Introducción a:

Java EE 7 & HTML 5

Eudris Cabrera Rodríguez

@eudriscabrera @eudriscabrera

#JavaEE7

26 Abril 2014, Santo Domingo, R. D.

Code Camp SQD 2014

Sobre Eudris Cabrera

- Desarrollador Informático / Consultor en PAFI

  (Programa de Administración Financiera Integrada / Ministerio de Hacienda).
- Desarrollador Java EE / SE, consultor y a veces entrenador en Java desde hace más de 7 años.
- Entusiasta de la tecnología y software libre.

Comunidades

- Github: @ecabrerar
- Google Groups: @letsrockjava

Agenda

- Breve Reseña sobre Java
- Conceptos Generales sobre Java EE
- Aspectos importantes de Java EE 7
- Primeros pasos con Java EE 7
- Cómo empezar con Java EE 7
- Algunos APIs de Java EE7

Objetivos

Destacar los aspectos más novedosos que ofrece Java EE 7 para el desarrollo de aplicaciones empresariales, enfocando el soporte para HTML5 y otras tecnologías relacionadas.

Breve reseña sobre Java

Entendiendo el ecosistema Java

- Plataforma Java:
  - Multi-plataforma.
  - Utiliza una máquina virtual para su ejecución (JVM)
  - Esta dividida en:
    - **Java SE**
    - **Java EE**
    - **Java ME**
    - **Javafx**
  - El estandar es manejado por Java Community Process (JCP)

James Gosling,
Creador de Java

Popularidad del lenguaje Java

- TIOBE Index lo sitúa en el segundo lugar para el mes de Abril del 2014, muy cercano a Lenguaje C, que ocupa el primer lugar.

  http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html

- Hoy en día, la tecnología Java ya está presente en 5 mil millones de tarjetas SIM y tarjetas inteligentes, 3 mil millones de teléfonos móviles, 80 millones de dispositivos de TV, incluyendo Blu-ray, printers, maquinas bancarias, eBooks Reader y Carros.

- Java SE 8 liberado en marzo del 2014, es la plataforma incluye la actualización más grande para el modelo de programación Java desde el lanzamiento de la plataforma en 1996.

Java Platform, Enterprise Edition (EE)

- Es un entorno independiente de la plataforma centrado en Java para desarrollar, crear e implementar en línea aplicaciones empresariales basadas en web.

- Es el estándar en software empresarial.

- Se apoya ampliamente en componentes de software modulares ejecutándose sobre un servidor de aplicaciones.

