Dr Katarzyna Grzesiak-Kopeć

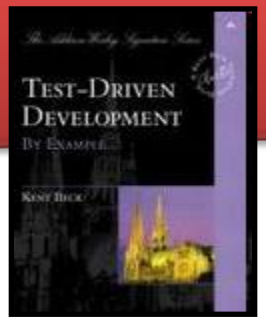
Inżynieria oprogramowania

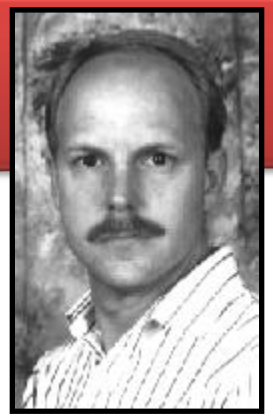
9.2 TDD & CI

Plan wykładu

- Tworzenie oprogramowania
- Najlepsze praktyki IO
- Inżynieria wymagań
- Technologia obiektowa i język UML
- Techniki IO
- Metodyki zwinne
- Refaktoryzacja
- Mierzenie oprogramowania
- Jakość oprogramowania
- Programowanie strukturalne
- Modelowanie analityczne
- Wprowadzenie do testowania

Podstawa





- XP
  - Programowanie poprzez testy (TDD)
  - Ciagła integracja (Continous Integration)
- Test Driven Development
  - Tworzenie oprogramowania sterowane testami
  - Kent Beck: Test Driven Development. By Example

