**Aula 1 – Android 2 – Mobilidade e nuvem**

**Prof. Wagner Mendes Voltz - wagnerfusca@gmail.com**

**Objetivo:**

* Ter uma aplicação java que utilize serviços
* Ter a aplicação java disponibilizada na nuvem
* Usar serviço de versionamento na nuvem
* Usar serviço de container web na nuvem
* Realizar deploy na nuvem

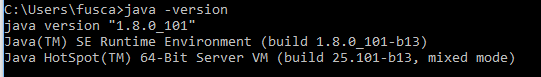
**Material Aula**

* Disponibilizar Java 8
  + Nas máquinas da FCV já tem instalado
  + Quem tem Java 7 também vai funcionar
* Disponibilizar Apache Maven 3.3.9
  + Caso você já tenha um Maven configurado, pode continuar a usar a sua versão.
* Disponibilizar Eclipse Neon (IDE Java)
  + Caso você tenha outra versão do Eclipse, pode usar a sua versão.
  + Caso você tenha Netbeans, pode usar esta IDE.
  + Caso você tenha Intellij, pode usar esta IDE.
* Ter uma conta no Heroku
* Instalar o Heroku Toolbelt
* Utilizar Postman

**Instalação JAVA**

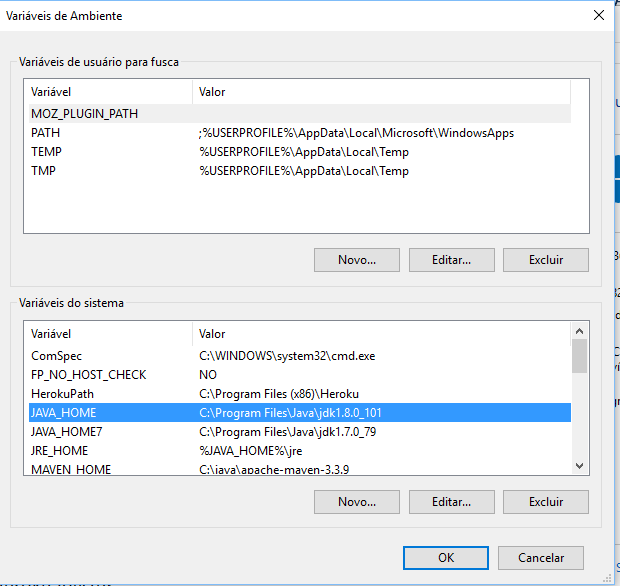
1) abra um prompt de commando (cmd.exe)

2) verifique se o java está instalado (digite **java -version**)



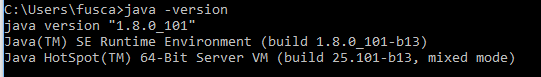
3) Caso não esteja configurado,

* Clique em instale jdk-8u101-windows-x64.exe
* Após instalação, pressione ***"bandeira windows" + PAUSE*** ***--> configurações avançadas do sistema --> aba avançado --> botão VARIAVEIS DE AMBIENTE*** e irá abrir a tela abaixo



* No quadrado VARIAVEIS DO SISTEMA (segundo quadrado), clique no botão NOVO. Adicione a variavel JAVA\_HOME (***em maiúsculo***) e adicione onde está a instalação da sua JDK (**não é a JRE!!!!!**)
* Após criar esta variável, clique em **PATH** e no início acrescente **%JAVA\_HOME%\bin\;** *(cuidado com o ponto e virgula, ele separa os comandos)*

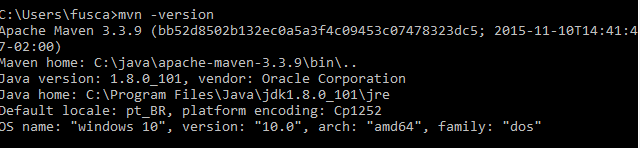
1. Pressione OK e abra um novo prompt de comando (cmd.exe) e digite **java –version**. Irá aparecer