Java Community Process

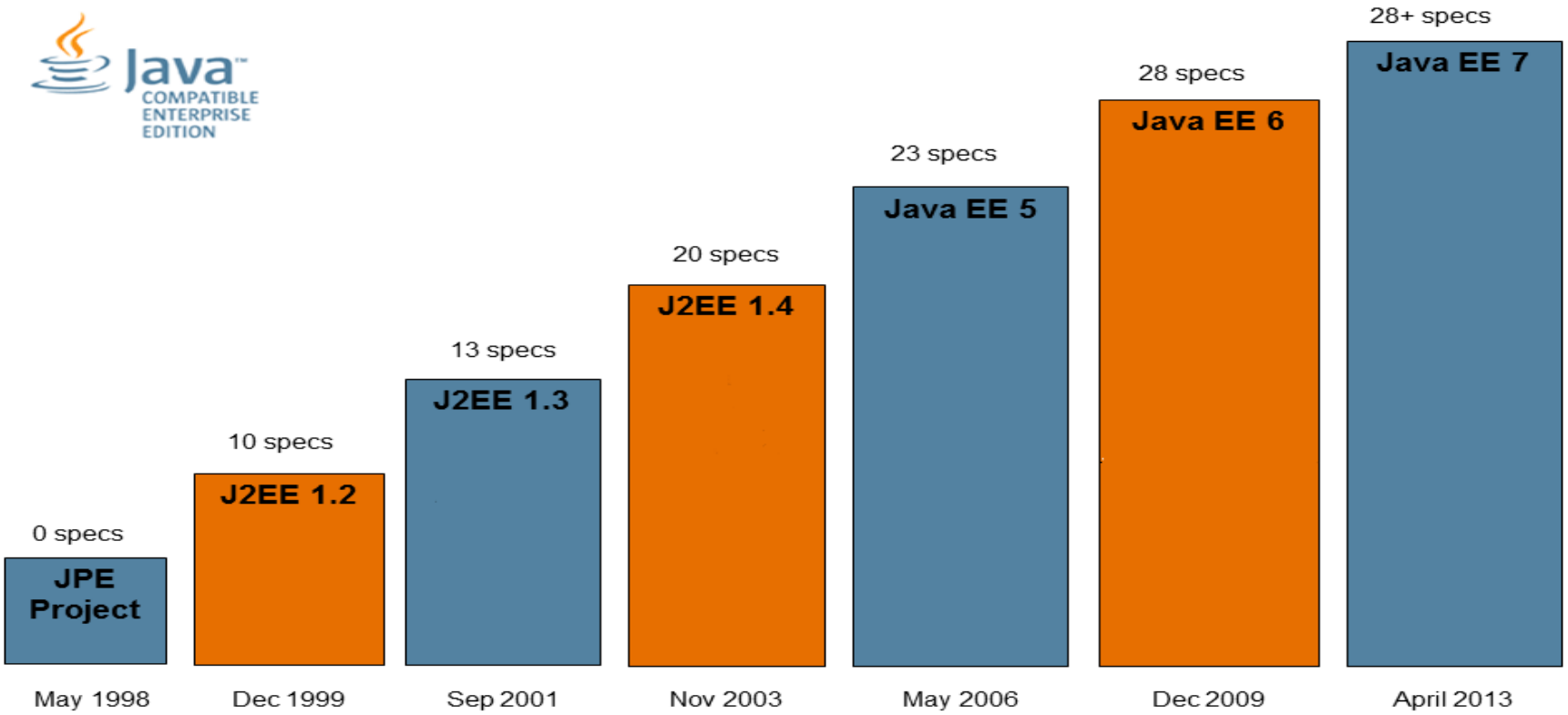
Diferencia entre JAVA SE y JAVA EE

JAVA SE es la versión estándar de java. Es la api base del lenguaje mientras que **JAVA EE** podríamos decir que es una versión extendida de **JAVA SE.**

La plataforma Java EE consta de un conjunto de servicios, API y protocolos que proporcionan la funcionalidad necesaria para desarrollar aplicaciones basadas en web de varios niveles.

Java EE simplifica el desarrollo de aplicaciones y reduce la necesidad de programación, al proporcional componentes modulares normalizados y reutilizables, así como al permitir controlar muchos aspectos de la programación automáticamente por nivel.