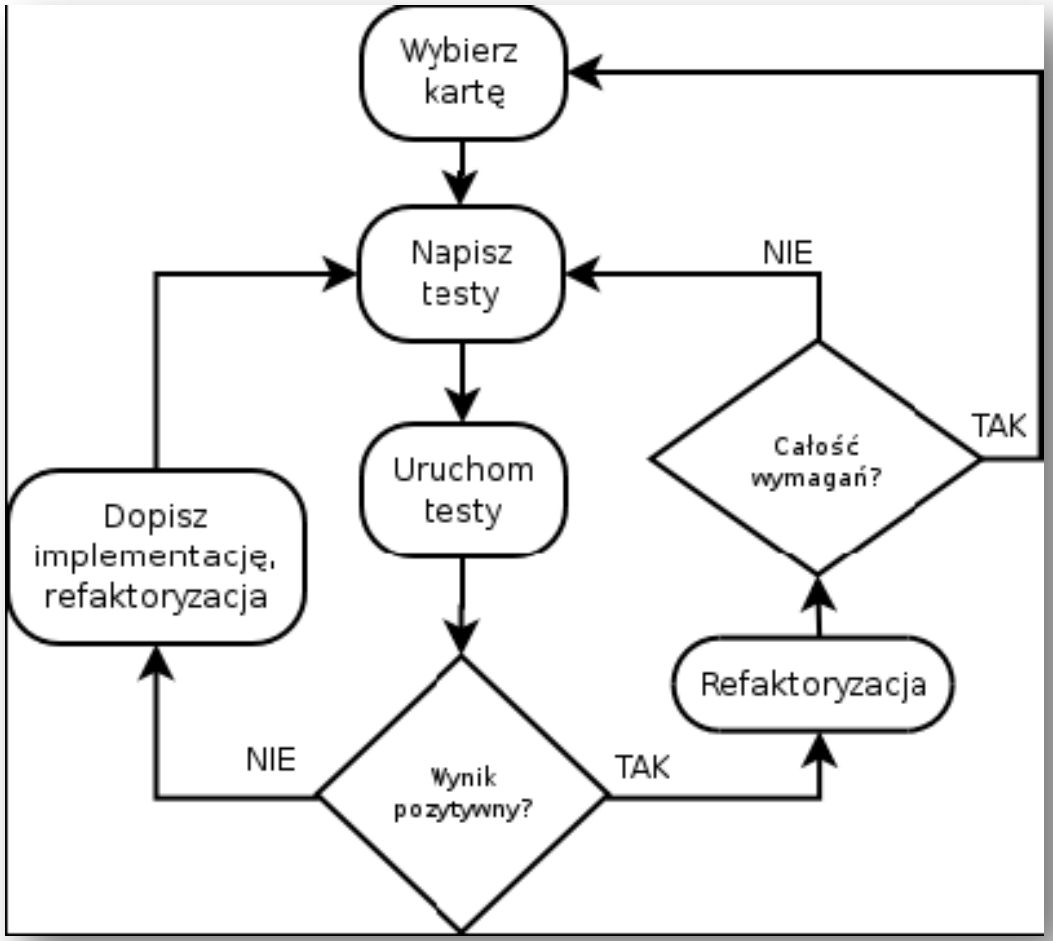
Testujemy systemy OO

- Obiekty
  - Weryfikacja zachowania i współpracy z innymi obiektami
- Sygnały
  - Nadawca, odbiorca, parametry
- Klasa
  - Specyfikacja klasy
  - Implementacja klasy
- Dziedziczenie
- Polimorfizm

TESTY Technika wodospadu

- Odsuwa w czasie integrację i agreguje testowanie
- Opóźnienia w implementacji – mniej czasu na testy
- Przekroczony budżet – obcinamy testy
- Zespół QA (Quality Assurance)
  - Programiści vs Testerzy

TDD



Zalety TDD

Erdogmus et al., On the Effectiveness of the Test-First Approach to Programming

- Feedback
  - Natychmiastowa informacja zwrotna
- Task-orientation
  - Lepsza dekompozycja realizacji projektu
- Quality assurance
  - Aktualne testy – określony poziom jakości kodu
- Low-level design
  - Lepsza dekompozycja problemu
- Wyższa jakość i produktywność

Krytyka TDD

- Tylko dla doświadczonych programistów
  - Pisanie testów do nieistniejącej implementacji
- Wymaga dużej samodyscypliny
  - Najpierw test potem funkcjonalność
- Wydłuża proces tworzenia oprogramowania
  - Dużo czasu na testy

Continous Integration (CI)

- Testowanie każdej zmiany w systemie
- Dedykowane narzędzia zintegrowane z:
  - Repozytorium kodu źródłowego
  - Systemem raportowym
- Natychmiastowe wykrycie błędów

Praktyki CI

M. Fowler, Continous Integration

- Jedno repozytorium kodu
- Automatyzacja budowy
- Testowanie jako część automatycznej budowy (samo-testowanie się)
- Wszyscy codziennie aktualizują źródła

Praktyki CI

M. Fowler, Continous Integration

- Zmiana w repozytorium kodu wyzwala proces integracji
- Szybka budowa aplikacji (~10min XP)
- Środowisko testowe sklonowanym środowiskiem produkcyjnym
- Wszyscy widzą co się dzieje

Zalety CI

- Zmniejszenie ryzyka
- Nie odsuwa w czasie integracji i nie agreguje testowanie
- Ułatwia znalezienie i usunięcie błędów
- Umożliwia szybkie wdrażanie

