FOR - J

Apostila de Exercícios

JEE - REST

Desenvolvedor de Web Services Rest

Exercício 1

```
package rest.exer1;

public interface Ola {
  public String servico();
}

package rest.exer1;

import java.util.Date;
import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.core.MediaType;

@Path("/ola")
public class RestOlaImp implements Ola {
  @GET
  @Produces(MediaType.TEXT_PLAIN)
  public String servico() {
    System.out.println("Executou o rest...");
    return new Date().toString();
  }
}

package rest.exer1;

import java.net.URI;
import javax.ws.rs.core.UriBuilder;
import org.glassfish.grizzly.http.server.HttpServer;
import org.glassfish.jersey.grizzly2.httpserver.GrizzlyHttpServerFactory;
import org.glassfish.jersey.server.ResourceConfig;

public class Servidor {
  public static void main(String[] args) {
    try {
      URI uri = UriBuilder.fromUri("http://localhost/").port(8080).build();
      ResourceConfig config = new ResourceConfig();
      config.packages("rest.exer1");
      HttpServer server = GrizzlyHttpServerFactory.createHttpServer(uri, config);
      System.out.println("servidor no ar teste - " + server);
      // Teste http://localhost:8080/ola
    } catch (Exception e) {
      System.out.println("Erro na execução do servidor JSE - " + e.getMessage());
    }
  }
}
```

Exercício 2

```
package rest.exer2;

import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.QueryParam;
import javax.ws.rs.core.MediaType;

@Path("/calculadora")
public class RestCalculadora {
  @GET
  @Produces(MediaType.TEXT_PLAIN)
  public String somar(@QueryParam("v1") Integer valor1, @QueryParam("v2") Integer valor2) {
    return "Valor da soma de " + valor1 + " + " + valor2 + " = " + (valor1 + valor2);
  }
}
```

```
package rest.exer2;

import java.net.URI;
import javax.ws.rs.core.UriBuilder;
import org.glassfish.grizzly.http.server.HttpServer;
import org.glassfish.jersey.grizzly2.httpserver.GrizzlyHttpServerFactory;
import org.glassfish.jersey.server.ResourceConfig;

public class Servidor {
  public static void main(String[] args) {
    try {
      URI uri = UriBuilder.fromUri("http://localhost/").port(8080).build();
      ResourceConfig config = new ResourceConfig();
      config.packages("rest.exer2");
      HttpServer server = GrizzlyHttpServerFactory.createHttpServer(uri, config);
      System.out.println("servidor no ar - " + server);
      // 1. http://localhost:8080/application.wadl
      // 2. http://localhost:8080/calculadora?v1=10&v2=10
    } catch (Exception e) {
      System.out.println("Erro na execução do servidor JSE - " + e.getMessage());
    }
  }
}
```

Exercício 3

```
package rest.exer3;

import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.PathParam;
import javax.ws.rs.Produces;
import javax.ws.rs.core.MediaType;

@Path("/cadastro/{username: [a-zA-Z]*}")
public class RestCadastro {
  @GET
  @Produces(MediaType.TEXT_PLAIN)
  public String processarCadatro(@PathParam("username") String nome) {
    System.out.println("Processando cadastro de = " + nome);
    return "Cadastro feito com sucesso para " + nome;
  }
}
```

```
package rest.exer3;

import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.PathParam;
import javax.ws.rs.Produces;
import javax.ws.rs.core.MediaType;

@Path("/notafiscal/{id:[0-9]*}")
public class RestNotaFiscal {
  @GET
  @Produces(MediaType.TEXT_PLAIN)
  public String processarNotafiscal(@PathParam("id") Long id) {
    System.out.println("Processando nota fiscal = " + id);
    return "nota processado " + id;
  }
}
```

```java
package rest.exer3;

import java.net.URI;
import javax.ws.rs.core.UriBuilder;
import org.glassfish.grizzly.http.server.HttpServer;
import org.glassfish.jersey.grizzly2.httpserver.GrizzlyHttpServerFactory;
import org.glassfish.jersey.server.ResourceConfig;

public class Servidor {
  public static void main(String[] args) {
    try {
      URI uri = UriBuilder.fromUri("http://localhost/").port(8080).build();
      ResourceConfig config = new ResourceConfig();
      config.packages("rest.exer3");
      HttpServer server = GrizzlyHttpServerFactory.createHttpServer(uri, config);
      System.out.println("servidor no ar - " + server);
      // Teste http://localhost:8080/cadastro/fernando
      // Teste http://localhost:8080/notafiscal/1552
    } catch (Exception e) {
      System.out.println("Erro na execução do servidor JSE - " + e.getMessage());
    }
  }
}
```

Exercício 4

```
package rest.exer4;

import javax.ws.rs.DefaultValue;
import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.QueryParam;
import javax.ws.rs.core.MediaType;

@Path("/venda")
public class RestVenda {
  @GET
  @Produces(MediaType.TEXT_PLAIN)
  public String vender(@QueryParam("nome") String nome, @QueryParam("produto") String produto,
      @QueryParam("tipo") @DefaultValue("pdf") String tipo) {
    System.out.println("Venda=" + nome);
    System.out.println("produto=" + produto);
    System.out.println("tipo=" + tipo);
    return "NOTA FISCAL=" + nome + " - " + tipo;
  }
}

package rest.exer4;

import java.net.URI;
import javax.ws.rs.core.UriBuilder;
import org.glassfish.grizzly.http.server.HttpServer;
import org.glassfish.jersey.grizzly2.httpserver.GrizzlyHttpServerFactory;
import org.glassfish.jersey.server.ResourceConfig;

public class Servidor {
  public static void main(String[] args) {
    try {
      URI uri = UriBuilder.fromUri("http://localhost/").port(8080).build();
      ResourceConfig config = new ResourceConfig();
      config.packages("rest.exer4");
      HttpServer server = GrizzlyHttpServerFactory.createHttpServer(uri, config);
      System.out.println("servidor no ar - " + server);
      // Teste http://localhost:8080/venda?nome=fernando franzini&produto=disco
      // Teste http://localhost:8080/venda?nome=fernando franzini&produto=bicicleta&tipo=texto
    } catch (Exception e) {
      System.out.println("Erro na execução do servidor JSE - " + e.getMessage());
    }
  }
}
```

