

Scenariusz prezentacji - JRuby on Rails

- przygotowanie projektu
 - upewnić się, że jboss ma smsreciever.jar i nie ma smsapp.war
 - mysql odpalony
 - netbeans odpalony
 - przygotowany w finderze pliki javowe z klientami jms
- stworzenie projektu
 - rails smsapp
 - uruchomienie projektu
- otworzenie go w NetBeans
- script/generate controller sms
 - czy działa?
 - akcja index z Hello World!
 - o konfiguracji tyrada
- stworzenie bazy danych smsapp_development w CocoaMysql
- konfiguracja database.yml
- ruby script/generate model Message
- 001 migracja dla modelu: tid, body, tel
 - rake db:migrate i CocoaMySQL
- scaffold :message
- ruby script/generate scaffold Message sms
 - pokazanie w przeglądarce
 - walidacja: validates_presence_of :body, :tel, :tid, :status, vl->
- 002 migracja dla modelu status
 - status: text, image
 - add_column :messages, :status_id, :integer
 - trzy statusy [{}, {}, {}]
 - model: has_one, belongs_to
- poprawki
 - usunięcie z sms_controller niepotrzebnych rzeczy, dodanie Messages.find :all
 - ulepszenie list.rhtml
 - usunięcie tid z _form.rhtml
- poprawki cd
 - zabawa z przeglądarka
 - tid = Time.now.to_i.to_s
 - status = Status.find_by_text "SENDING"
 - przekopiowanie ikon
- periodic updater
 - sms_list do partiala
 - <%= periodically_call_remote :update => "sms-list", :url => { :action => 'periodic' }, :frequency => 5%
 - zabawa w przeglądarce
 - dodanie <%= javascript_include_tag :defaults %> do sms.rhtml
 - zmiana wartości w bd i pokazanie, że ajax działa
- ActiveRecord-JDBC
 - modyfikacja database.yml
 - development:
 - adapter: jdbc
 - host: localhost
 - url: jdbc:mysql://localhost/smsapp_development
 - driver: com.mysql.jdbc.Driver
 - database: smsapp_development
 - username: root
 - password:
 - to samo dla production
 - modyfikacja environment.rb
 - if RUBY_PLATFORM =~ /java/
 - require 'rubygems'
 - gem 'ActiveRecord-JDBC'
 - require 'jdbc_adapter'
 - end
- pokazanie klientów jms napisanych w javie
- skopiowanie pluginu activemessaging oraz setClasspath.rb
- a13m
 - ./setClasspath.rb
 - jruby script/generate processor SmsConfirmationReciever
 - messaging.rb
 - s.destination :sms_confirmation, '/queue/sms-confirmation'
 - s.destination :sms_send, '/queue/sms-send'
- broker.yml
 - development:
 - adapter: jms
 - username: guest
 - password: guest
 - initial_context_factory: org.jnp.interfaces.NamingContextFactory
 - url_pkg_prefixes: org.jboss.naming:org.jnp.interfaces

Scenariusz prezentacji - JRuby on Rails

```
provider: localhost
factory_name: ConnectionFactory
type: :queue
```

- to samo dla production
- SmsController
 - require 'activemessaging/processor'
 - include ActiveMessaging::MessageSender
 - publish :sms_send, @message.body, {"tel" => @message.tel, "tid" => @message.tid}
- usuniecie _reciever z Procesor, Sprawdzenie czy dziala
- SmsConfirmationProcessor
 - msg = Message.find_by_tid @message.headers[:tid]
 - puts "Found msg: #{msg.inspect}"

 - status = Status.find_by_text message
 - puts "Found status: #{status.inspect}"

 - msg.status = status
 - msg.save!
- sprawdzenie, czy dziala
 - uruchomienie JBossa
 - uruchomienie jruby script/server
 - pokazanie jboss web-console
 - poprawienie routes.rb
 - zabawa z smsami
- nie zapomniec o setClasspath
- goldspike
 - przekopowac plugin
 - dodac war.rb
 - jrake war:standalone:create
 - przekopowac do jbossa