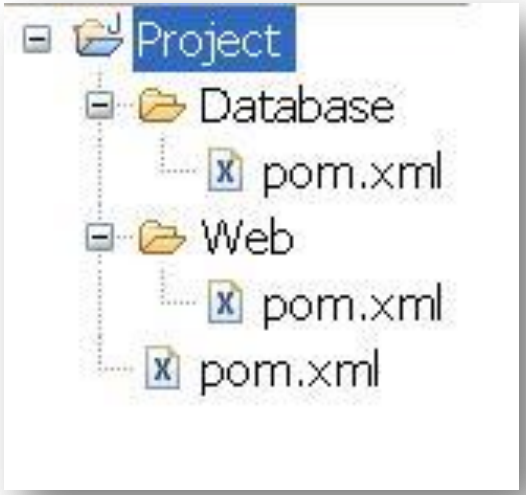
Platforma programistyczna

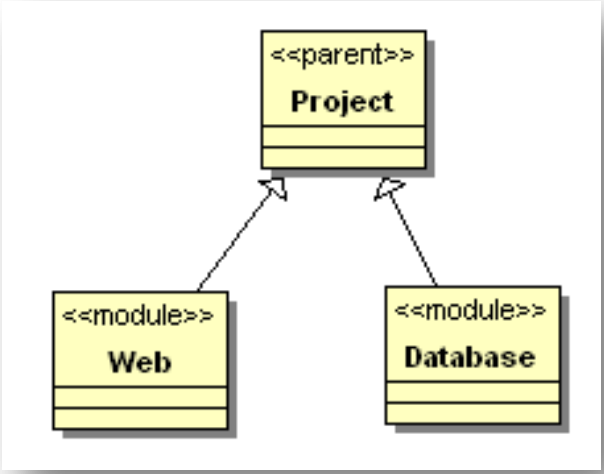
- Środowisko programistyczne
  - Apache Maven, Subversion, Eclipse IDE
- Biblioteki programistyczne
  - Springframework, JUnit, Mockito
- Narzędzia testujące
  - Selenium Remote Control, Soap UI
- Serwer ciągłej integracji
  - Hudson
- Serwer natychmiastowej komunikacji
  - Openfire Server

Adam Perlik, Środowisko ciągłej integracji do budowy aplikacji zgodnych z J2EE

Apache Maven

- http://maven.apache.org
- Automatyzacja budowy oprogramowania
- Koncepcji Obiektowego Modelu Projektu (Project Object Model)





Maven

- Wywoływanie całego cyklu budowy aplikacji za pomocą linii komend
- Zarządzanie zależnościami
- Tworzenie profili dla poszczególnych środowisk

Cykl budowy aplikacji

- JEE wiele rodzajów plików wynikowych
  - WAR (Web Archive), EAR (Enterprise Archive), RAR (Resource Archive) itd.
- Konwencja ułożenia plików
  - src/main/java; src/test/java; src/main/resources; src/test/resources; target/
- Budowa aplikacji
  - `> mvn install`

Zarządzanie zależnościami

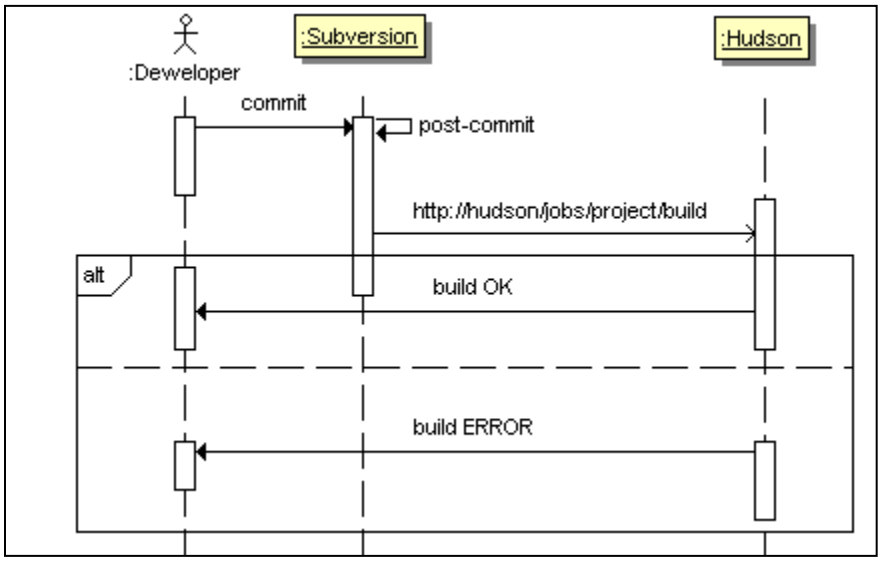
- Dwa rodzaje repozytorium bibliotek
  - Centralne
    - Repozytorium: http://repo2.maven.org/maven2/
    - Indeks zawartości: http://mvnrepository.com
  - Lokalne
- Konfiguracja dla całego zespołu

pom.xml

```
<dependencies>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.4</version>
        </dependency>
</dependencies>
```

Subversion

- http://subversion.tigris.org/
- System kontroli wersji
- Ciągła integracja – zmiana w repozytorium kodu wyzwala proces integracji
  - Triggery: *post-commit*