Evolución de Java EE

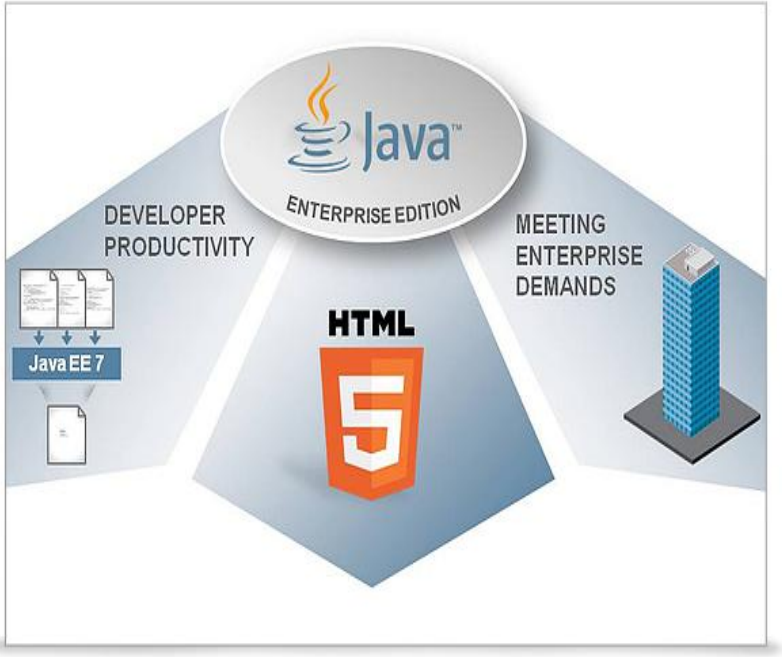


Java EE 6 cambió el juego ....

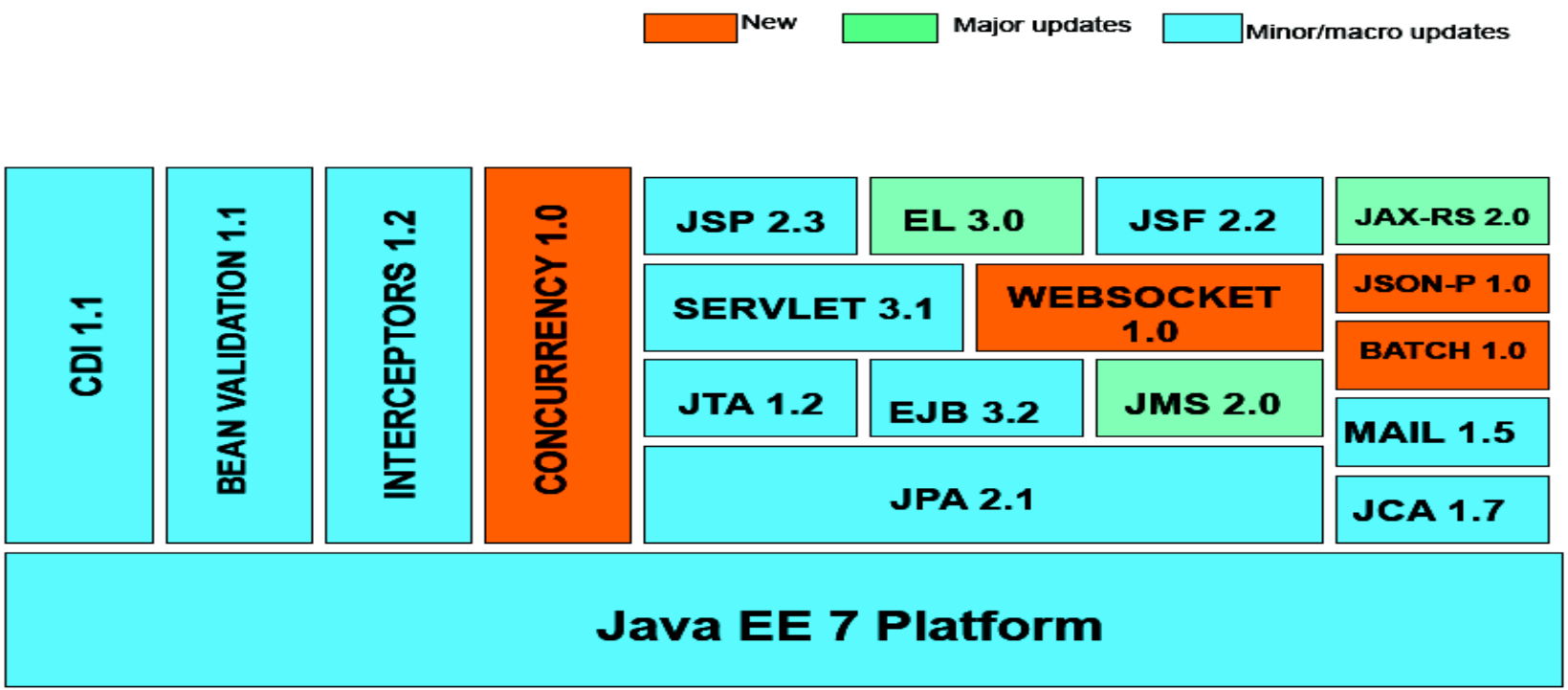
- Más Ligero.
- Introduce el concepto de perfiles (**Web Profile** and **Full Profile**).
- **EJB** empaquetado en archivos war.
- **Servlet 3.0**
  - web.xml (opcional), **@WebServlet**, **@WebFilter**
- Soporte para servicios web **RESTful** con **JAX-RS 1.1**
- Contextos e Inyección de Dependencia (CDI) para Java EE

