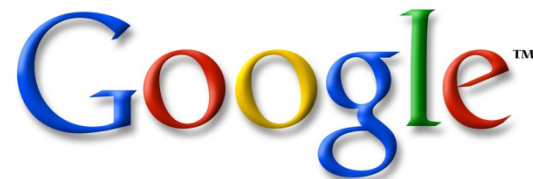


Programowanie urządzeń mobilnych w systemie Android

dr inż. Jacek Czerniak

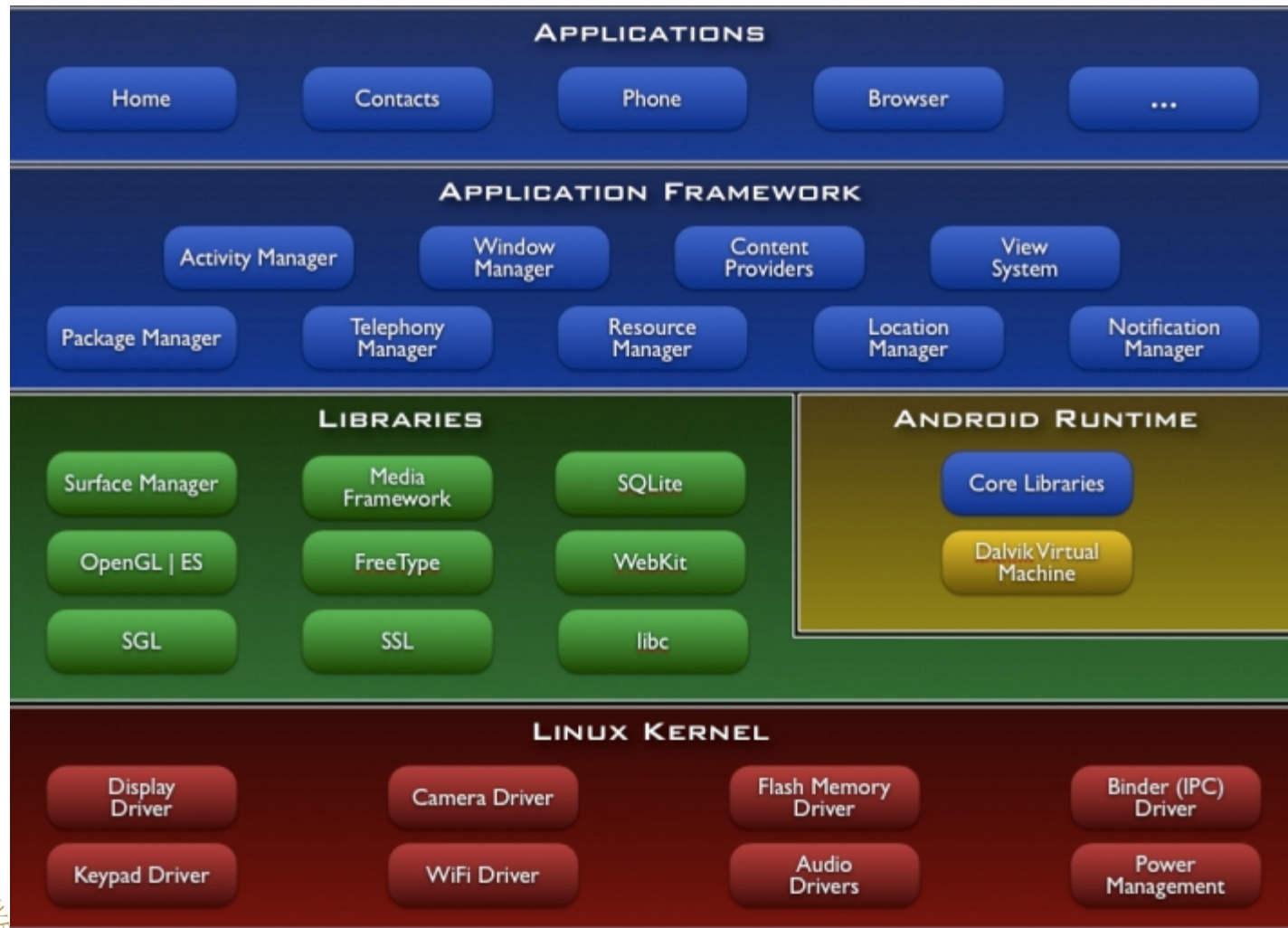
Zakład Informatyki

jczerniak@ukw.edu.pl





Architektura



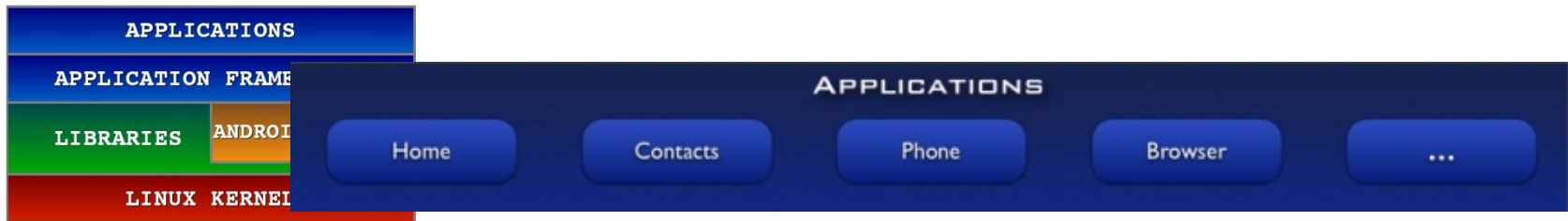


Warstwy systemu Android

- ❑ **Linux Kernel Layer** – warstwa jądra systemu operacyjnego oparta na Linuxie
- ❑ **Native Libraries Layer** – natywne biblioteki systemu Android
- ❑ **Application Framework Layer** – warstwa, z którą bezpośrednio komunikują się aplikacje min. zarządzanie oknami, zasobami itp.
- ❑ **Application Layer** – warstwa aplikacji, z którą w interakcję wchodzi użytkownicy aby np. wykonać połączenie telefoniczne



Application Layer



- Android wprowadza pewien rdzeń aplikacji:
 - ✓ Email Client
 - ✓ SMS Program
 - ✓ Calendar
 - ✓ Maps
 - ✓ Browser
 - ✓ Contacts
 - ✓ Etc

- Wszystkie aplikacje są napisane w języku Java.♪



Application Framework Layer



- ❑ **Activity Manager** – zarządza cyklem życia aktywności aplikacji. Opisany jest bardziej szczegółowo w artykule „Architektura systemu Android: Activity„
- ❑ **Content Providers** – zarządza wymianą danych pomiędzy aplikacjami.