Exercício 5

```java
package rest.exer5;

import java.math.BigDecimal;
import javax.xml.bind.annotation.XmlRootElement;

// Declaração de Elemento JAXB para conversão automatica.
@XmlRootElement
public class Funcionario {
  private String nome;
  private Long cpf;
  private BigDecimal salario;

  public String getNome() {
    return nome;
  }
  public void setNome(String nome) {
    this.nome = nome;
  }
  public Long getCpf() {
    return cpf;
  }
  public void setCpf(Long cpf) {
    this.cpf = cpf;
  }
  public BigDecimal getSalario() {
    return salario;
  }
  public void setSalario(BigDecimal salario) {
    this.salario = salario;
  }
  public String toString() {
    return "Funcionario [nome=" + nome + ", cpf=" + cpf + ", salario=" + salario + "]";
  }
}
```

```java
package rest.exer5;

import java.math.BigDecimal;
import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.core.MediaType;

@Path("/funcionario/")
public class RestEmpresa {

  private Funcionario criar() {
    Funcionario contato = new Funcionario();
    contato.setNome("Fernando Franzini");
    contato.setCpf(98012321323L);
    contato.setSalario(new BigDecimal("59999.99"));
    return contato;
  }

  @Path("/xml")
  @GET
  @Produces(MediaType.APPLICATION_XML)
  public Funcionario servicoXml() {
    System.out.println("Criando objeto java xml");
    Funcionario contato = criar();
    return contato;
  }

  @Path("/json")
  @GET
  @Produces(MediaType.APPLICATION_JSON)
  public Funcionario servicoJson() {
    System.out.println("Criando objeto java json");
    Funcionario contato = criar();
    return contato;
  }
}
```

```java
package rest.exer5;

import java.net.URI;
import javax.ws.rs.core.UriBuilder;
import org.glassfish.grizzly.http.server.HttpServer;
import org.glassfish.jersey.grizzly2.httpserver.GrizzlyHttpServerFactory;
import org.glassfish.jersey.server.ResourceConfig;

public class Servidor {

  public static void main(String[] args) {
    try {
      URI uri = UriBuilder.fromUri("http://localhost/").port(8080).build();
      ResourceConfig config = new ResourceConfig();
      config.packages("rest.exer5");
      HttpServer server = GrizzlyHttpServerFactory.createHttpServer(uri, config);
      System.out.println("servidor no ar - " + server);
      // Teste http://localhost:8080/funcionario/xml
      // Teste http://localhost:8080/funcionario/json
    } catch (Exception e) {
      System.out.println("Erro na execução do servidor JSE - " + e.getMessage());
    }
  }
}
```

Exercício 6

```java
package rest.exer6;

import java.util.Date;
import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.QueryParam;
import javax.ws.rs.core.MediaType;

@Path("/servicos/")
public class RestServico {
  @Path("/horario")
  @GET
  @Produces(MediaType.TEXT_PLAIN)
  public String hora() {
    return "hora servidor é = " + new Date().toString();
  }

  @Path("/soma")
  @GET
  @Produces(MediaType.TEXT_PLAIN)
  public Integer somar(@QueryParam("v1") Integer valor1, @QueryParam("v2") Integer valor2) {
    return valor1 + valor2;
  }
}
```

```java
package rest.exer6;

import java.net.URI;
import javax.ws.rs.core.UriBuilder;
import org.glassfish.grizzly.http.server.HttpServer;
import org.glassfish.jersey.grizzly2.httpserver.GrizzlyHttpServerFactory;
import org.glassfish.jersey.server.ResourceConfig;

public class Servidor {
  public static void main(String[] args) {
    try {
      URI uri = UriBuilder.fromUri("http://localhost/").port(8080).build();
      ResourceConfig config = new ResourceConfig();
      config.packages("rest.exer6");
      HttpServer server = GrizzlyHttpServerFactory.createHttpServer(uri, config);
      System.out.println("servidor no ar - " + server);
    } catch (Exception e) {
      System.out.println("Erro na execução do servidor JSE - " + e.getMessage());
    }
  }
}
```

```java
package rest.exer6;

import javax.ws.rs.client.Client;
import javax.ws.rs.client.ClientBuilder;
import javax.ws.rs.client.WebTarget;
import javax.ws.rs.core.Response;

public class Cliente {

  public static void cliente1() {
    Client cliente = ClientBuilder.newClient();
    WebTarget web = cliente.target("http://localhost:8080/servicos/horario");
    Response resposta = web.request().get();
    if (resposta.getStatus() == 200) {
      String horario = resposta.readEntity(String.class);
      System.out.println(horario);
    } else {
      System.out.println("erro na resposta = " + resposta.toString());
    }
    resposta.close();
    cliente.close();
  }

  public static void cliente2() {
    Client cliente = ClientBuilder.newClient();
    WebTarget web = cliente.target("http://localhost:8080/servicos/soma?v1=15&v2=15");
    Response resposta = web.request().get();
    if (resposta.getStatus() == 200) {
      Integer soma = resposta.readEntity(Integer.class);
      System.out.println("Soma é = " + soma);
    } else {
      System.out.println("erro na resposta = " + resposta.toString());
    }
    resposta.close();
    cliente.close();
  }
  public static void cliente3() {
    // chamada de endereço dinamico com parametros dinamicos.
    Client cliente = ClientBuilder.newClient();
    WebTarget web = cliente.target("http://localhost:8080/servicos");
    WebTarget requisicao = web.path("/soma").queryParam("v1", 20).queryParam("v2", 30);
    Response resposta = requisicao.request().get();
    if (resposta.getStatus() == 200) {
      Integer soma = resposta.readEntity(Integer.class);
      System.out.println("Soma é = " + soma);
    } else {
      System.out.println("erro na resposta = " + resposta.toString());
    }
    resposta.close();
    cliente.close();
  }

  public static void main(String[] args) {
    cliente1();
    cliente2();
    cliente3();
  }
}
```

