) –

3. 編集したOSGi_HelloWorldプロジェクトをエクスポートし、
.jarファイルを作成する。

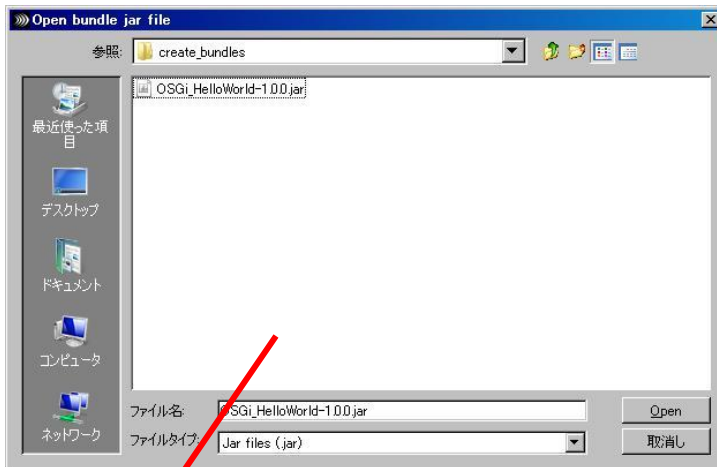


作成された.jarファイル
(Bundle)

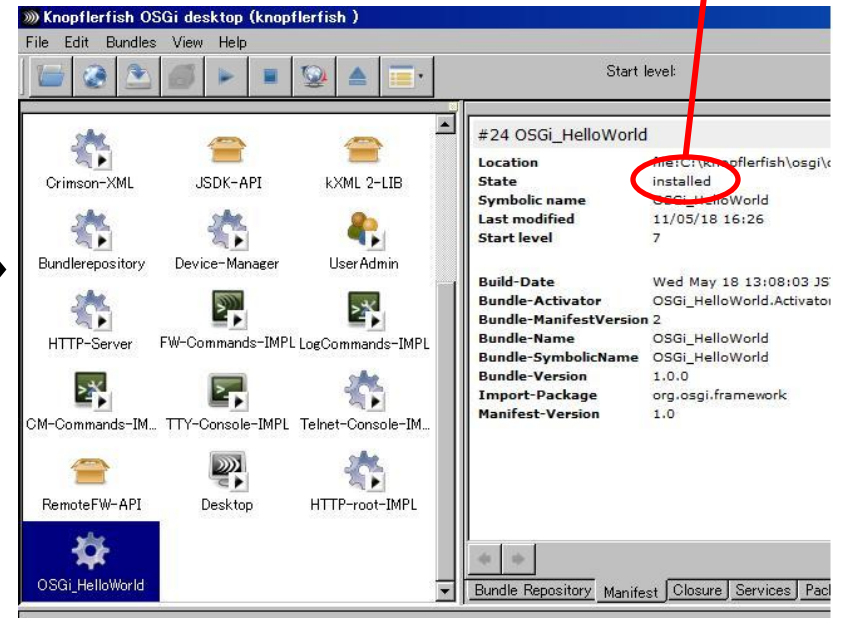


2.4 OSGi – 実装・HelloWorld(5/6) –

4. Knopflerfishを起動し, 作成したBundleをインストールする.



インストールしたBundleの状態が installed になっていることを確認



Knopflerfishでは, Bundleを開くだけでインストールが完了する

2.4 OSGi – 実装・HelloWorld(6/6) –

5. インストール完了したBundleに対しstart命令とstop命令を実行し、動作を確認する。

start 命令を実行し、
HelloWorld.が表示された

stop 命令を実行し、
Bye.が表示された

Knopflerfish コンソール

```
> start OSGi_HelloWorld
[stdout] HelloWorld.
Started: OSGi_HelloWorld (#24)
> stop OSGi_HelloWorld
[stdout] Bye.
Stopped: OSGi_HelloWorld (#24)
>
>
```

3. 今後の課題

- OSGiの実装を進める.
- 今回までの発表では自動車を対象にしたOSGiについて調べたが、別の観点での利用例を調べたい.

4. 参考文献

- 技術情報 株式会社ウィッツ
(<http://www.witz-inc.co.jp/technical/>)
- モノづくりスペシャリストのための情報ポータル - @IT MONOist
(<http://monoist.atmarkit.co.jp/>)
- システムクリエイト株式会社 OSGi入門
(<http://www.s-cre.co.jp/web/osgi/osgi.php?url=1>)
- 濱千代正弥、片桐雅仁：自動車ネットワークサービスの連携アーキテクチャ