- ❑ **Telephony Manager** – zarządza wszystkimi rozmowami głosowymi. Używany kiedy potrzebny jest dostęp do funkcji połączeń głosowych z aplikacji.
- ❑ **Location Manager** – zarządzanie lokalizacją urządzenia przy użyciu GPS lub nadajników komórkowych.
- ❑ **Resource Manager** – zarządza różnymi rodzajami zasobów, których używa aplikacja.

Android S/W Stack – App Framework (Cont)♪



□ Funkcjonalności

♪ Moduł ♪	Zastosowanie ♪
View System♪	Used to build an application, including lists, grids, text boxes, buttons, and embedded web browser♪
Content Provider♪	Enabling applications to access data from other applications or to share their own data♪
Resource Manager♪	Providing access to non-code resources (localized strings, graphics, and layout files)♪
Notification Manager♪	Enabling all applications to display customer alerts in the status bar♪
Activity Manager♪	Managing the lifecycle of applications and providing a common navigation backstack♪



Native Libraries



- ❑ **Surface Manager** – biblioteka odpowiedzialna za tworzenie okien na ekranie
- ❑ **SGL** – biblioteka odpowiedzialna za wyświetlanie grafiki 2D
- ❑ **Open GL|ES** – biblioteka odpowiedzialna za generowanie grafiki 3D
- ❑ **Media Framework** – biblioteka wspiera odtwarzanie i nagrywanie różnych formatów obrazu, audio i wideo



