IBM SWG Kraków Lab

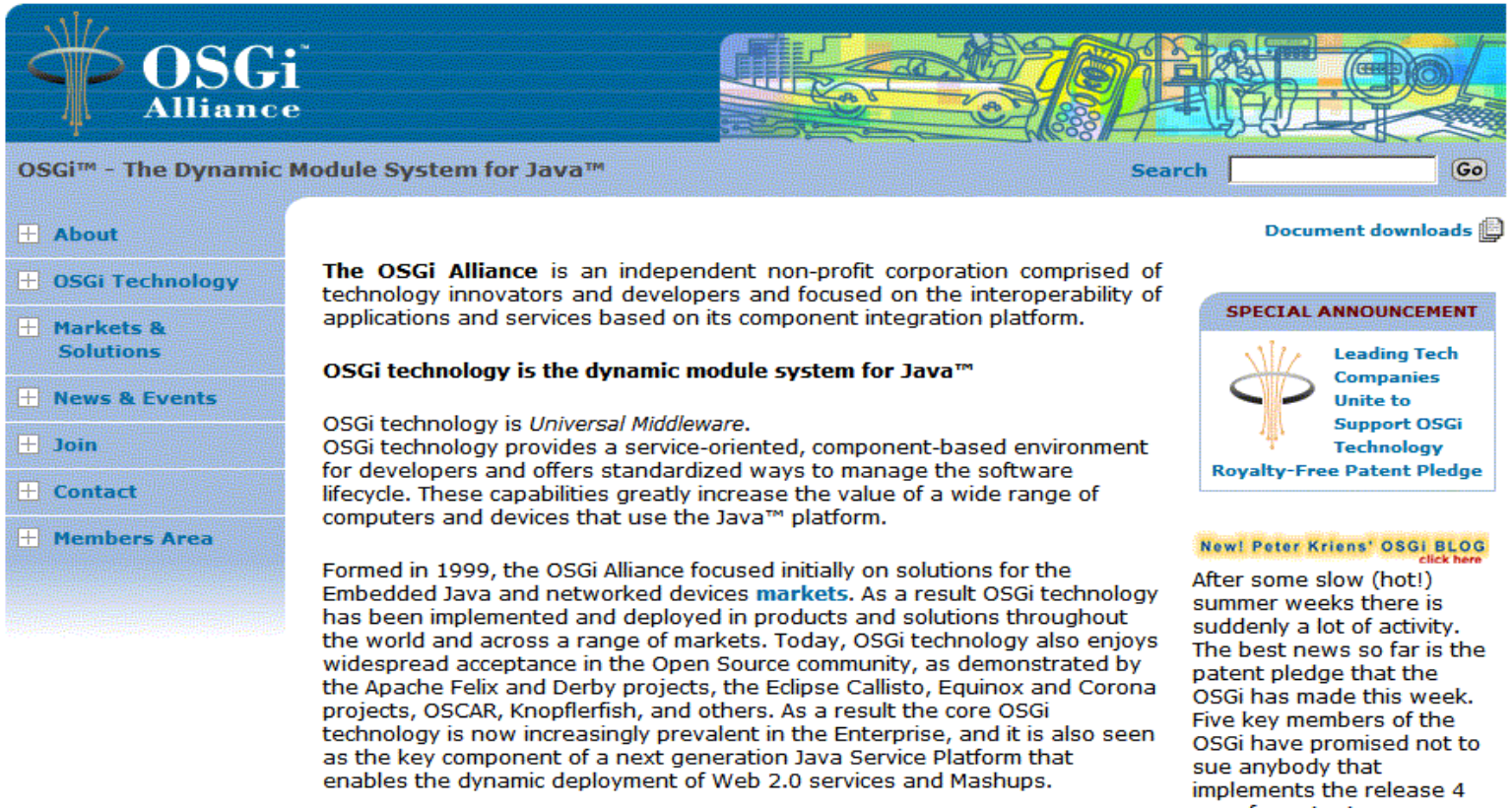
# OSGi

## Wiele usług pod jednym dachem

# OSGi Alliance

- **Organizacja non-profit skupiona wokół uniwersalnej warstwy usług (universal middleware)**
- **Celem działania: stworzenie uniwersalnej platformy**
- **Członkowie: firmy IT, producenci sprzętu telekomunikacyjnego, koncerny motoryzacyjne, producenci sprzętu AGD/RTV**

