Curso de Especialização em Desenvolvimento de Aplicações para Dispositivos Móveis e Cloud Computing

DM110

Desenvolvimento Java EE

Prof. Roberto Ribeiro Rocha
rrocha.roberto@gmail.com

Na aula de hoje

- Apresentação
- Introdução ao Java EE
- Ambiente de desenvolvimento
- Utilização de componentes Web

Antes de começar...

1) Verificar ambiente...
2) Criar a pasta DM110 e deszipar os arquivos de DM110-tools
3) Criar a pasta ws-dm110 e abrir uma “shell” e entrar nela
4) Fazer o clone do Github do projeto

```
git clone https://github.com/rrocharoberto/2019-dm110.git
```

5) Entrar na pasta ws-dm110/2019-dm110 e executar o Maven
    1) `mvn clean install`

Introdução ao Java EE

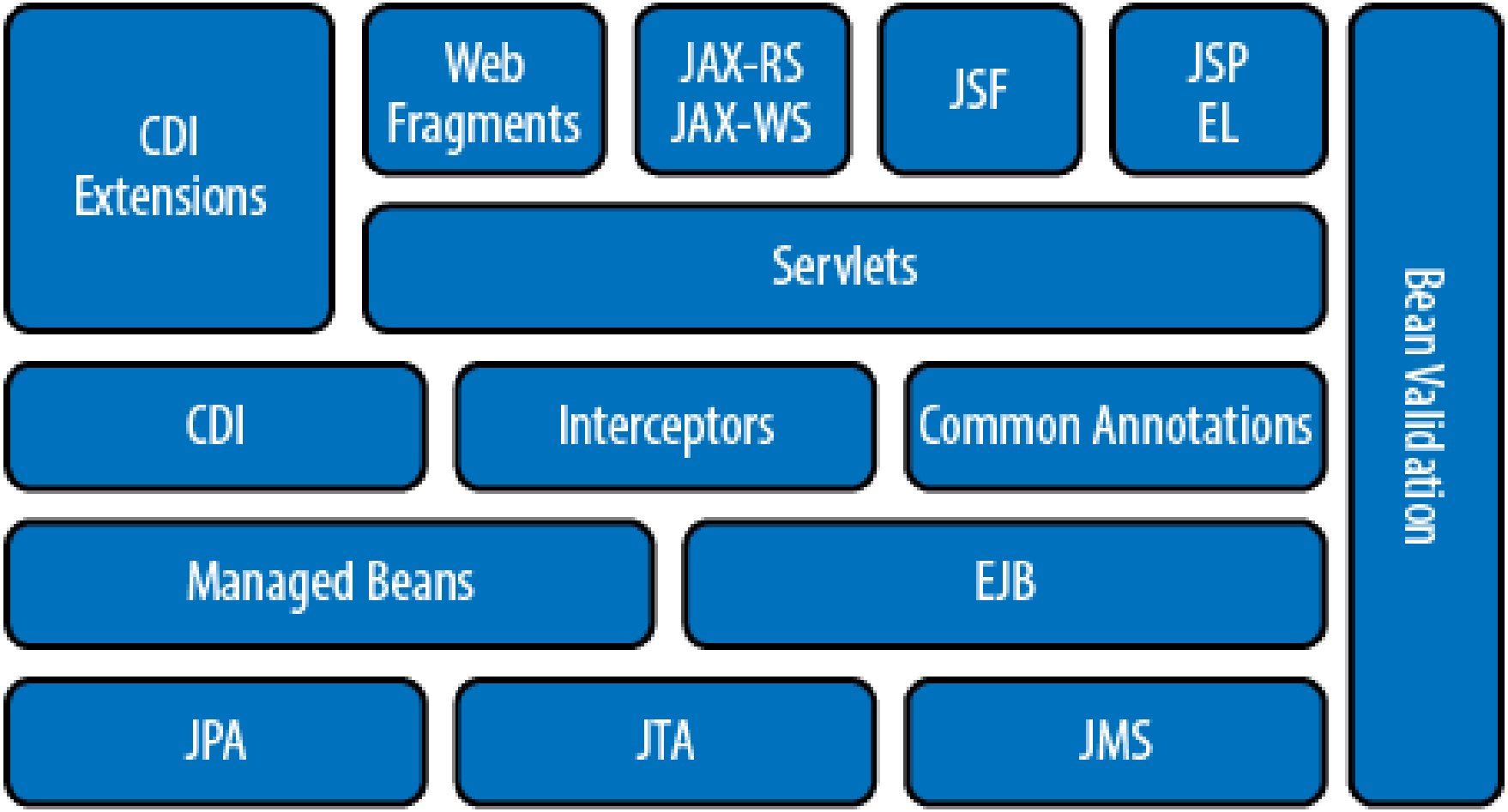
- Java Platform, Enterprise Edition
- Especificação do Java para construção de aplicações corporativas baseadas em Web
- Lançado pela Sun Microsystems em 1999