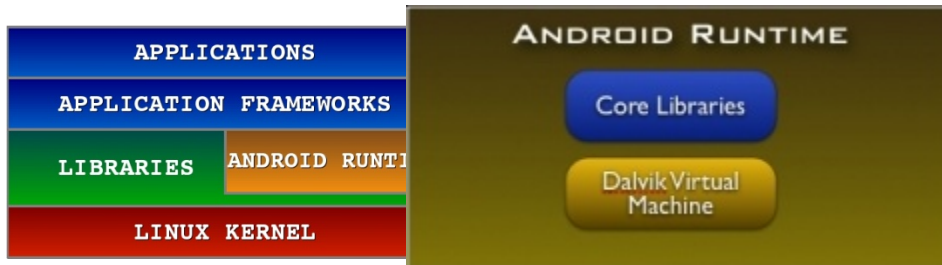
Native Libraries c.d.♪



- ❑ **FreeType** – biblioteka odpowiedzialna za generowanie i renderowanie fontów
- ❑ **WebKit** – biblioteka silnika przeglądarki internetowej
- ❑ **libc** – biblioteka standardowa języka C
- ❑ **SQLite** – biblioteka obsługi baz danych SQL (Android wykorzystuje silnik SQLite dla swoich potrzeb)
- ❑ **OpenSSL** – biblioteka wspierająca szyfrowanie SSL/TLS



Runtime Layer



Na tym samym poziomie co warstwa bibliotek ulokowana jest również warstwa wykonawcza (Android runtime layer) zawierająca również zestaw podstawowych bibliotek Java. Ta warstwa obejmuje także wirtualną maszynę Dalvik'a (Dalvik Virtual Machine).APIs

- Data Structures
- Utilities
- File Access
- Network Access
- Graphics
- Etc



Dalvik Virtual Machine

- Dalvik jest oprogramowaniem typu open-source powstałym na szkieletach maszyny Apache Harmony. Jego autorem jest Dan Bornstein, który nadał mu nazwę po wiosce rybackiej Dalvík w Eyjafjörður w Islandii, skąd pochodzą jego przodkowie♪





Podstawowe cechy Dalvik'a:

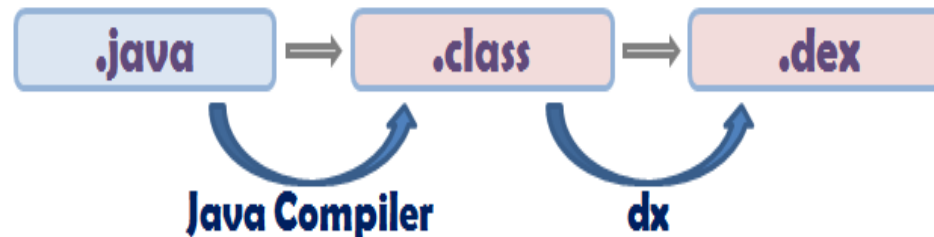
- ❑ Jest maszyną wirtualną operującą na rejestrach (w przeciwieństwie do Java VM, która operuje na stosie)
- ❑ Jest zoptymalizowany do niskich wymagań dotyczących pamięci
- ❑ Został zaprojektowany w sposób umożliwiający uruchomienie wielu instancji wirtualnych maszyn na raz
- ❑ Opiera się na mechanizmach kernela Linuxa do izolacji procesów, zarządzania pamięcią i obsługi wątków
- ❑ Ze względu na brak możliwości wykonywania bezpośrednio skompilowanych klas Java, operuje na plikach .dex



Android S/W Stack – Runtime (c.d.) ♪

□ Dalvik Virtual Machine (c.d.)

- ✓ Pliki wykonywalne w Dalvik (.dex)
 - .dex format jest zoptymalizowany do niskiego zużycia pamięci.
 - kompilacja



Opierając się na jądrze Linux:

- ✓ Wielowątkowość
- ✓ Niskopoziomowe zarządzanie pamięcią ♪



Linux Kernel Layer



- **Hardware Abstraction** – pierwsza warstwa abstrakcji pośrednicząca w komunikacji pomiędzy sprzętem a kernelem systemu operacyjnego
- **Memory Management Programs** – zarządzanie pamięcią w systemie
- **Security Settings** – ustawienia zabezpieczeń systemu
- **Power Management Software** – zarządzanie zasilaniem urządzenia
- **Other Hardware Drivers** – inne sterowniki sprzętowe
- **Support for Shared Libraries** – wsparcie dla bibliotek współdzielonych
- **Network Stack** – stos sieciowy systemu