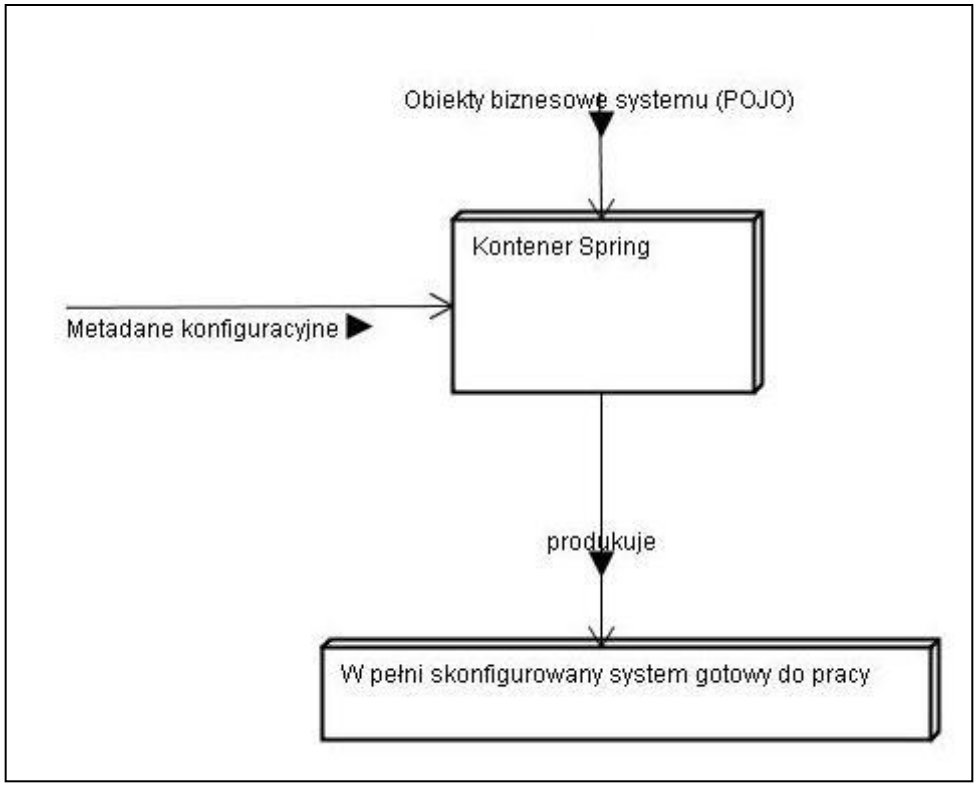


Springframework

- http://www.springsource.org/
- Szkielet programistyczny oparty na wzorcu projektowym Wstrzykiwanie Zależności (Dependency Injection)
  - Klasy nie tworzą swoich zależności
  - Zależności są klasom przekazywane

Springframework

- Plain Ordinary Java Objects
  - Klasy realizujące logikę biznesową
- Metadane konfiguracyjne XML



Springframework

```
private OrderServiceImpl orderService;
private BillingServiceImpl billingService;

public Controller() {
        this.orderService = new OrderServiceImpl();
        this.billingService = new BillingServiceImpl();
}
```

```
private IOrderService orderService;
private IBillingService billingService;

public Controller(IOrderService oService, IbillingService bService) {
        this.orderService = oService;
        this.billingService = bService;
}
```

Springframework

controller.xml

```
<beans>
	<bean id="orderController" class="org.perlo.OrderControllerImpl">
		<property name="orderService" ref="oService"/>
		<property name="billingService" ref="bService"/>
	</bean>
	<bean id="oService" class="org.perlo.OrderServiceImpl"/>
	<bean id="bService" class="org.perlo.BillingServiceImpl"/>
</beans>
```

```
ApplicationContext context = new ClassPathXmlApplicationContext(
	new String[] {"controler.xml"});
Controller orderController = (Controller) context.getBean(”orderController”);
```

```
ApplicationContext context = new ClassPathXmlApplicationContext(
	new String[] {"controler.xml", "controler-test.xml"});
Controller orderController = (Controller) context.getBean(”orderController”);
```