Exercício 7

```
package rest.exer7;

import java.math.BigDecimal;
import javax.xml.bind.annotation.XmlRootElement;

@XmlRootElement
public class Pessoa {
  private String nome;
  private Integer idade;
  private BigDecimal salario;

  public Pessoa() {
    // JAB precisa de um default.
  }

  public Pessoa( String nome, Integer idade, BigDecimal salario) {
    this.nome = nome;
    this.idade = idade;
    this.salario = salario;
  }

  // gerar get e set + toString
```

```java
package rest.exer7;

import javax.ws.rs.Consumes;
import javax.ws.rs.POST;
import javax.ws.rs.Path;
import javax.ws.rs.core.MediaType;

@Path("/telefonia/")
public class RestTelefonia {
  @Path("/xml")
  @POST
  @Consumes(MediaType.APPLICATION_XML)
  public void gravar1(Pessoa pessoa) {
    System.out.println("gravando XML = " + pessoa);
  }

  @Path("/json")
  @POST
  @Consumes(MediaType.APPLICATION_JSON)
  public void gravar2(Pessoa pessoa) {
    System.out.println("gravando JSON = " + pessoa);
  }
}
```

```java
package rest.exer7;

import java.math.BigDecimal;
import javax.ws.rs.client.Client;
import javax.ws.rs.client.ClientBuilder;
import javax.ws.rs.client.Entity;
import javax.ws.rs.client.WebTarget;
import javax.ws.rs.core.Response;

public class Cliente {
  public static void viaXml() {
    Pessoa fernando = new Pessoa("Fernando", 35, new BigDecimal(1000));
    Client cliente = ClientBuilder.newClient();
    WebTarget web = cliente.target("http://localhost:8080/telefonia/xml");
    Response resposta = web.request().post(Entity.xml(fernando));
    if (resposta.getStatus() == 204) {
      // 204 resposta sem corpo.
      System.out.println("objeto pessoa enviado com sucesso via XML.");
    } else {
      System.out.println("erro na resposta = " + resposta.toString());
    }
    resposta.close();
    cliente.close();
  }
  public static void viaJson() {
    Pessoa fernando = new Pessoa("Fernando", 35, new BigDecimal(1000));
    Client cliente = ClientBuilder.newClient();
    WebTarget web = cliente.target("http://localhost:8080/telefonia/json");
    Response resposta = web.request().post(Entity.json(fernando));
    if (resposta.getStatus() == 204) {
      // 204 resposta sem corpo.
      System.out.println("objeto pessoa enviado com sucesso JSON.");
    } else {
      System.out.println("erro na resposta = " + resposta.toString());
    }
    resposta.close();
    cliente.close();
  }
  public static void main(String[] args) {
    viaXml();
    viaJson();
  }
}
```

Exercício 8

```java
package rest.exer8;

import java.math.BigDecimal;
import javax.xml.bind.annotation.XmlRootElement;

@XmlRootElement
public class Funcionario {
  private String nome;
  private Long cpf;
  private BigDecimal salario;

  public String getNome() {
    return nome;
  }
  public void setNome(String nome) {
    this.nome = nome;
  }
  public Long getCpf() {
    return cpf;
  }
  public void setCpf(Long cpf) {
    this.cpf = cpf;
  }
  public BigDecimal getSalario() {
    return salario;
  }
  public void setSalario(BigDecimal salario) {
    this.salario = salario;
  }
  public String toString() {
    return "Funcionario [nome=" + nome + ", cpf=" + cpf + ", salario=" + salario + "]";
  }
}
```

```java
package rest.exer8;

import java.math.BigDecimal;
import javax.ws.rs.DefaultValue;
import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.QueryParam;
import javax.ws.rs.core.MediaType;
import javax.ws.rs.core.Response;
import javax.ws.rs.core.Response.Status;

@Path("/teste")
public class RestRespostaDinamica {
  @GET
  // Veja que não tem produces, pq não vamos engessar...vamos retornar
  // dinamicamente.
  public Response teste(@QueryParam("numero") @DefaultValue("1") Integer numero) {
    if (numero == 1) {
      return Response.status(Status.FORBIDDEN)
          .type(MediaType.TEXT_PLAIN).entity("Não permitido").build();
    }
    Funcionario f = new Funcionario();
    f.setNome("Fer");
    f.setCpf(123456L);
    f.setSalario(BigDecimal.TEN);
    if (numero == 2) {
      return Response.ok().type(MediaType.APPLICATION_XML).entity(f).build();
    }
    if (numero == 3) {
      return Response.ok().type(MediaType.APPLICATION_JSON).entity(f).build();
    }
    return Response.ok().type(MediaType.TEXT_PLAIN).entity("Codigo não tratado").build();
  }
}
```

```java
package rest.exer8;

import java.net.URI;
import javax.ws.rs.core.UriBuilder;
import org.glassfish.grizzly.http.server.HttpServer;
import org.glassfish.jersey.grizzly2.httpserver.GrizzlyHttpServerFactory;
import org.glassfish.jersey.server.ResourceConfig;

public class Servidor {

  public static void main(String[] args) {
    try {
      URI uri = UriBuilder.fromUri("http://localhost/").port(8080).build();
      ResourceConfig config = new ResourceConfig();
      config.packages("rest.exer8");
      HttpServer server = GrizzlyHttpServerFactory.createHttpServer(uri, config);
      System.out.println("servidor no ar - " + server);
      // http://localhost:8080/teste
      // http://localhost:8080/teste?numero=2
      // http://localhost:8080/teste?numero=3
    } catch (Exception e) {
      System.out.println("Erro na execução do servidor JSE - " + e.getMessage());
    }
  }
}
```