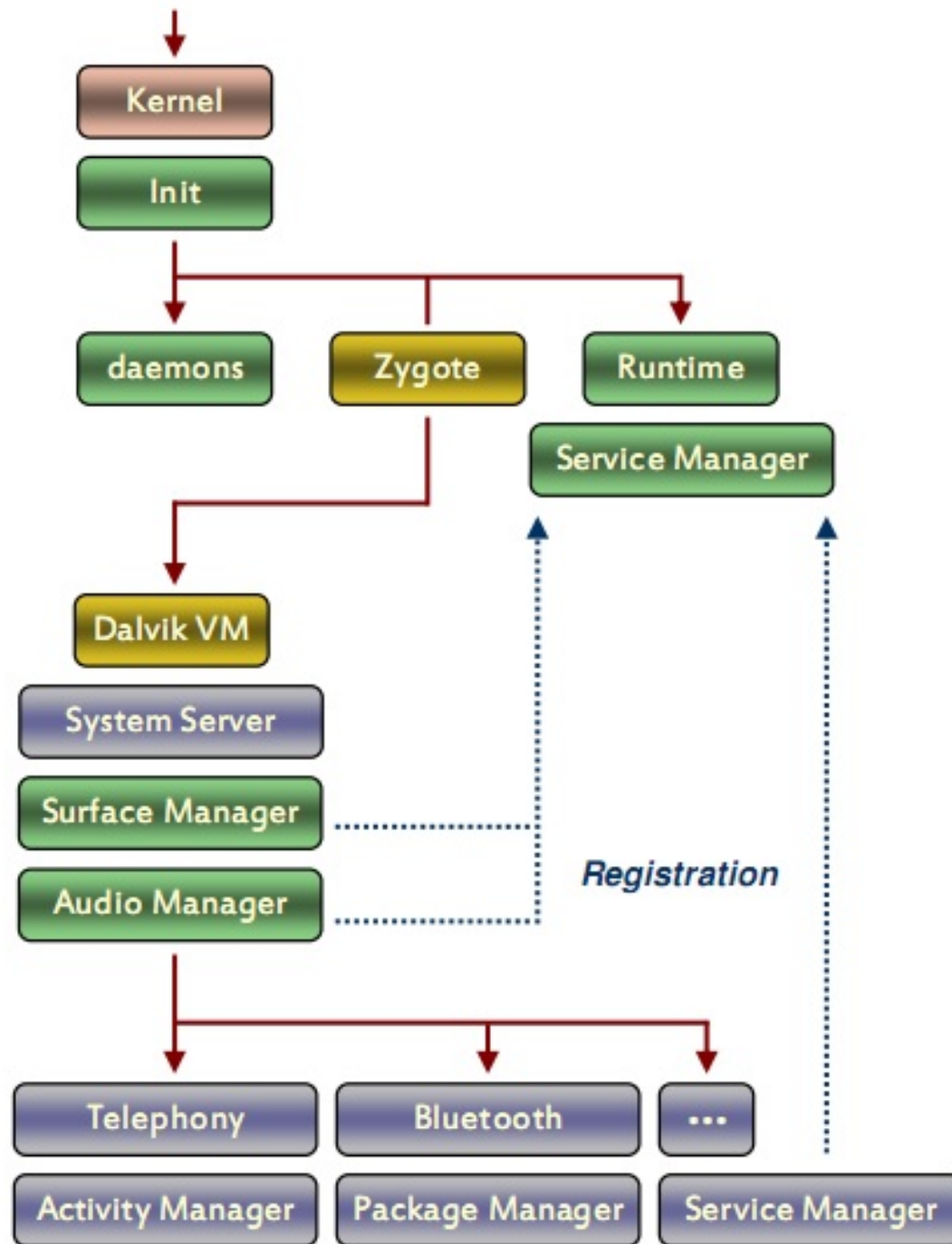
Wersje jądra

Android Version	Linux Kernel Version
1.0	2.6.25
1.5 (Cupcake)	2.6.27
1.6 (Donut)	2.6.29
2.2 (Froyo)	2.6.32
2.3 (Gingerbread)	2.6.35
3.0 (Honeycomb)	2.6.36
4.0.x (Ice Cream Sandwich)	3.0.1
4.1./4.2 (Jelly Bean)	3.0.31