**Instalação Apache Maven**

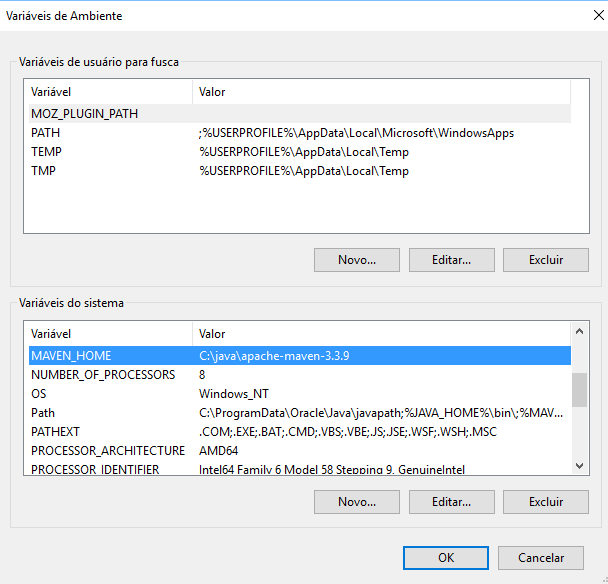
1) abra um prompt de commando (cmd.exe)

2) verifique se o Apache Maven está instalado (digite **mvn -version**)



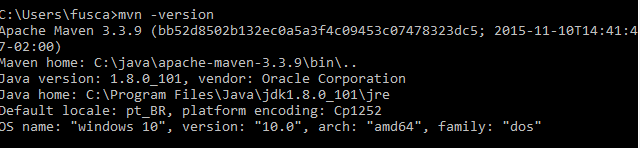
3) Caso não esteja configurado,

* Descompacte o arquivo ***apache-maven-3.3.9-bin.zip***
* Em seguida, pressione ***"bandeira windows" + PAUSE*** ***--> configurações avançadas do sistema --> aba avançado --> botão VARIAVEIS DE AMBIENTE*** e irá abrir a tela abaixo



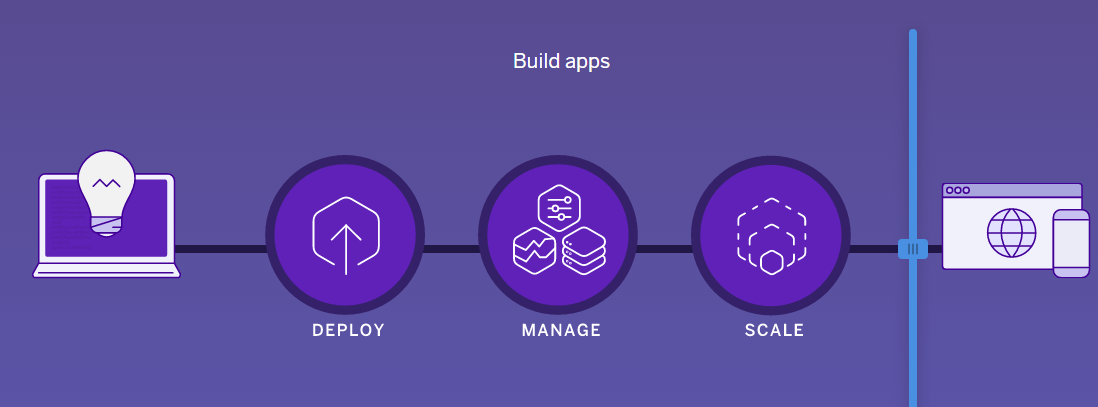
* No quadrado VARIAVEIS DO SISTEMA (segundo quadrado), clique no botão NOVO. Adicione a variavel MAVEN\_HOME (***em maiúsculo***) e adicione onde está a Apache Maven descompactado.
* Após criar esta variável, clique em **PATH** e logo após a JAVA\_HOME, adicione a variável **%MAVEN\_HOME%\bin\;** *(cuidado com o ponto e virgula, ele separa os comandos)*