# osgi.org



# Cechy platformy OSGi

- **Wykonywanie wielu modułów aplikacji w tej samej wirtualnej maszynie Javy**
- **Współdzielenie klas/interfejsów przez moduły**
- **Instalowanie i aktualizacje modułów bez konieczności restartu aplikacji**
- **Zdalne zarządzanie modułami**

# Korzyści ze stosowania OSGi

- **Producenci oprogramowania**
  - Dostęp do szerszego grona klientów
  - Obniżenie kosztów wytworzenia, testowania i utrzymywania oprogramowania
  - Znaczne ograniczenie czasu wprowadzania produktu na rynek

# Korzyści ze stosowania OSGi

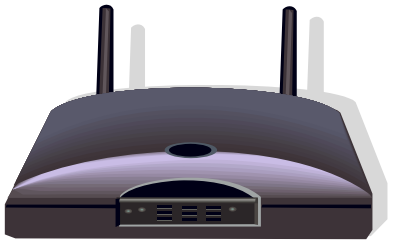
- **Producenci urządzeń**
  - Łatwiejszy wybór dostawcy oprogramowania
  - Łatwość aktualizacji oprogramowania
  - Możlliwość rozszerzenia funkcjonalności produktu nawet po wprowadzeniu na rynek
  - Ułatwiony serwis produktów
  - Niewielkie wymagania wobec sprzętu

# Korzyści ze stosowania OSGi

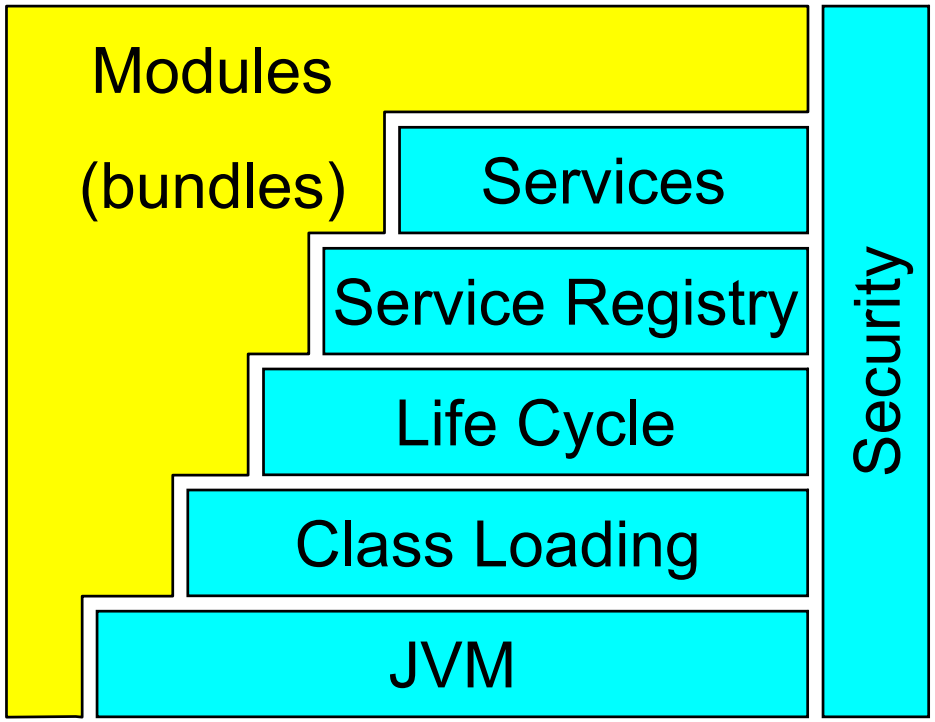
- **Użytkownicy**
  - Niższy koszt zakupu produktów i/lub wyższa jakość
  - Ochrona inwestycji
  - Możliwość integracji zakupionych produktów
    - Nowa, niespotykana wcześniej funkcjonalność
  - Szybszy, prostszy i tańszy serwis

# Praktyczne zastosowania OSGi

- **Aplikacje złożone z modułów**
- **Urządzenia przenośne**
- **Sprzęt RTV/AGD**
- **Motoryzacja**
- **Budynki inteligentne**
- **Urządzenia sieciowe**



# Architektura




- **Warstwy**
  - Security Layer
  - Life Cycle Layer
  - Module Layer
  - Service Layer
  - Actual Services