Exercício 9

```
package rest.exer9;

public class NegocioException extends Exception {
  public NegocioException(String erro) {
    super(erro);
  }
}
```

```
package rest.exer9;

import javax.ws.rs.core.MediaType;
import javax.ws.rs.core.Response;
import javax.ws.rs.core.Response.Status;
import javax.ws.rs.ext.ExceptionMapper;
import javax.ws.rs.ext.Provider;

@Provider
public class NegocioExceptionResponse implements ExceptionMapper<NegocioException> {
  @Override
  public Response toResponse(NegocioException exception) {
    return Response.status(Status.NOT_FOUND).type(MediaType.TEXT_PLAIN)
        .entity(exception.getMessage()).build();
  }
}
```

```java
package rest.exer9;

import javax.xml.bind.annotation.XmlRootElement;

@XmlRootElement
public class Titulo {
  private String sacado;
  private Double valor;

  public Titulo() {
  }

  public Titulo(String sacado, double valor) {
    this.sacado = sacado;
    this.valor = valor;
  }

  public void validar() throws NegocioException {
    String erros = "";
    if (sacado == null) {
      erros += "sacado é obrigatorio; ";
    }
    if (valor == null) {
      erros += "valor é obrigatorio; ";
    }
    if (!erros.isEmpty()) {
      throw new NegocioException(erros);
    }
  }

  // gerar get/set/equals e hash/code
```

```java
package rest.exer9;

import java.util.List;

public interface ServicoTitulo {
  void gravar(Titulo titulo) throws NegocioException;
  void deletar(String sacado) throws NegocioException;
  List<Titulo> listar() throws NegocioException;
}
```

```
package rest.exer9;

import java.util.ArrayList;
import java.util.List;
import javax.ws.rs.Consumes;
import javax.ws.rs.DELETE;
import javax.ws.rs.GET;
import javax.ws.rs.POST;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.QueryParam;
import javax.ws.rs.core.MediaType;

@Path("/titulos/")
public class RestTitulosImp implements ServicoTitulo {

  // simulador de um banco de dados.
  private static final List<Titulo> banco = new ArrayList<Titulo>();
  static {
    banco.add(new Titulo("Sacado 1", 100));
    banco.add(new Titulo("Sacado 2", 200));
  }

  @Path("/gravar")
  @POST
  @Consumes(MediaType.APPLICATION_JSON)
  @Override
  public void gravar(Titulo titulo) throws NegocioException {
    titulo.validar();
    banco.add(titulo);
  }
  @Path("/deletar")
  @DELETE
  @Consumes(MediaType.TEXT_PLAIN)
  @Override
  public void deletar(@QueryParam("sacado") String sacado) throws NegocioException {
    if (sacado == null) {
      throw new NegocioException("Sacado é obrigatorio para deleção.");
    }
    Titulo deletar = null;
    for (Titulo t : banco) {
      if (t.getSacado().equals(sacado)) {
        deletar = t;
        break;
      }
    }
    if (deletar == null) {
      throw new NegocioException("Sacado inexistente.");
    }
    banco.remove(deletar);
  }

  @Path("/listar")
  @GET
  @Produces(MediaType.APPLICATION_JSON)
  @Override
  public List<Titulo> listar() throws NegocioException {
    return new ArrayList<Titulo>(banco);
  }
}
```

```
package rest.exer9;

import java.net.URI;
import javax.ws.rs.core.UriBuilder;
import org.glassfish.grizzly.http.server.HttpServer;
import org.glassfish.jersey.grizzly2.httpserver.GrizzlyHttpServerFactory;
import org.glassfish.jersey.server.ResourceConfig;

public class Servidor {
  public static void main(String[] args) {
    try {
      URI uri = UriBuilder.fromUri("http://localhost/").port(8080).build();
      ResourceConfig config = new ResourceConfig();
      config.packages("rest.exer9");
      HttpServer server = GrizzlyHttpServerFactory.createHttpServer(uri, config);
      System.out.println("servidor no ar - " + server);
    } catch (Exception e) {
      System.out.println("Erro na execução do servidor JSE - " + e.getMessage());
    }
  }
}
```

```
package rest.exer9;

import java.util.List;
import javax.ws.rs.client.Client;
import javax.ws.rs.client.ClientBuilder;
import javax.ws.rs.client.Entity;
import javax.ws.rs.client.WebTarget;
import javax.ws.rs.core.GenericType;
import javax.ws.rs.core.Response;

public class Cliente {

  private static WebTarget criar() {
    Client cliente = ClientBuilder.newClient();
    WebTarget web = cliente.target("http://localhost:8080/titulos");
    return web;
  }
  public static void criar(Titulo titulo) {
    WebTarget web = criar().path("/gravar");
    Response resposta = web.request().post(Entity.json(titulo));
    System.out.println(resposta.getStatus() + " - " + resposta.readEntity(String.class));
    resposta.close();
  }

  public static void listar() {
    WebTarget web = criar().path("/listar");
    Response resposta = web.request().get();
    System.out.println(resposta.getStatus());
    List<Titulo> titulos = resposta.readEntity(new GenericType<List<Titulo>>() {
    });
    for (Titulo titulo : titulos) {
      System.out.println(titulo.getSacado() + " - " + titulo.getValor());
    }
    resposta.close();
  }
```

```
    public static void deletar(String sacado) {
      WebTarget web = criar().path("/deletar");
      Response resposta = web.queryParam("sacado", sacado).request().delete();
      System.out.println(resposta.getStatus() + " - " + resposta.readEntity(String.class));
      resposta.close();
    }

    public static void main(String[] args) {
      criar(new Titulo());
      criar(new Titulo("Fernando", 120));
      criar(new Titulo("Luana", 220));
      criar(new Titulo("Xicao", 550));
      listar();
      deletar("bart");
      deletar("Xicao");
      listar();
    }
  }
```