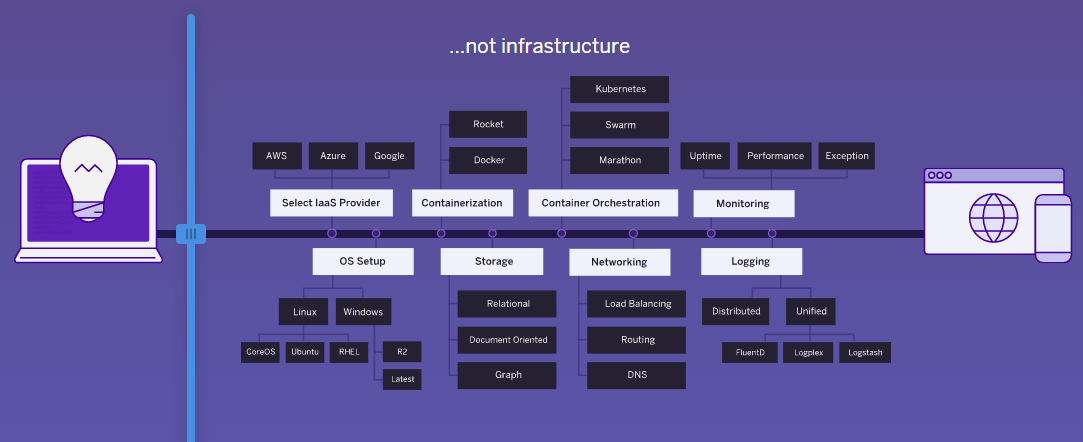
1. Pressione OK e abra um novo prompt de comando (cmd.exe) e digite **maven -version**. Irá aparecer

**

EXPLICAÇÃO NUVEM

Ver Slide – Estudo Caso Nuvem





Como iremos fazer esta aplicação:

1. Com Java + Eclipse + Serviços/RestEasy + Apache Maven + Apache Tomcat

Para testar aplicação

1. Postman

Para infraestrutura

1. Heroku

**Criar o projeto JAVA**

[*http://www.mkyong.com/webservices/jax-rs/resteasy-hello-world-example/*](http://www.mkyong.com/webservices/jax-rs/resteasy-hello-world-example/)

1. ir no prompt de comando (*cmd.exe*)
2. Criar uma pasta para o projeto (sugestão de nome: projetoNuvem)
3. Entre dentro a pasta (ex: C:\ projetoNuvem)
4. Digite o comando para criar o projeto

*mvn archetype:generate -DgroupId=br.edu.fcv -DartifactId=Calculadora -DarchetypeArtifactId=maven-archetype-webapp -DinteractiveMode=false*

1. Entre na pasta do projeto (*cd Calculadora)*
2. Digite o comando para importar as bibliotecas no Eclipse

*mvn eclipse:eclipse -Dwtpversion=2.0*

1. Gere uma versão do sistema
   1. Com testes unitários: *mvn clean install*
   2. Sem testes unitários: *mvn clean install -DskipTests*
2. Importar o projeto para o Eclipse (*File 🡪 Import 🡪 Maven 🡪 Existing Maven Projects*)
3. Adicione o RestEasy no projeto
   1. *Acesse o arquivo pom.xml*
   2. Insira a dependência resteasy

*<dependency>*

*<groupId>org.jboss.resteasy</groupId>*

*<artifactId>resteasy-jaxrs</artifactId>*

*<version>3.1.2.Final</version>*

*</dependency>*

1. Gere uma versão do sistema
   1. Com testes unitários: *mvn clean install*

**Criar serviços**

1. Crie um source folder src/main/java1
2. Renomeie para src/main/java
3. Crie o pacote br.edu.fcv.rest dentro do src/main/java
4. Crie uma classe MessageRestService
5. Adicione o código abaixo:

*package br.edu.fcv.rest;*

*import javax.ws.rs.GET;*

*import javax.ws.rs.Path;*

*import javax.ws.rs.PathParam;*

*import javax.ws.rs.core.Response;*

*@Path("/message")*

*public class MessageRestService {*

*@GET*

*@Path("/{param}")*

*public Response printMessage(@PathParam("param") String msg) {*

*String result = "Restful example : " + msg;*

*return Response.status(200).entity(result).build();*

*}*

*}*

1. Mude o arquivo, mude o **web.xml (src🡪main🡪webapp🡪WEB-INF)**

*<web-app id="WebApp\_ID" version="2.4"*

*xmlns="http://java.sun.com/xml/ns/j2ee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"*

*xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee*

*http://java.sun.com/xml/ns/j2ee/web-app\_2\_4.xsd">*

*<display-name>Calculadora</display-name>*

*<!-- this need same with resteasy servlet url-pattern -->*

*<context-param>*

*<param-name>resteasy.servlet.mapping.prefix</param-name>*

*<param-value>/rest</param-value>*

*</context-param>*

*<context-param>*

*<param-name>resteasy.resources</param-name>*

*<param-value>br.edu.fcv.rest.MessageRestService</param-value>*

*</context-param>*

*<listener>*

*<listener-class>*

*org.jboss.resteasy.plugins.server.servlet.ResteasyBootstrap</listener-class>*

*</listener>*

*<servlet>*

*<servlet-name>resteasy-servlet</servlet-name>*

*<servlet-class>*

*org.jboss.resteasy.plugins.server.servlet.HttpServletDispatcher</servlet-class>*

*</servlet>*

*<servlet-mapping>*

*<servlet-name>resteasy-servlet</servlet-name>*

*<url-pattern>/rest/\*</url-pattern>*

*</servlet-mapping>*

*</web-app>*

**Suba num contanier web (por exemplo Apache Tomcat)**

**Link para teste:http://localhost:8080/Calculadora/rest/message/1**

----------------------------

**HEROKU**

1. Acesse o site [www.heroku.com](http://www.heroku.com)
2. Crie um login e senha
3. Baixe o arquivo Heroku Toolbelt ([*https://devcenter.heroku.com/articles/heroku-command-line#download-and-install*](https://devcenter.heroku.com/articles/heroku-command-line#download-and-install))
4. Após instalação do Heroku Toolbelt, abra um prompt de comando (*cmd.exe*)
5. Conecte no heroku e crie um repositório
   * Digite: **heroku login** e preencha os seus dados
   * Digite: **heroku create** CALCULADORA\_SEU\_NOME
6. Abra a pasta do projeto
7. Versione o código do projeto
   * Digite: **git init**
   * Digite: **heroku git:remote -a** CALCULADORA\_SEU\_NOME
   * Digite**: git add pom.xml**
   * Digite**: git add src**
   * Digite**: git commit -m “mensagem para versão”**
   * Digite**: git push heroku master**

*Fonte: https://dashboard.heroku.com/apps/* CALCULADORA\_SEU\_NOME */deploy/heroku-git*

**\*\* COMANDOS IMPORTANTES HEROKU\*\***

* heroku login (faz login no heroku)
* heroku create (cria um vinculo heroku/git com aquela pasta)
* heroku git:remote -a newname (renomeia o repositório)
* heroku open (abrir o a aplicação)
* heroku logs --tail ou heroku logs(exibe logs)
* heroku ps (exibe quantos containers estao sendo executados)--> dyno
* heroku ps:scale web=1 (sobe um container para executar as aplicações)
* git init (criar um repositorio)
* git clone (obter um repositorio ja existente)
* git push heroku master (envia codigo para o git)
* git add . (adiciona todas as alteracoes)
* git commit -m "Demo" ("envia" alteração para o repositorio)
* git push heroku master (envia efetivamente a alteração para o repositorio master)
* git pull heroku master (busca a alteração para o repositorio master)
* git status(exibe o que tem de diferenca do repositorio com o local)
* git checkout \* (descarta todas as alteracoes locais)