# Module Layer

- **Moduł = bundle**
- **Funkcjonowanie modułów w środowisku OSGi**
  - Identyfikacja i wersjonowanie modułów
  - Eksportowanie i importowanie pakietów przez moduły
  - Zależności międzymodułowe
  - Opis modułu umieszczany w deskryptorze (plik tekstowy)

# Module Layer – sposób budowania modułów

- **Dostarczane w postaci plików JAR**
- **Deskryptorem jest plik META-INF/MANIFEST.MF**
  - Bundle-SymbolicName: jednoznacznie identyfikuje moduł
  - Bundle-Version: wersja modułu – w środowisku wykonawczym można zainstalować kilka wersji modułu
  - Import-Package: wykorzystywane w module pakiety innych modułów
  - Export Package: pakiety udostępniane innym modułom
  - Bundle-Activator: klasa aktywująca moduł

# Life Cycle Layer

- **BundleActivator**
  - Moduł opcjonalnie dostarcza klasę implementującą zadeklarowaną w deskryptorze
    - Moduły nie posiadające klasy BundleActivator nazywa się bibliotecznymi
  - Moduł jest informowany o przejściu w inny stan przez wywołania metod
    - start(BundleContext context)
      - Rejestracja usług, uruchomienie wątków, itp.
    - stop(BundleContext context)
      - Wyrejestrowanie usług, zatrzymanie wątków.

# Life Cycle Layer

- **BundleContext**
  - Stanowi API modułu do środowiska OSGi
    - Zarządzanie modułami: installBundle, getBundle, ...
    - Zarządzanie usługami: registerService, getService, ungetService, getServiceReference, ...
- **BundleListener**
  - Otrzymuje zdarzenia BundleEvent związane z cyklem życia modułów: INSTALLED, STARTED, STOPPED, UPDATED, UNINSTALLED, RESOLVED, UNRESOLVED, STARTING, STOPPING

# Life Cycle Layer

- **Bundle**
  - Reprezentuje moduł zainstalowany w środowisku
    - Zarządzanie cyklem życia: start, stop, getState
    - Deinstalacja: update, uninstall
    - Zarządzanie usługami: getRegisteredServices, getServicesInUse, ...
    - Dostęp do zasobów modułu: loadClass, getResources, ...
    - Dostęp do szczegółowych danych moudłu: getBundleId, getHeaders, getLocation, ...

# Service Layer

- **Moduł może rejestrować usługi i korzystać z usług innych modułów**
- **Usługa posiada interfejs(y) i implementację**
  - Zwykle w różnych pakietach

```
public interface HelloService {
    public abstract String hello(String name);
}
```

```
public class HelloServiceImpl implements HelloService {
    public String hello(String name) {
        return "Hello, " + name;
    }
}
```

# Service Layer – rejestracja usług

- **Usługa zaimplementowana jako zwykła klasa**
- **Listę metod udostępnianych przez usługę definiują interfejsy**
- **Informacja o zarejestrowanej usłudze przechowywana w obiekcie ServiceRegistration**

```
HelloService service = new HelloServiceImpl();
ServiceRegistration registration =
        bundleContext.registerService(HelloService.class.getName(),
                service, new Hashtable());
// ...
registration.unregister();
```

# Service Layer – korzystanie z usług

- **Moduł rejestrujący eksportuje pakiet z interfejsami**
- **Moduł kliencki**
  - Importuje pakiet z interfejsami
  - Wyszukuje usługę
  - Wywołuje metody „biznesowe”

```
ServiceReference reference = bundleContext.getServiceReference(
        HelloService.class.getName());
HelloService service = (HelloService)
        bundleContext.getService(reference);
service.hello("Marcin");
bundleContext.ungetService(reference);
```

# Service Layer – dynamiczne korzystanie z usług

- **ServiceListener otrzymuje informacje o zdarzeniach ServiceEvent: REGISTERED, MODIFIED, UNREGISTERING**
- **Jeszcze łatwiej posłużyć się klasą ServiceTracker**
  - Utworzenie obiektu ServiceTracker
  - Dostarczenie obiektu ServiceTrackerCustomizer
    - Metody: addingService, modifiedService, removedService