Springframework

- Kontener Odwrócenia Kontroli (Inversion of Control Container)
  - Szkielet Spring tworzy repozytorium obiektów, które sprawuje nad nimi kontrolę
  - ~~new OrderServiceImpl();~~

JUnit

- Biblioteka do testów jednostkowych
  - Kent Beck, Erich Gamma
  - http://junit.org
- Testy jednostkowe
- Testy integracyjne
  - JUnit + Spring
- Testy funkcjonalne
  - JUnit + Selenium Remote Control

JUnit

- Test Fixtures
  - Warunki wstępne do wykonania testu
- Test Suites
  - Zestawy testów współdzielące TF i traktowane jako jedna grupa
- Test Execution
- Assertions
  - Warunki wyznaczające wynik testu

JUnit Test Suite

- Klasa testowa
- Metody setUp, tearDown
  - realizują warunki wstępne i zapewniają zwalnianie zasobów
- Pozostałe metody
  - Kolejne procedury testowe (Test Execution)

Mock objects

- Obiekty zastępcze (zaślepki)
- Izolacja obiektów od ich zależności
  - Testy jednostkowe
- Deklaratywne definiowanie zachowania poszczególnych operacji interfejsów, które zaślepiamy

Mockito

- http://mockito.org

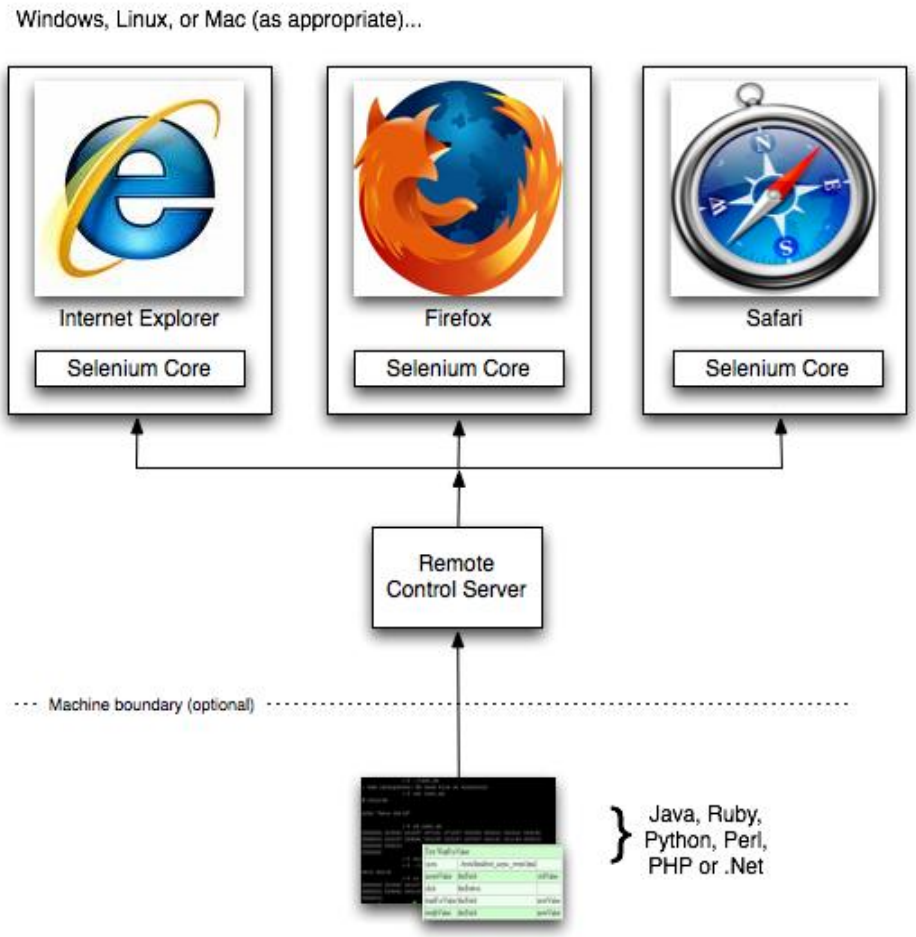

```
@Before
    public void setUp()
        RulesService rulesService = Mockito.mock(RulesService.class);
        Rules testRules = new FileRules("test-rules.xml");
        Mockito.when(rulesService.getRules()).thenReturn(testRules);
        scoringService = new ScoringServiceImpl(rulesService);
}
```