Exercício 10

```
package rest.exer10;

import java.io.IOException;
import javax.ws.rs.container.ContainerRequestContext;
import javax.ws.rs.container.ContainerRequestFilter;
import javax.ws.rs.ext.Provider;

@Provider
public class FiltroHorario implements ContainerRequestFilter {
  @Override
  public void filter(ContainerRequestContext request) throws IOException {
    System.out.println("-->filtro de request");
  }
}
```

```
package rest.exer10;

import java.text.SimpleDateFormat;
import java.util.Date;
import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.core.MediaType;

@Path("/horas1")
public class RestHorario {
  private SimpleDateFormat sdf = new SimpleDateFormat("HH:mm:ss dd/MM/yyyy");
  @GET
  @Produces(MediaType.TEXT_PLAIN)
  public String horario() {
    System.out.println("=>RestHorario.horario()");
    return sdf.format(new Date());
  }
}
```

```java
package rest.exer10;

import java.net.URI;
import javax.ws.rs.core.UriBuilder;
import org.glassfish.grizzly.http.server.HttpServer;
import org.glassfish.jersey.grizzly2.httpserver.GrizzlyHttpServerFactory;
import org.glassfish.jersey.server.ResourceConfig;

public class Servidor {
  public static void main(String[] args) {
    try {
      URI uri = UriBuilder.fromUri("http://localhost/").port(8080).build();
      ResourceConfig config = new ResourceConfig();
      config.packages("rest.exer10");
      HttpServer server = GrizzlyHttpServerFactory.createHttpServer(uri, config);
      System.out.println("servidor no ar teste - " + server);
      // Teste http://localhost:8080/horas1
    } catch (Exception e) {
      System.out.println("Erro na execução do servidor JSE - " + e.getMessage());
    }
  }
}
```

Exercício 11

```java
package rest.exer11;

import java.io.IOException;
import javax.ws.rs.container.ContainerRequestContext;
import javax.ws.rs.container.ContainerRequestFilter;
import javax.ws.rs.ext.Provider;

@Provider
public class FiltroCabecalhos implements ContainerRequestFilter {
  @Override
  public void filter(ContainerRequestContext request) throws IOException {
    request.getHeaders().forEach((h, l) -> {
      System.out.println("Header: " + h);
      l.forEach(i -> System.out.println("===>" + l));
    });
  }
}
```

```java
package rest.exer11;

import java.text.SimpleDateFormat;
import java.util.Date;
import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.core.MediaType;

@Path("/horas2")
public class RestHorario {
  private SimpleDateFormat sdf = new SimpleDateFormat("HH:mm:ss dd/MM/yyyy");
  @GET
  @Produces(MediaType.TEXT_PLAIN)
  public String horario() {
    System.out.println("=>RestHorario.horario()");
    return sdf.format(new Date());
  }
}
```

```java
package rest.exer11;

import java.net.URI;
import javax.ws.rs.core.UriBuilder;
import org.glassfish.grizzly.http.server.HttpServer;
import org.glassfish.jersey.grizzly2.httpserver.GrizzlyHttpServerFactory;
import org.glassfish.jersey.server.ResourceConfig;

public class Servidor {
  public static void main(String[] args) {
    try {
      URI uri = UriBuilder.fromUri("http://localhost/").port(8080).build();
      ResourceConfig config = new ResourceConfig();
      config.packages("rest.exer11");
      HttpServer server = GrizzlyHttpServerFactory.createHttpServer(uri, config);
      System.out.println("servidor no ar teste - " + server);
      // Teste http://localhost:8080/horas2
    } catch (Exception e) {
      System.out.println("Erro na execução do servidor JSE - " + e.getMessage());
    }
  }
}
```