  (posteriormente adquirida pela Oracle)
- Extensão ao Java SE (Standard Edition)

Introdução ao Java EE

- Engloba diversas especificações, como:
  - JNDI – Java Naming and Directory Interface
  - EJB – Enterprise JavaBeans
  - Web: Servlets, JSP (JavaServer Pages) e JSF (JavaServer Faces)
  - JMS – Java Message Service
  - JPA – Java Persistence API
  - JMX – Java Management Extensions
  - JTA – Java Transaction API
  - JAX-RS – RESTful Web Services
  - JAAS – Java Autenthication and Authorization Service

Principais especificações



Fonte: https://www.oreilly.com/library/view/java-ee-6/9781449338329/ch01.html

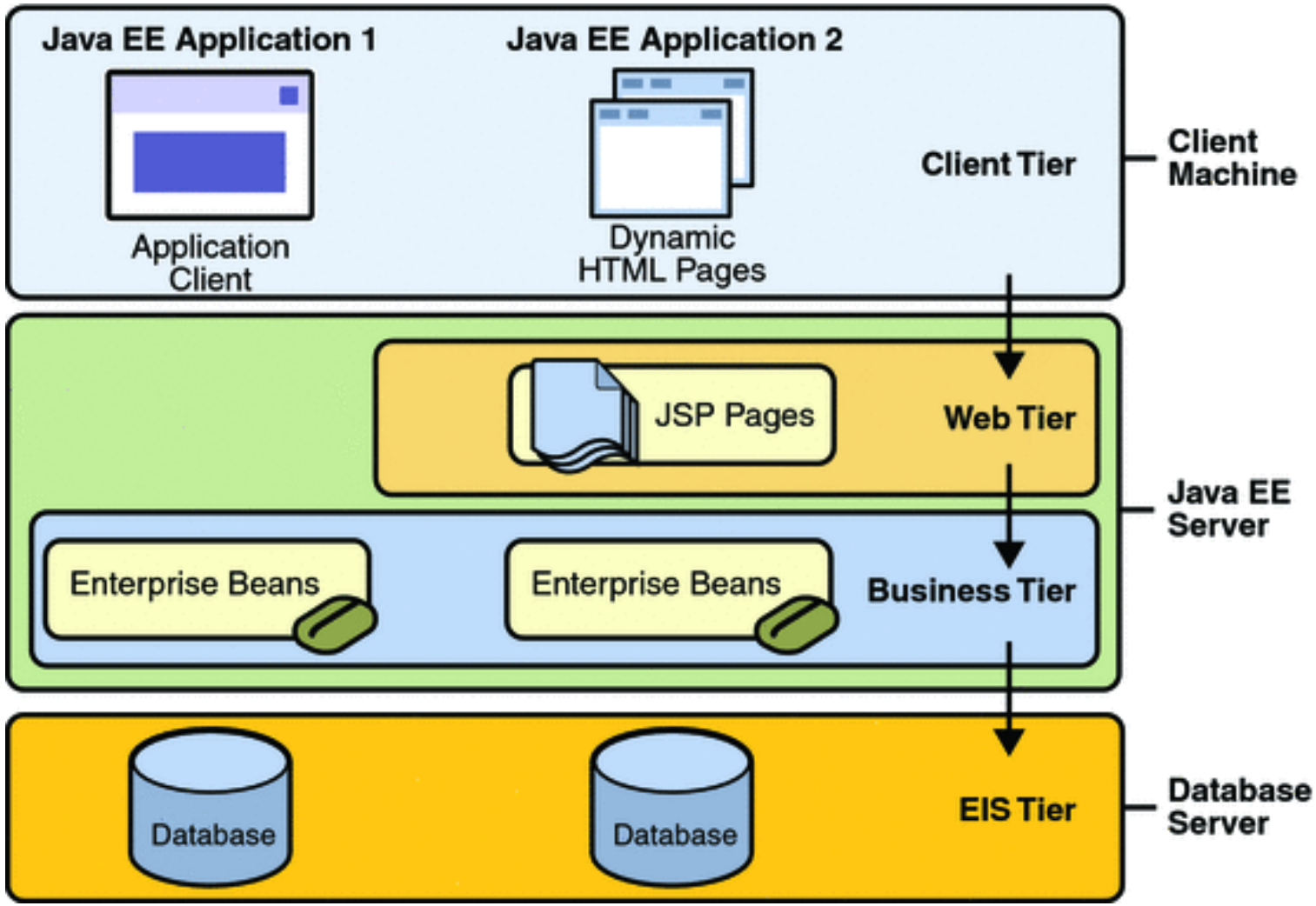
Servidor de Aplicações Java EE

- Framework de software que provê recursos para execução de aplicações Java EE, como:
  - Containers para execução de componentes específicos, como:
    - Web Container
    - EJB Container
  - Serviços disponíveis para os componentes:
    - Barramento JNDI
    - Filas e tópicos JMS
    - JDBC Datasources

Servidor de Aplicações Java EE

- Hospeda as aplicações corporativas
- Provê serviços como:
  - segurança, gerenciamento de transações, serviços de nomes (JNDI – Java Naming and Directory Interface) e conectividade remota, etc.
- Gerencia os componentes das aplicações corporativas através de containers
  - Web Container – gerencia a execução das páginas Web dinâmicas e Servlets
  - EJB Container – gerencia a execução dos componentes corporativos

Camadas Java EE



Fonte: https://docs.oracle.com/javaee/5/tutorial/doc/bnaay.html

Camada Cliente

- Composta por aplicações clientes
- Localizadas em uma máquina (virtual ou física) diferente do servidor Java EE
- Aplicações clientes podem ser:
  - Web browsers
  - Aplicações stand alone
  - Aplicações móveis
  - Outros servidores

Camada Web