Selenium Remote Control

- http://seleniumhq.org
- Testy funkcjonalne oprogramowania, którego interfejs dostępowy stanowi przeglądarka internetowa
- Testowanie systemu w sposób dynamiczny
  - Symulacja klikającego użytkownika

Selenium Remote Control

- Sterowniki obsługi poszczególnych przeglądarek
- Testy implementowane przy wykorzystaniu API (Application Programming Interface)



Selenium Remote Control


```
public void testClickingLink() throws Exception {
    selenium.open(”http://localhost:8080/scoring/index.jsf”);
    selenium.click("link=Dodaj użytkownika");
    selenium.waitForPageToLoad(5*1000);
    String expectedTitle = "Dodawanie nowego użytkownika";
    assertEquals(expectedTitle, selenium.getTitle());
}
```

SoapUI

- http://www.soapui.org/
- Funkcjonalne oraz wydajnościowe testowanie utworzonych usług sieciowych
  - Usługi sieciowe oparte o protokół SOAP (Simple Object Access Protocol)
- Wywołanie usługi na podstawie dostarczonego wraz z nią pliku WSDL (Web Service Definition Language)
- Graficzny interfejs
- Definiowanie zarówno testów funkcjonalnych jak i wydajnościowych

Hudson

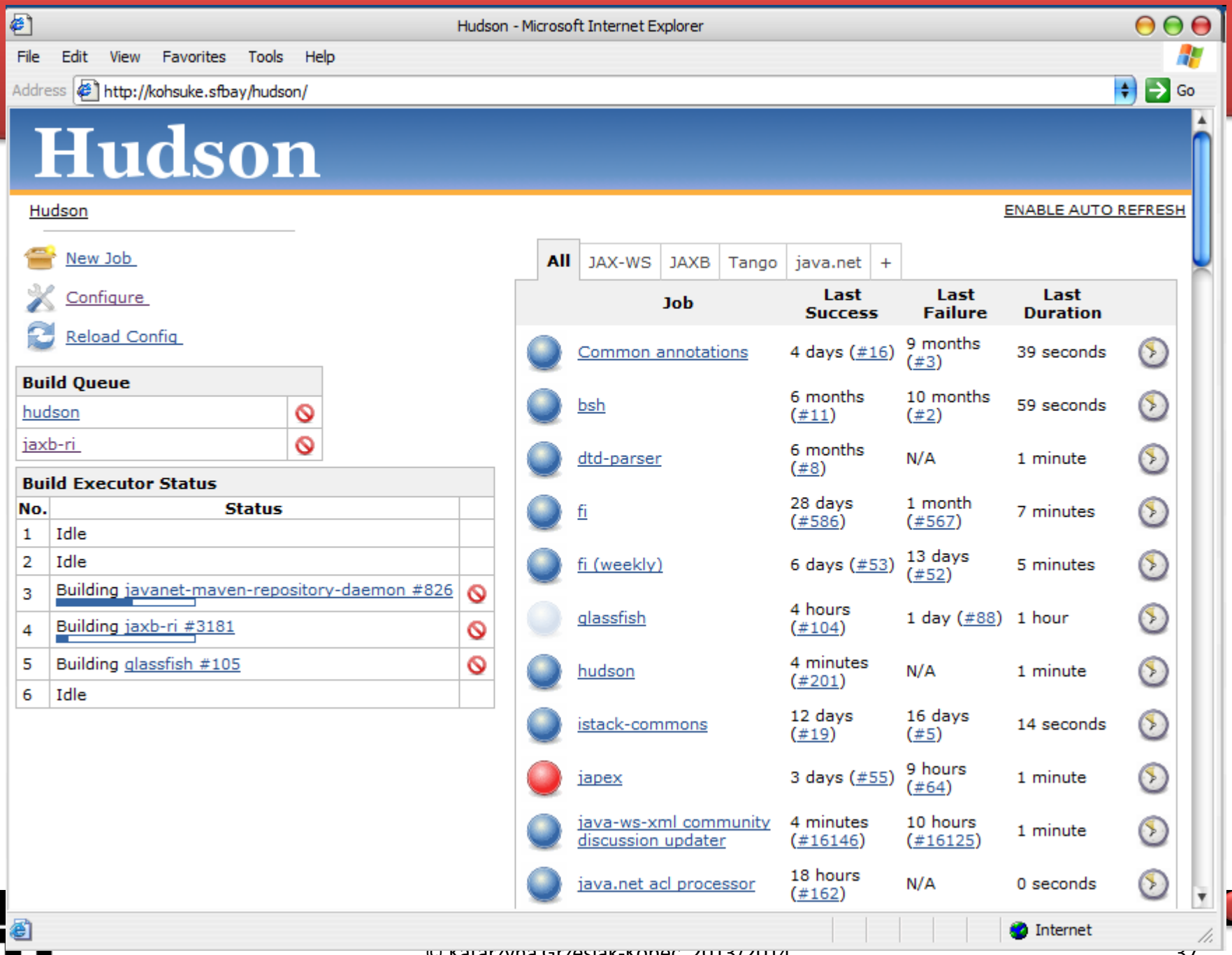
- https://hudson.dev.java.net/
- Serwer ciągłej integracji
- Definiowanie zadań realizujących cykl budowy aplikacji
- Potrafi skorzystać z istniejących skryptów takich narzędzi jak Maven czy Ant
- Pozwala na wywołanie dowolnego skryptu powłoki