Exercício 12

```
package rest.exer12;

import java.io.IOException;
import javax.ws.rs.container.ContainerRequestContext;
import javax.ws.rs.container.ContainerResponseContext;
import javax.ws.rs.container.ContainerResponseFilter;
import javax.ws.rs.ext.Provider;

@Provider
public class FiltroRespostaHorario implements ContainerResponseFilter {
  @Override
  public void filter(ContainerRequestContext crc,
      ContainerResponseContext response) throws IOException {
    System.out.println("-->filtro de response");
  }
}
```

```
package rest.exer12;

import java.text.SimpleDateFormat;
import java.util.Date;
import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.core.MediaType;

@Path("/horas3")
public class RestHorario {
  private SimpleDateFormat sdf = new SimpleDateFormat("HH:mm:ss dd/MM/yyyy");
  @GET
  @Produces(MediaType.TEXT_PLAIN)
  public String horario() {
    System.out.println("=>RestHorario.horario()");
    return sdf.format(new Date());
  }
}
```

```java
package rest.exer12;

import java.net.URI;
import javax.ws.rs.core.UriBuilder;
import org.glassfish.grizzly.http.server.HttpServer;
import org.glassfish.jersey.grizzly2.httpserver.GrizzlyHttpServerFactory;
import org.glassfish.jersey.server.ResourceConfig;

public class Servidor {
  public static void main(String[] args) {
    try {
      URI uri = UriBuilder.fromUri("http://localhost/").port(8080).build();
      ResourceConfig config = new ResourceConfig();
      config.packages("rest.exer12");
      HttpServer server = GrizzlyHttpServerFactory.createHttpServer(uri, config);
      System.out.println("servidor no ar teste - " + server);
      // Teste http://localhost:8080/horario3
    } catch (Exception e) {
      System.out.println("Erro na execução do servidor JSE - " + e.getMessage());
    }
  }
}
```

Exercício 13

```java
package rest.exer13;

import java.io.IOException;
import javax.ws.rs.container.ContainerRequestContext;
import javax.ws.rs.container.ContainerResponseContext;
import javax.ws.rs.container.ContainerResponseFilter;
import javax.ws.rs.ext.Provider;

@Provider
public class FiltroRespostaHeader implements ContainerResponseFilter {
  @Override
  public void filter(ContainerRequestContext crc,
      ContainerResponseContext response) throws IOException {
    response.getHeaders().add("Criado-Por", "AulaJava");
  }
}
```

```java
package rest.exer13;

import java.text.SimpleDateFormat;
import java.util.Date;
import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.core.MediaType;

@Path("/horas4")
public class RestHorario {
  private SimpleDateFormat sdf = new SimpleDateFormat("HH:mm:ss dd/MM/yyyy");
  @GET
  @Produces(MediaType.TEXT_PLAIN)
  public String horario() {
    System.out.println("=>RestHorario.horario()");
    return sdf.format(new Date());
  }
}
```

```java
package rest.exer13;

import java.net.URI;
import javax.ws.rs.core.UriBuilder;
import org.glassfish.grizzly.http.server.HttpServer;
import org.glassfish.jersey.grizzly2.httpserver.GrizzlyHttpServerFactory;
import org.glassfish.jersey.server.ResourceConfig;

public class Servidor {
  public static void main(String[] args) {
    try {
      URI uri = UriBuilder.fromUri("http://localhost/").port(8080).build();
      ResourceConfig config = new ResourceConfig();
      config.packages("rest.exer13");
      HttpServer server = GrizzlyHttpServerFactory.createHttpServer(uri, config);
      System.out.println("servidor no ar teste - " + server);
      // Teste http://localhost:8080/horas3
    } catch (Exception e) {
      System.out.println("Erro na execução do servidor JSE - " + e.getMessage());
    }
  }
}
```

Exercício 14

```java
package rest.exer14;

import java.io.IOException;

import javax.ws.rs.container.ContainerRequestContext;
import javax.ws.rs.container.ContainerRequestFilter;
import javax.ws.rs.core.Response;
import javax.ws.rs.ext.Provider;

@Provider
public class FiltroAutenticador implements ContainerRequestFilter {
  @Override
  public void filter(ContainerRequestContext request) throws IOException {
    if (request.getHeaderString("usuario") == null) {
      request.abortWith(Response.status(Response.Status.FORBIDDEN)
          .entity("Usuario obrigatorio!").build());
    }
  }
}
```

```java
package rest.exer14;

import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import javax.ws.rs.client.ClientRequestContext;
import javax.ws.rs.client.ClientRequestFilter;

public class FiltroCliente implements ClientRequestFilter {
  @Override
  public void filter(ClientRequestContext request) throws IOException {
    List<Object> header = new ArrayList<Object>();
    header.add("Fernando");
    request.getHeaders().add("usuario", header);
  }
}
```

```java
package rest.exer14;

import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.core.MediaType;

@Path("/logar")
public class RestAutenticacao {
  @GET
  @Produces(MediaType.TEXT_PLAIN)
  public String logar() {
    return "Bem vindo ao sistema!";
  }
}
```

```java
package rest.exer14;

import java.net.URI;
import javax.ws.rs.core.UriBuilder;
import org.glassfish.grizzly.http.server.HttpServer;
import org.glassfish.jersey.grizzly2.httpserver.GrizzlyHttpServerFactory;
import org.glassfish.jersey.server.ResourceConfig;

public class Servidor {
  public static void main(String[] args) {
    try {
      URI uri = UriBuilder.fromUri("http://localhost/").port(8080).build();
      ResourceConfig config = new ResourceConfig();
      config.packages("rest.exer14");
      HttpServer server = GrizzlyHttpServerFactory.createHttpServer(uri, config);
      System.out.println("servidor no ar teste - " + server);
      // Teste http://localhost:8080/logar
    } catch (Exception e) {
      System.out.println("Erro na execução do servidor JSE - " + e.getMessage());
    }
  }
}
```

```java
package rest.exer14;

import javax.ws.rs.client.Client;
import javax.ws.rs.client.ClientBuilder;
import javax.ws.rs.client.WebTarget;
import javax.ws.rs.core.Response;

public class Cliente {
  public static void main(String[] args) {
    Client cliente = ClientBuilder.newClient();
    // Na primeira vez, execute sem registrar.
    // Na segunda vez, registre e veja liberar o 403
    cliente.register(FiltroCliente.class);
    WebTarget web = cliente.target("http://localhost:8080/logar");
    Response resposta = web.request().get();
    if (resposta.getStatus() == 200) {
      String mensagem = resposta.readEntity(String.class);
      System.out.println(mensagem);
    } else {
      System.out.println("erro na resposta = " + resposta.toString());
    }
    resposta.close();
    cliente.close();
  }
}
```