*dica: http://rogerdudler.github.io/git-guide/index.pt\_BR.html*

**\*\*\*\* RODAR A APLICACAO DIRETO NUM TOMCAT NO HEROKU**

**\*\*\*\* AO REALIZAR PUSH, JÁ COMPILAR E FAZER O DEPLOY**

--------------------------------

*1 - mude o pom do projeto (vincular tomcat ao projeto)*

*2 - crie uma classe iniciadora (Main.java com método main)*

*3 - crie um servlet*

*4 - altere o Procfile*

*5 - suba as alterações no heroku*

**1 – POM +TOMCAT EMBUTIDO**

**Retire a tag**

<packaging>war</packaging>

Colocar antes da tag DEPENDENCIES

<properties>

<tomcat.version>9.0.0.M10</tomcat.version>

</properties>

Dentro da tag DEPENDENCIES

<dependency>

<groupId>org.apache.tomcat.embed</groupId>

<artifactId>tomcat-embed-core</artifactId>

<version>${tomcat.version}</version>

</dependency>

<dependency>

<groupId>org.apache.tomcat.embed</groupId>

<artifactId>tomcat-embed-logging-juli</artifactId>

<version>9.0.0.M6</version>

</dependency>

<dependency>

<groupId>org.apache.tomcat.embed</groupId>

<artifactId>tomcat-embed-jasper</artifactId>

<version>${tomcat.version}</version>

</dependency>

<dependency>

<groupId>org.apache.tomcat</groupId>

<artifactId>tomcat-jasper</artifactId>

<version>${tomcat.version}</version>

</dependency>

<dependency>

<groupId>org.apache.tomcat</groupId>

<artifactId>tomcat-jasper-el</artifactId>

<version>${tomcat.version}</version>

</dependency>

<dependency>

<groupId>org.apache.tomcat</groupId>

<artifactId>tomcat-jsp-api</artifactId>

<version>${tomcat.version}</version>

</dependency>

Dentro da tag BUILD

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<version>2.3.2</version>

<inherited>true</inherited>

<configuration>

<source>1.6</source>

<target>1.6</target>

</configuration>

</plugin>

<plugin>

<groupId>org.codehaus.mojo</groupId>

<artifactId>appassembler-maven-plugin</artifactId>

<version>1.1.1</version>

<configuration>

<assembleDirectory>target</assembleDirectory>

<programs>

<program>

<mainClass>launch.Main</mainClass>

<name>webapp</name>

</program>

</programs>

</configuration>

<executions>

<execution>

<phase>package</phase>

<goals>

<goal>assemble</goal>

</goals>

</execution>

</executions>

</plugin>

</plugins>

--------------------------

\*\*\*\* FAÇA UM mvn clean install ANTES DE SEGUIR PARA O PASSO 2 \*\*\*

**2 - classe MAIN.JAVA**

\*\*\*\*\* no PACOTE src/main/java/launch

package launch;

import java.io.File;

import org.apache.catalina.startup.Tomcat;

public class Main {

public static void main(String[] args) throws Exception {

String webappDirLocation = "src/main/webapp/";

Tomcat tomcat = new Tomcat();

//The port that we should run on can be set into an environment variable

//Look for that variable and default to 8080 if it isn't there.

String webPort = System.getenv("PORT");

if(webPort == null || webPort.isEmpty()) {

webPort = "8080";

}

tomcat.setPort(Integer.valueOf(webPort));

tomcat.addWebapp("/", new File(webappDirLocation).getAbsolutePath());

System.out.println("configuring app with basedir: " + new File("./" + webappDirLocation).getAbsolutePath());

tomcat.start();

tomcat.getServer().await();

}

}

--------------------------

**3 – classe HelloServlet**

\*\*\*\* NO PACOTE src/main/java/servlet

package servlet;

import java.io.IOException;

import javax.servlet.ServletException;

import javax.servlet.ServletOutputStream;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

@WebServlet(

name = "MyServlet",

urlPatterns = {"/hello"}

)

public class HelloServlet extends HttpServlet {

@Override

protected void doGet(HttpServletRequest req, HttpServletResponse resp)

throws ServletException, IOException {

ServletOutputStream out = resp.getOutputStream();

out.write("hello heroku".getBytes());

out.flush();

out.close();