- Gera conteúdo web dinâmico para os clientes
- Coleta entradas dos usuários e processa os resultados dos componentes da camada de negócios
- Controla o fluxo de janelas e páginas no cliente
- Realiza lógicas básicas
- Mantém dados temporariamente em componentes JavaBeans
  - Classes serializáveis
  - Construtor padrão
  - Getters e Setters para suas propriedades

Tecnologias da Camada Web

- Servlets – classes Java que processam requisições HTTP
- JSP (JavaServer Pages) – arquivos semelhantes a páginas HTML, que processam conteúdo dinâmico gerando HTML estáticos
- JSTL (JavaServer Pages Standards Tag Library) – biblioteca de tags que encapsulam funcionalidades em páginas JSP
- JavaServer Faces – framework baseado em componentes para criar interfaces de usuário
- JAX-RS – RESTful Web Services – API de web service Restful.

Camada de Negócios

- Possui componentes que encapsulam a lógica de negócios da aplicação
- Processa e envia os dados de negócios para armazenamento
- Recupera as informações de negócios e disponibiliza para a aplicação corporativa

Tecnologias da Camada de Negócios

- Enterprise JavaBeans (EJB) – componentes que lidam com a parte lógica de negócio da aplicação corporativa
- Java Persistence API (JPA) – provê o mapeamento objeto-relacional, permitindo o acesso aos bancos de dados relacionais
- Java Message Service (JMS) – permite aos componentes criarem, enviarem, receberem e processarem mensagens.
  - Integrado com Message-Driven Beans para processamento das mensagens

Tecnologias da Camada de Negócios

- JNDI – Java Naming and Directory Interface – serviço de nomes (registro e buscar de objetos pelo nome)
- JTA – Java Transaction API – possui recursos para gerenciar as transações dentro do container.
- JAAS – Java Autenthication and Authorization Service