Exercício 15

```
package rest.exer15;

import java.util.Random;
import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.core.MediaType;

@Path("/numerorandomico")
public class RestNumero {
  @GET
  @Produces(MediaType.TEXT_PLAIN)
  public String processarCadatro() {
    Random numero = new Random();
    return "Numero = " + numero.nextInt(10000);
  }
}
```

```
package rest.exer15;

import java.net.URI;
import javax.ws.rs.core.UriBuilder;
import org.glassfish.grizzly.http.server.HttpServer;
import org.glassfish.jersey.grizzly2.httpserver.GrizzlyHttpServerFactory;
import org.glassfish.jersey.server.ResourceConfig;

public class Servidor {
  public static void main(String[] args) {
    try {
      URI uri = UriBuilder.fromUri("http://localhost/").port(8080).build();
      ResourceConfig config = new ResourceConfig();
      config.packages("rest.exer15");
      HttpServer server = GrizzlyHttpServerFactory.createHttpServer(uri, config);
      System.out.println("servidor no ar - " + server);
      // Teste http://localhost:8080/numerorandomico
    } catch (Exception e) {
      System.out.println("Erro na execução do servidor JSE - " + e.getMessage());
    }
  }
}
```

```java
package rest.exer15;

import java.io.IOException;
import javax.ws.rs.client.ClientRequestContext;
import javax.ws.rs.client.ClientResponseContext;
import javax.ws.rs.client.ClientResponseFilter;

public class FiltroNumero implements ClientResponseFilter {
  @Override
  public void filter(ClientRequestContext requestContext,
      ClientResponseContext response) throws IOException {
    System.out.println("tamanho da resposta - "+ response.getLength());
  }
}
```

```java
package rest.exer15;

import javax.ws.rs.client.Client;
import javax.ws.rs.client.ClientBuilder;
import javax.ws.rs.client.WebTarget;
import javax.ws.rs.core.Response;

public class Cliente {
  public static void main(String[] args) {
    Client cliente = ClientBuilder.newClient();
    cliente.register(FiltroNumero.class);
    WebTarget web = cliente.target("http://localhost:8080/numerorandomico");
    Response resposta = web.request().get();
    if (resposta.getStatus() == 200) {
      String mensagem = resposta.readEntity(String.class);
      System.out.println(mensagem);
    } else {
      System.out.println("erro na resposta = " + resposta.toString());
    }
    resposta.close();
    cliente.close();
  }
}
```

Exercício 16

```java
package rest.exer16;

import java.io.IOException;
import java.util.zip.GZIPOutputStream;
import javax.ws.rs.WebApplicationException;
import javax.ws.rs.ext.Provider;
import javax.ws.rs.ext.WriterInterceptor;
import javax.ws.rs.ext.WriterInterceptorContext;

@Provider
public class GzipServidor implements WriterInterceptor {
  @Override
  public void aroundWriteTo(WriterInterceptorContext ctx)
      throws IOException, WebApplicationException {
    GZIPOutputStream os = new GZIPOutputStream(ctx.getOutputStream());
    ctx.setOutputStream(os);
    ctx.proceed();
    return;
  }
}
```

```java
package rest.exer16;

import javax.xml.bind.annotation.XmlRootElement;

//Declaração de Elemento JAXB para conversão automatica.
@XmlRootElement
public class Telefone {
  private String nome;
  private String fone;
  public Telefone() {
  }
  public Telefone(String nome, String fone) {
    super();
    this.nome = nome;
    this.fone = fone;
  }
  public String getNome() {
    return nome;
  }
  public void setNome(String nome) {
    this.nome = nome;
  }
  public String getFone() {
    return fone;
  }
  public void setFone(String fone) {
    this.fone = fone;
  }
}
```

```java
package rest.exer16;

import java.util.ArrayList;
import java.util.List;
import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.core.MediaType;

@Path("/telefones")
public class ListaTelefonica {
  @GET
  @Produces(MediaType.APPLICATION_JSON)
  public List<Telefone> processarCadastro() {
    List<Telefone> lista = new ArrayList<>();
    for (int i = 0; i < 1000; i++) {
      lista.add(new Telefone("Nome " + i, "3325-750" + i));
    }
    return lista;
  }
}
```

```java
package rest.exer16;

import java.net.URI;
import javax.ws.rs.core.UriBuilder;
import org.glassfish.grizzly.http.server.HttpServer;
import org.glassfish.jersey.grizzly2.httpserver.GrizzlyHttpServerFactory;
import org.glassfish.jersey.server.ResourceConfig;

public class Servidor {
  public static void main(String[] args) {
    try {
      URI uri = UriBuilder.fromUri("http://localhost/").port(8080).build();
      ResourceConfig config = new ResourceConfig();
      config.packages("rest.exer16");
      HttpServer server = GrizzlyHttpServerFactory.createHttpServer(uri, config);
      System.out.println("servidor no ar - " + server);
      // Teste http://localhost:8080/telefones
    } catch (Exception e) {
      System.out.println("Erro na execução do servidor JSE - " + e.getMessage());
    }
  }
}
```