Java EE 7 productividad y enfocado en HTML5

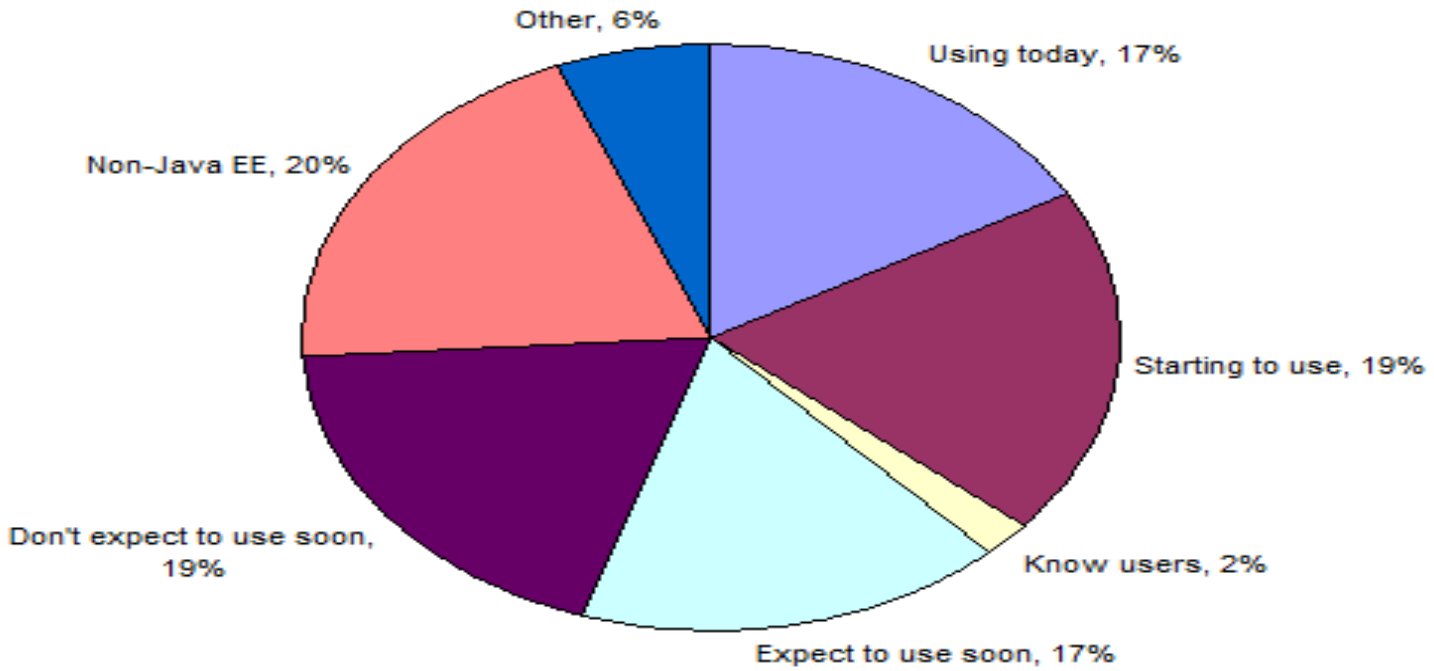
- Construido sobre la base de **Java EE6.**
- Soporte para **HTML5.**
- **4 nuevas** especificaciones.
- **3** especificaciones con **cambios importantes.**
- **6** especificaciones con **cambios menores.**
- **5** especificaciones con **cambios micro.**



Plataforma Java EE 7



Adopción de Java EE 7



https://blogs.oracle.com/theaquarium/entry/survey_sez_java_ee_71

Cómo empezar con Java EE 7 ?