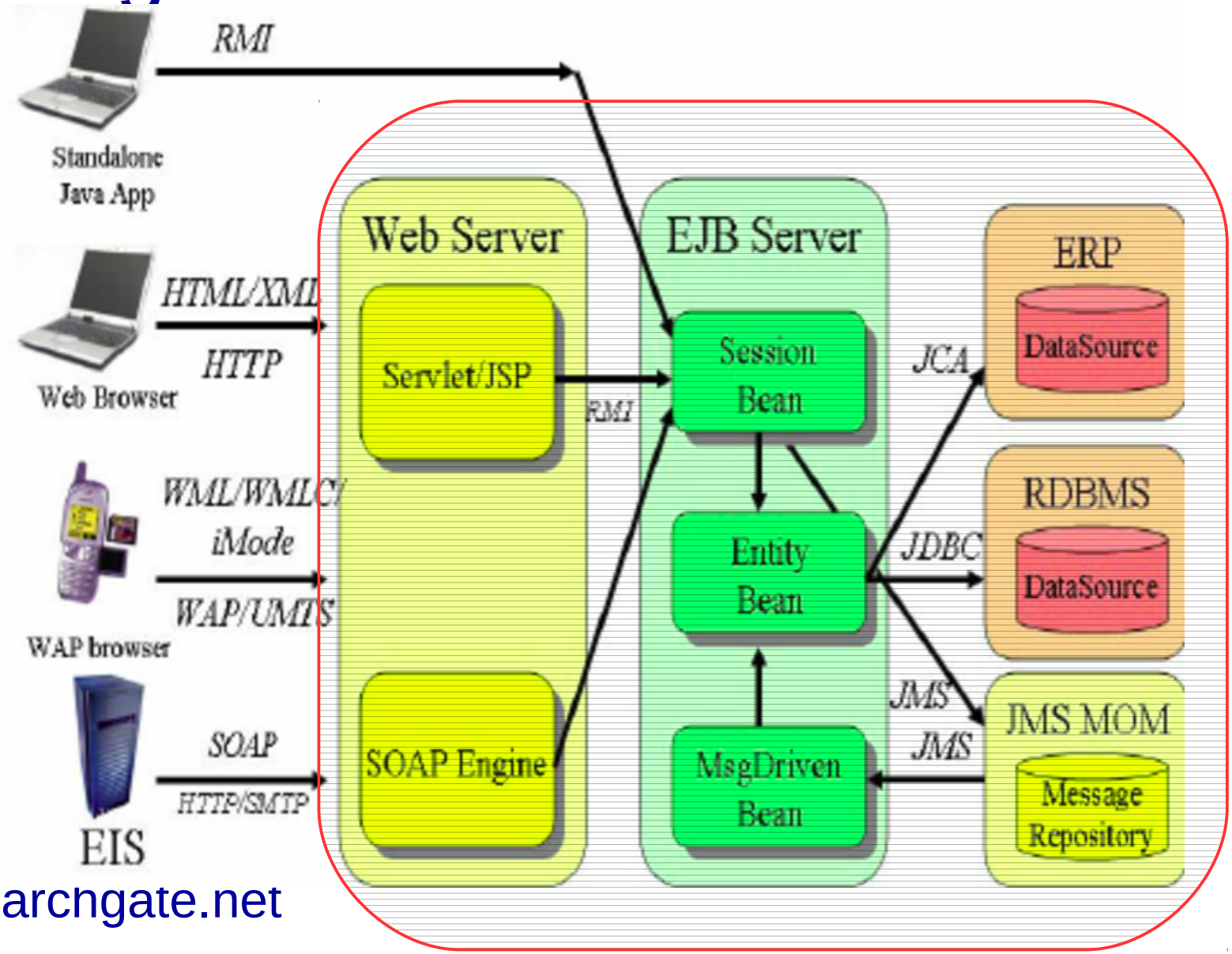
Camada EIS

- Sistemas de informações corporativos, externos ao servidor Java EE
- EIS podem ser:
  - Servidores de Banco de Dados
  - Mainframes
  - ERPs
  - Sistemas legados

Tecnologias da Camada EIS

- Java Database Connectivity API (JDBC) – provê conexão com bases de dados
- Java EE Connector Architecture – permite a conexão entre as aplicações corporativas Java EE e os demais Sistemas de Informações Corporativas
  - permite também a integração de sistemas legados no ambiente Java EE (multi-threads, sockets, etc).