# Service Layer – ServiceTrackerCustomizer

```
public Object addingService(ServiceReference reference) {
    HelloService helloService =
            (HelloService) context
                    .getService(reference);
    return helloService;
}
public void modifiedService(ServiceReference reference, Object
service) {
}
public void removedService(ServiceReference reference, Object service)
{
}
```

# Service Layer – ServiceTracker

```
ServiceTracker serviceTracker = new ServiceTracker(
        HelloService.class.getName(),
        new HelloServiceTrackerCustomizer());
serviceTracker.open();
// ...
serviceTracker.close();
```

# Security Layer

- **Bazuje na Java2 Security**
  - Podpisywanie modułów (jarsigner)
- **Specyficzne rodzaje uprawnień**
  - Dostęp do zasobów modułu
    - PackagePermission, ResourcePermission
  - Zarządzanie modułami
    - AdminPermission
  - Dostęp do usług
    - ServicePermission
- **Warstwa bezpieczeństwa nie jest wymagana przez specyfikację**
  - Może niepotrzebnie podnosić wymagania sprzętowe

# Kto stosuje OSGi

- **Eclipse – zawiera środowisko Equinox**
  - Możliwa jest zmiana konfiguracji Eclipse bez restartu środowiska
- **Nokia – telefony nowej generacji**
  - Telefony posiadające możliwość zdalnego zarządzania aplikacjami
- **BMW – obsługa urządzeń elektronicznych znajdujących się w samochodach serii 5**
- **Siemens – oprogramowanie sterujące urządzeniami RTV/AGD**
- **I wielu innych...**

# Implementacje OSGi

- **Equinox –** http://www.eclipse.org/
  - Implementacja Open Source, licencja CPL
  - Certyfikat OSGi Alliance
- **Knopplerfish –** http://www.knopplerfish.org/
  - Implementacja Open Source, dostępna również licencja komercyjna
  - Certyfikat OSGi Alliance
- **Felix –** http://incubator.apache.org/felix/
  - Implementacja Open Source, licencja APL

# Przyszłość OSGi

- **Włączenie OSGi do stosu standardów Javy**
  - JSR 232: Mobile Operational Management
  - JSR 291: Dynamic Component Support for Java SE
- **Adaptacja platformy OSGi przez producentów oprogramowania**
  - Spring Framework

# Trademarks and Disclaimers



The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States,other countries, or both:

| | | | | |
|---|---|---|---|---|
| IBM | iSeries | OS/400 | Informix | WebSphere |
| IBM(logo) | pSeries | AIX | Cloudscape | MQSeries |
| e(logo)business | xSeries | CICS | DB2 Universal Database | DB2 |
| Tivoli | zSeries | OS/390 | IMS | Lotus |

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in theUnited States, other countries, or both. Microsoft, Windows, Windows NT, andthe Windows logo are registered trademarks of Microsoft Corporation in the United States, other countries, or both. Intel, ActionMedia, LANDesk, MMX, Pentium and ProShare aretrademarks of Intel Corporation in the United States, other countries, or both. UNIX is a registered trademark of The Open Group in the United States and other countries. Linux is a registered trademark of Linus Torvalds. Other company, product and service names may be trademarks or service marks of others.

Product data has been reviewed for accuracy as of the date of initial publication. Product data is subject to change without notice. This document could include technical inaccuracies or typographical errors. IBM may make improvements and/or changes in the product(s) and/or program(s) described herein at any time without notice. Any statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business. Any referenceto an IBM Program Product in this document is not intended to state or imply that only that program product may be used. Any functionally equivalent program, that does not infringe IBM's intellectually property rights, may be used instead.

Information is provided "AS IS" without warranty of any kind. THE INFORMATION PROVIDED IN THIS DOCUMENT IS DISTRIBUTED "AS IS" WITHOUT ANY WARRANTY, EITHER EXPRESS OR IMPLIED. IBM EXPRESSLY DISCLAIMS ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT. IBM shall have no responsibility to update this information. IBM products are warranted, if at all, according to the terms and conditions of the agreements (e.g., IBM Customer Agreement, Statement of Limited Warranty, International Program License Agreement, etc.) under which they are provided. Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products in connection with this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. IBM makes no representations or warranties, express or implied, regarding non-IBM products and services.

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents or copyrights. Inquiries regarding patent or copyright licenses should be made, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. The actual throughput orperformance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performanceimprovements equivalent to the ratios stated here.

Note to U.S. Government Users -Documentation related to restricted rights-Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract and IBM Corp.