Descargar e instalar:

JDK 7(ó superior):
http://www.oracle.com/technetwork/java/javase/downloads/index.html

Instalar el IDE de su preferencia:

NetBeans 7.4 o superior, versión completa o "**Java EE**"
http://netbeans.org/downloads/

Eclipse 4.3 (Kepler) o superior
http://www.eclipse.org/kepler/

Otros

Soporte para HTML5 y tecnologías relacionadas

JavaServer Faces 2.2 (JSR 344)

- Tecnología y Framework para el desarrollo de aplicaciones web en Java.
- Incluye:
  - APIs para el manejo de eventos, validar entradas, esquema de control de navegación
  - Administración de estados.
  - Basado en componentes
  - Eventos gestionados desde el servidor
- Disponible desde la versión 1.0, 2004 en Java EE
- Existen librerías tipo extensiones para el manejo de componentes visuales:
  - RichFaces
  - ICEFaces
  - PrimeFaces

JavaServer Faces 2.2 (JSR 344)

- **¿Cómo trabaja JSF?**
  - Utiliza taglib los cuales están asociados a clases manejadoras.
  - Todas las etiquetas son procesadas y presentada mediante HTML, mapeando cada etiqueta con su representación en el server. **Codificación**
  - Cada petición es manejada vía POST y **decodificado** los valores para ser procesadas.

JavaServer Faces 2.2 (JSR 344)

- **Managed Beans**
  - Representan la separación de la vista con la regla de negocio.
  - Son componentes reusables.
  - Facilitan el procesamiento de la información desde el formulario al servidor y viceversa.

JavaServer Faces 2.2 (JSR 344)

- **Managed Beans**

```
 */
@ManagedBean(name = "miFormulario")
@SessionScoped
public class MiFormularioBean implements Serializable{

    @Size(max = 5)
    private String nombre;
    private String apellido;
    int contador;

    public String procesarDatosForm(){
        /**
         * Controlar cualquier cosa que sea neceario en su procesamiento...
         */
        nombre=nombre.toUpperCase();
        //
        return "resultado?faces-redirect=true";
    }

    public int getContador() {
        return contador;
    }

    public void setContador(int contador) {
        this.contador = contador;
    }
```

Definición

Propiedades

Set & Get

JavaServer Faces 2.2 (JSR 344)

- **JSF 2.2**
  - Pertenece al JEE 7.
  - Cambio de espaciado de nombre.
  - Incluye cambios en los siguiente aspecto:
    - Soporte HTML5.
    - Componente File Upload.
    - Faces Flow.
    - Protección sobre Cross Site Request Forgery
    - Multi-Templating.

JavaServer Faces 2.2 (JSR 344)

- **JSF 2.2 - File Upload**
  - Incluye la etiqueta h:inputFile. Debe estar dentro de un form con el enctype “multipart/form-data”
  - La propiedad del Bean es del tipo javax.servlet.http.Part.