Proces boot'owania systemu

- Kernel Linuxa
- Init
- Procesy systemowe (usb, adb, radio itp.)
- Zygote – Pierwsza, nadrzędna maszyna wirtualna Dalvik
- System Server – w oddzielnej maszynie Dalvik
- Komponenty takie jak Activity Manager uruchamiane są przez System Server
- Surface Manager i Audio Manager



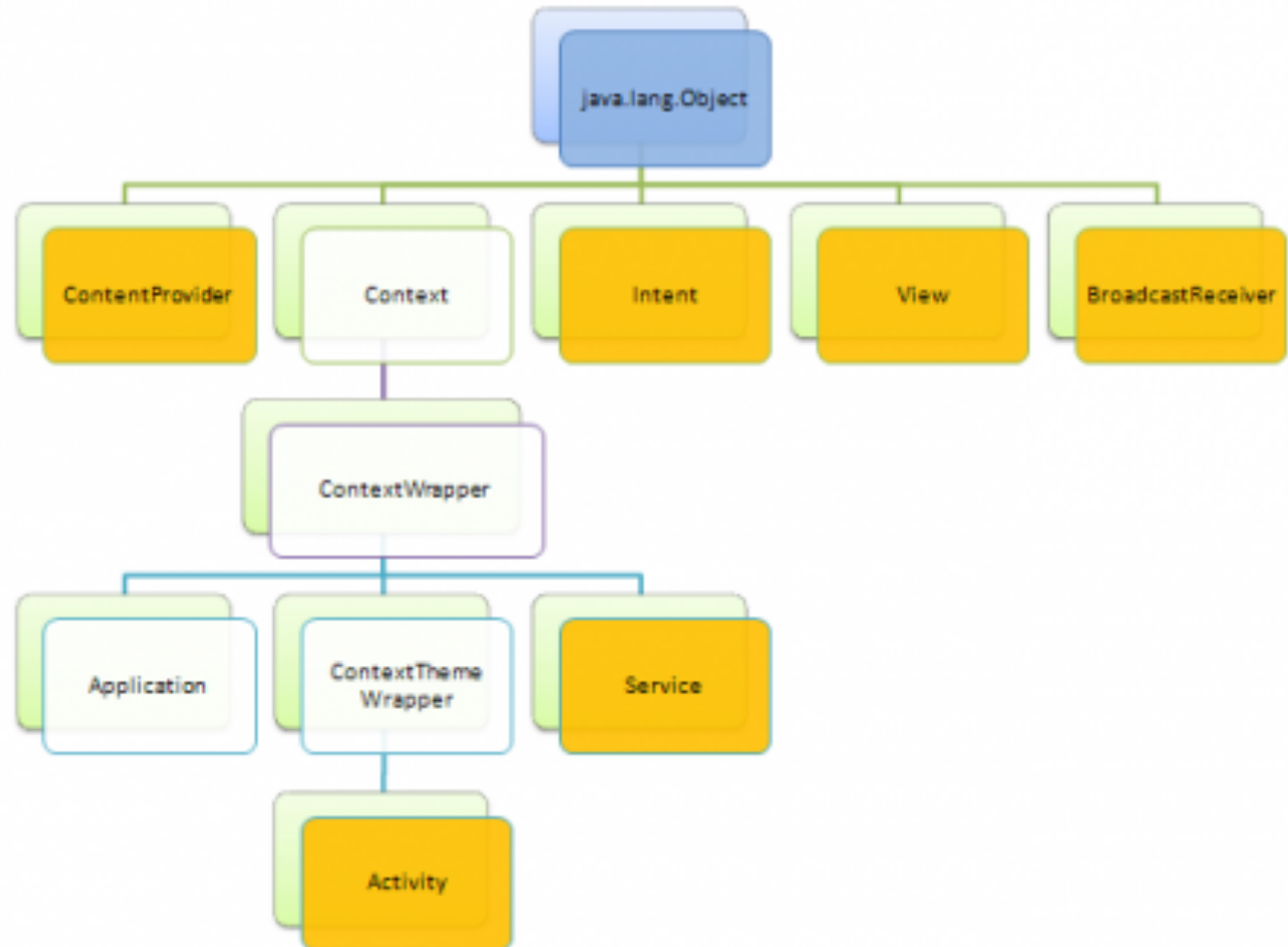
Podstawowe elementy strukturalne stanowią:



- Klasy Java
- Manifest
- Zasoby (Resources)
- Pliki (Files)



Klasy Java





Klasa View

- Klasa View stanowi klasę bazową dla wszystkich widgetów GUI (Button, RadioButton, TextView, itd...) zwanych Widokami (Views). Nie należy przy tym mylić aplikacji typu widget z widgetami GUI. Te pierwsze stanowią miniaturowe aplikacje, które mogą być włączane w inne (np. aplikacje dla Home screen'a).