Exercício 17

```java
package rest.exer17;

import java.io.IOException;
import java.io.InputStream;
import java.util.zip.GZIPInputStream;

import javax.ws.rs.WebApplicationException;
import javax.ws.rs.ext.ReaderInterceptor;
import javax.ws.rs.ext.ReaderInterceptorContext;

public class GzipCliente implements ReaderInterceptor {
  @Override
  public Object aroundReadFrom(ReaderInterceptorContext context) throws IOException,
      WebApplicationException {
    InputStream originalInputStream = context.getInputStream();
    context.setInputStream(new GZIPInputStream(originalInputStream));
    return context.proceed();
  }
}
```

```java
package rest.exer17;

import java.util.List;
import javax.ws.rs.client.Client;
import javax.ws.rs.client.ClientBuilder;
import javax.ws.rs.client.WebTarget;
import javax.ws.rs.core.GenericType;
import javax.ws.rs.core.Response;
import rest.exer16.Telefone;

public class Cliente {
  public static void main(String[] args) {
    Client cliente = ClientBuilder.newClient();
    // Execute a primeira vez e veja que vai dar exception pq o conteudo vem zipado.
    cliente.register(GzipCliente.class);
    WebTarget web = cliente.target("http://localhost:8080/telefones");
    Response resposta = web.request().get();
    List<Telefone> telefones = resposta.readEntity(new GenericType<List<Telefone>>() {
    });
    for (Telefone t : telefones) {
      System.out.println(t.getNome() + " - " + t.getFone());
    }
    resposta.close();
  }
}
```

Exercício 18

```
package rest.exer18;

import java.net.URI;
import javax.ws.rs.core.UriBuilder;
import org.glassfish.grizzly.http.server.HttpServer;
import org.glassfish.jersey.grizzly2.httpserver.GrizzlyHttpServerFactory;
import org.glassfish.jersey.server.ResourceConfig;

public class ServidorPostman {
  public static void main(String[] args) {
    try {
      URI uri = UriBuilder.fromUri("http://localhost/").port(8080).build();
      ResourceConfig config = new ResourceConfig();
      config.packages("rest.exer1");
      config.packages("rest.exer5");
      HttpServer server = GrizzlyHttpServerFactory.createHttpServer(uri, config);
      System.out.println("servidor no ar teste - " + server);
      // teste http://localhost:8080/application.wadl = veja que aparece todos os serviços
      // Teste http://localhost:8080/ola
      // Teste http://localhost:8080/funcionario/xml
      // Teste http://localhost:8080/funcionario/json
    } catch (Exception e) {
      System.out.println("Erro na execução do servidor JSE - " + e.getMessage());
    }
  }
}
```

Exercício 19

1. dezipar tomcat e criar uma pasta.
2. adicionar tomcat no eclipse.
3. criar projeto web - jee-rest

4. adicionar pom ou jars

```
 <dependencies>
    <dependency>
       <groupId>org.glassfish.jersey.containers</groupId>
       <artifactId>jersey-container-servlet</artifactId>
       <version>2.25.1</version>
    </dependency>

    <dependency>
       <groupId>org.glassfish.jersey.media</groupId>
       <artifactId>jersey-media-moxy</artifactId>
       <version>2.25.1</version>
    </dependency>

    <dependency>
       <groupId>org.glassfish.jersey.media</groupId>
       <artifactId>jersey-media-json-jackson</artifactId>
       <version>2.25.1</version>
    </dependency>

    <!-- grizzly2 -->
    <dependency>
       <groupId>org.glassfish.jersey.containers</groupId>
       <artifactId>jersey-container-grizzly2-http</artifactId>
       <version>2.25.1</version>
    </dependency>
 </dependencies>
```

5. criar html index

```
<!DOCTYPE html>
<html>
<head>
   <title>Insert title here</title>
</head>
<body>
   Projeto web no ar...
   http://localhost:8080/jee-rest/rest/application.wadl
</body>
</html>
```

6. adicionar web.xml conf Jersey

```
<servlet>
   <servlet-name>Jersey Web Application</servlet-name>
   <servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-class>
   <init-param>
      <param-name>jersey.config.server.provider.packages</param-name>
      <param-value>rest.exer1, rest.exer2</param-value>
   </init-param>
   <init-param>
      <param-name>com.sun.jersey.api.json.POJOMappingFeature</param-name>
      <param-value>true</param-value>
   </init-param>
   <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
   <servlet-name>Jersey Web Application</servlet-name>
   <url-pattern>/rest/*</url-pattern>
</servlet-mapping>
```

7. copiar src do outro projeto
8. rodar e testar
 - http://localhost:8080/jee-rest/rest/ola
 - http://localhost:8080/jee-rest/rest/calculadora?v1=5&v2=5