Esquema geral Java EE



Fonte: researchgate.net

Ambiente de Desenvolvimento de Aplicações Java EE

- WildFly Application Server
- Eclipse IDE for Java EE Developers
- Apache Maven

WildFly Application Server

- Servidor de aplicações Java EE livre e de código aberto (LGPL)
- Anteriormente conhecido como JBoss AS
- Desenvolvido pela Red Hat, Inc.

- Prática 1: fazer a configuração inicial do Wildfly.

Apache Maven

- Ferramenta de construção de aplicações e gerenciamento de dependências
- Provê ciclo de vida pré-definido (diferentes de outras ferramentas anteriores a ele)
- Possui uma ampla gama de plugins específicos
  - Incluindo aplicações JavaEE

Estrutura de Projeto Maven JavaEE

- `Newproject/` → Projeto agregador (pai)

```
|-- pom.xml
|-- newproject-war/ → Projeto web
|   `-- pom.xml
|-- newproject-ejb/ → Projeto EJB
|   `-- pom.xml
|-- newproject-ejb-client/ → Projeto client
|   `-- pom.xml
`-- newproject-ear/ → Projeto EAR
    `-- pom.xml
```

Prática 2

- Compilar o projeto: `mvn clean install`
- Fazer o deploy do EAR no Wildfly (via browser).
- Testar os serviços.

Eclipse IDE for Java EE Developers

- Prática 3:
  - Abrir o Eclipse na workspace `ws-dm110`.
  - Importar os projetos no Eclipse.
  - Fazer update dos projetos via plugin do Maven.
  - Verificar a estrutura e conteúdos dos projetos.
  - Criar uma instância do server Wildfly.
  - Interromper o Wildfly do prompt.
  - Executar o Wildfly dentro do Eclipse.

Servlets

- Classes Java que respondem a requisições HTTP:
  - doGet – HTTP GET
  - doPost – HTTP POST
  - doPut – HTTP PUT
  - doDelete – HTTP DELETE
- Implementação da classe:

  javax.servlet.http.HttpServlet

Hello World Servlet

```
@WebServlet("/hello")
public class HelloServlet extends HttpServlet {
  public void doGet(
      HttpServletRequest request,
      HttpServletResponse response)
      throws ServletException, IOException {
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    out.println("<h1>Hello World!</h1>");
  }
}
```

Web Services (REST)

- Interface com anotações:
  - @Path
  - @GET
  - @POST
  - @Produces(MediaType.TEXT_HTML)
  - @Produces(MediaType.APPLICATION_JSON)
  - @Consumes(MediaType.APPLICATION_JSON)
  - Outras...
- Classe que implementa a interface com a lógica de processamento das requisições

Hello Service

```
@Path("/hello")
public interface HelloService {

@GET  @Path("/say/{name}")
@Produces(MediaType.TEXT_HTML)
String sayHello(@PathParam("name") String name);

@POST  @Path("/message")
@Produces(MediaType.APPLICATION_JSON)
MessageTO message(@FormParam("first") String p1,
                  @FormParam("last") String p2);

@POST  @Path("/storeMessage")
@Produces(MediaType.APPLICATION_JSON)
@Consumes(MediaType.APPLICATION_JSON)
public int storeNewMessage(MessageTO message);
```

Exercício Proposto

- Criar uma calculadora de 4 operações através de um Servlet.
  - Dica: utilizar o objeto `HttpServletRequest` para obter os valores e o operador vindos da URL do browser.
  - Exemplo de URL:

localhost:8080/dm110-web/servletCalc?op1=2&op2=3&op=soma

Exercício Extra

- Criar uma calculadora de 4 operações através de um Web Service.
  - Dica 1: utilizar (pensar em) nomes específicos para compor a URL de acordo com cada operador.
  - Dica 2: utilizar dois parâmetros para os operandos
  - Exemplo de URL

localhost:8080/dm110-web/api/calc/somar/{op1}/{op2}

Obrigado :)