Klasa Activity

- Aktywności (Activities) są najczęściej stosowane, gdyż aktywność to jeden ekran twojej aplikacji. Każda aktywność to klasa dziedzicząca klasę "Activity". Klasa aktywności wyświetli interfejs użytkownika złożony z Widoków (Views) i będzie reagować na zdarzenia. Wiele aplikacji będzie miało więcej niż jeden ekran.



Klasa `ContentProvider`

- Aplikacje mogą przechowywać swoje dane w plikach, bazie SQLite, czy za pomocą innych mechanizmów. Dostawcy Treści (Content Providers) to klasy implementujące standardowy zestaw metod umożliwiając innym aplikacjom dostęp do składowanych danych.



Klasa BroadcastReceiver

- "Subskrybenci" (Broadcast Receivers) oraz "Zamierzenia" (Intents) umożliwiają wspólnie wykonanie aplikacji w reakcji na zewnętrzne zdarzenie - np. telefon zaczyna dzwonić, lub jest południe. Klasa BroadcastReceiver i jej subclasses działają jako "subskrybenci" w mechanizmie komunikacji typu publish/subscribe (pub/sub) zaimplementowanym w systemie Android. Jest to klasa bazowa dla kodu, który odbiera "zamierzenia" (Intents) wysłane za pomocą **sendBroadcast()**.



Klasa Intent

- Z kolei klasa Intent i jej subklasy stanowią w tym mechanizmie "wiadomości". Zamierzenia zawierają ponadto informację pozwalającą systemowi Android zdecydować, która aplikacja przejąć i odpowiedzieć na zdarzenie.



Klasa Service

- ❑ Usługi (Services) to kod, który działa nieprzerwanie przez dłuższy okres czasu i nie zawiera interfejsu użytkownika. Przykładowo tworząc odtwarzacz multimedialny mielibyśmy kilka Aktywności pozwalających wybrać listę piosenek do odtwarzania, lecz samo ich odtwarzanie powinno być obsługiwane przez usługę, która będzie działać nawet po przejściu do innych ekranów.



Manifest

- Manifest jest kolejną ważną częścią aplikacji opartej na systemie Android. Jest to plik XML, który pełni wiele zasadniczych funkcji. W pierwszej kolejności określa pakiety Javy dla aplikacji – jest to mechanizm organizacji klas. Nazwy pakietów są unikatowymi identyfikatorami w obrębie aplikacji. Ponadto opisuje komponenty aplikacji. Określa klasy oraz ich możliwości. Przykładem może być określenie dla aplikacji, które zamierzenie (Intent) będzie mogła obsłużyć. Są to informacje dla systemu Android, który potem decyduje, której aplikacji przekazać prawo do działania.



Manifest c.d.

Inne funkcje to :

- ❑ determinacja procesów do obsługi poszczególnych komponentów aplikacji,
- ❑ określenie zakresu uprawnień dla aplikacji w przypadku dostępu do chronionych części API oraz interakcją z innymi aplikacjami,
- ❑ określenie uprawnień jakie inne aplikacje powinny zawierać aby mieć dostęp do naszej aplikacji,
- ❑ określenie minimalnego poziomu API, który aplikacja wymaga.



Zasoby

- elementy graficzne
- GUI layouts (pliki XML)
- definicje Menu (pliki XML)
- łańcuchy znaków (Textual strings)



Pliki

Aplikacje Androida korzystają z różnych typów plików:

- pliki podstawowe,
- pliki baz danych,
- pliki OBB (Opaque Binary Blob)
reprezentujące zaszyfrowane pliki systemowe,
- pliki buforowane.



□ Dziękuję za uwagę