}

}

--------------------------

**4 – Crie o arquivo Procfile na raiz do projeto e adicione a linha abaixo**

web: sh target/bin/webapp

--------------------------

**5 - suba as alterações no heroku**

git status

git add pom.xml

git add Procfile

git add src

git commit -m "Ready to deploy"

git push heroku master

heroku open

--------------------------

*fonte: https://devcenter.heroku.com/articles/create-a-java-web-application-using-embedded-tomcat*

-------------------------------

**\*\*\*\* Para rodar localmente**

WINDOWS 🡪 target\bin\webapp.bat

LINUX 🡪 sh target/bin/webapp

* <http://localhost:8080/rest/message/123>
* [https:// CALCULADORA\_SEU\_NOME.herokuapp.com/rest/message/123](https://calculadora2016.herokuapp.com/rest/message/123)

Caso dê erro: use este comando: heroku ps:scale web=1

-------------------------------

**Code Error Logs**

<https://devcenter.heroku.com/articles/error-codes>

**Exercícios**

1. **Calculadora: soma**
2. **Calculadora: subtração**
3. **Calculadora: multiplicação**
4. **Calculadora: divisão**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**RESTEASY USANDO POST**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. Com form

-------------------------------

<form method="POST" action="login">

Email Address: <input type="text" name="email">

<br>

Password: <input type="text" name="password">

<input type="submit">

</form>

-------------------------------

@Path("/")

public class LoginService

{

@Path("login")

@POST

public String login(@**FormParam**("email") String e, @**FormParam**("password") String p) {

return "Logged with " + e + " " + p;

}

1. Sem form (e usando json)

\*necessário usar a biblioteca **Jackson JSON processor para converter objetos json e devolver json para o cliente**

**POM.XML**

**<dependency>**

**<groupId>org.jboss.resteasy</groupId>**

**<artifactId>resteasy-jackson-provider</artifactId>**

**<version>**3.1.2.Final**</version>**

**</dependency>**

\*\*\* FAÇA mvn clean install \*\*\*\*

**Classe modelo**

**package br.edu.fcv.rest;**

**public class Produto {**

**private String nome;**

**private int quantidade;**

**public String getNome() {**

**return nome;**

**}**

**public void setNome(String nome) {**

**this.nome = nome;**

**}**

**public int getQuantidade() {**

**return quantidade;**

**}**

**public void setQuantidade(int quantidade) {**

**this.quantidade = quantidade;**

**}**

**}**

**JSONService.java**

**package br.edu.fcv.rest;**

**import javax.ws.rs.Consumes;**

**import javax.ws.rs.GET;**

**import javax.ws.rs.POST;**

**import javax.ws.rs.Path;**

**import javax.ws.rs.Produces;**

**import javax.ws.rs.core.Response;**

**@Path("/json/product")**

**public class JSONService {**

**@GET**

**@Path("/get")**

**@Produces("application/json")**

**public Produto getProdutoJSON() {**

**Produto produto = new Produto();**

**produto.setNome("Moto X - Android");**

**produto.setQuantidade(10);**

**return produto;**

**}**

**@POST**

**@Path("/post")**

**@Consumes("application/json")**

**public Response criarProdutoJSON(Produto produto) {**

**String result = "Product created : " + produto;**

**return Response.status(201).entity(result).build();**

**}**

**}**

**Mude o web.xml**

*<context-param>*

*<param-name>resteasy.resources</param-name>*

*<param-value>br.edu.fcv.rest.MessageRestService,*

*br.edu.fcv.rest.JSONService</param-value>*

*</context-param>*

Get : <http://localhost:8080/rest/json/produto/get>

Post: <http://localhost:8080/rest/json/produto/post> (abrir no postMan)

Headers:

* Content-Type
* application/json

Body:

* raw
* {"nome":"Notebook","quantidade":"133"}
* JSON(application/json)

*Fonte: http://www.mkyong.com/webservices/jax-rs/integrate-jackson-with-resteasy/*