Hudson

- W trakcie budowy uruchamia zdefiniowane testy
- Publikuje wyniki testów
- Może informować o wynikach testów
  - E-mail
  - Instant Messaging
- Raportuje o stopniu pokrycia kodu
- Istotne aspekty
  - Łatwość użycia
  - Rozszerzalność (ponad 50 plug-inów)

Hudson

- Automatyczne testowanie w określonych interwałach czasowych
- Możliwość wyzwolenia cyklu budowy aplikacji przez systemy zewnętrzne
  - Wprowadzenie zmian w repozytorium Subversion


Hudson - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://kohsuke.sfbay/hudson/
Go
Hudson
Hudson
ENABLE AUTO REFRESH
New Job
Configure
Reload Config
Build Queue
hudson
jaxb-ri
Build Executor Status
No. Status
1 Idle
2 Idle
3 Building javanet-maven-repository-daemon #826
4 Building jaxb-ri #3181
5 Building glassfish #105
6 Idle
All JAX-WS JAXB Tango java.net +
Job Last Success Last Failure Last Duration
Common annotations 4 days (#16) 9 months (#3) 39 seconds
bsh 6 months (#11) 10 months (#2) 59 seconds
dtd-parser 6 months (#8) N/A 1 minute
fi 28 days (#586) 1 month (#567) 7 minutes
fi (weekly) 6 days (#53) 13 days (#52) 5 minutes
glassfish 4 hours (#104) 1 day (#88) 1 hour
hudson 4 minutes (#201) N/A 1 minute
istack-commons 12 days (#19) 16 days (#5) 14 seconds
japex 3 days (#55) 9 hours (#64) 1 minute
java-ws-xml community discussion updater 4 minutes (#16146) 10 hours (#16125) 1 minute
java.net acl processor 18 hours (#162) N/A 0 seconds
Internet

Openfire Server

- http://www.igniterealtime.org/projects/openfire/index.jsp
- Serwer natychmiastowej komunikacji
- Wykorzystuje protokół XMPP (Extensible Messaging and Presence Protocol) bazujący na języku XML – **JABBER**
  - otwarty protokół wymiany wiadomości między dwoma dowolnymi punktami w Internecie w czasie rzeczywistym

Jabber

- Nie pełni żadnej roli w procesie samego testowania
- Ważne ogniwo w procesie ciągłej integracji
  - Hudson może powiadamiać zespół o wynikach integracji

KONIEC