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html">
    <h:head>
        <title>JSF 2.2</title>
    </h:head>
    <h:body>
        Subir Archivos
        <h:form enctype="multipart/form-data">
            <h:inputFile value="#{Form.archivo}"/>
            <h:commandButton action="#{Form.upload()}" value="Subir Archivo"/>
        </h:form>
    </h:body>
</html>
```

JavaServer Faces 2.2 (JSR 344)

- **JSF 2.2 - File Upload**

```
 */
@Named("Form")
@SessionScoped
public class MiForm implements Serializable{
    private Part archivo;

    public String upload() throws IOException {
        //Obteniendo el archivo.
        FileOutputStream fOut=new FileOutputStream("/tmp/mi_archivo");
        int read=0;
        byte[] data=new byte[1024];
        InputStream in=archivo.getInputStream();

        while((read = in.read(data)) != -1){
            fOut.write(data, 0, read);
        }

        return null;
    }

    public void setArchivo(Part archivo) {
        this.archivo = archivo;
    }

    public Part getArchivo() {
        return archivo;
    }


}
```

JavaServer Faces 2.2 (JSR 344)

- **JSF 2.2 - Soporte HTML5**

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:f="http://xmlns.jcp.org/jsf/core"
      xmlns:h="http://xmlns.jcp.org/jsf/html"
      xmlns:p="http://xmlns.jcp.org/jsf/passthrough"
      xmlns:jsf="http://xmlns.jcp.org/jsf">
```

```
<p>
    <label>Email:</label>
    <br/>
    <input type="email" jsf:id="email" name="email" value="#{formularioBean.email}" required="required"/>
</p>
<p>
    <label>Fecha de Llegada:</label>
    <br/>
    <input type="date" jsf:id="fecha" name="fecha" value="#{formularioBean.fecha}" required="required"/>
</p>
<p>
    <progress jsf:id="progressBar" max="100" value="0"/>
</p>
```

Html5

Html 5

identificador Jsf

JSF HTML 5 DEMO

RESTful Web Services (JSR 339)

JAX-RS 1.0:

- Liberada en el 2008 para la versión Java EE 6. JSR-311
- Simplifica el proceso de creación de servicios Web mediante Plain Old Java Objects (POJOS) y anotaciones.
- No requiere configuración adicional para utilizarlos en entornos Java EE 6.
- Incluye anotaciones para implementar servicios Web:
  - @Path, @Get, @Put, @Post, @Delete, @Produces, @Consumes, entre otros.

RESTful Web Services (JSR 339)

Ejemplo Clase JAX-RS

```java
//Omitiendo los import javax.ws.rs.*;
/**
 * REST Web Service
 *
 * @author vacax
 */
@Path("restful")
public class MiServicioRestful {

    @Context
    private UriInfo context;
    @EJB
    private MiEJB miEJB;

    public MiServicioRestful() {
    }

    /**
     * Es ejecutado mediante la URL $contexto/webresources/restful y retorna
     * una lista de estudiantes en formato JSON.
     */
    @GET
    @Produces("application/json")
    public List<Estudiante> getListadoEstudiantes() {
        System.out.println("Recuperando lista de estudiantes");
        return miEJB.getListaEstudiantes();
    }

}
```

RESTful Web Services (JSR 339)

- **JAX-RS 2.0:**
  - Introduce elementos que ayudan a la productividad.
  - Simplifica el API
  - Incluye los siguientes aspectos:
    - API para el Cliente
    - Llamadas Asincrónicas vía Http
    - Filtros e interceptores

RESTful Web Services (JSR 339)

- **API para el Cliente**

```
Client client = ClientFactory.newClient();
 WebTarget target = client.target("http://example.com/shop");
Form form = new Form().param("customer", "Bill")
                      .param("product", "IPhone 5")
                      .param("CC", "4444 4444 4444 4444");

Response response = target.request().post(Entity.form(form));
assert response.getStatus() == 200;

Order order = response.readEntity(Order.class);
```

RESTful Web Services (JSR 339)

- **Llamadas Asincrónicas vía Http (API Cliente Asincrónico)**

```
InvocationCallback<Response> callback = new InvocationCallback {
   public void completed(Response res) {
      System.out.println("Request success!");
   }

   public void failed(ClientException e) {
      System.out.println("Request failed!");n
   }
};
client.target("http://example.com/customers").queryParam("name", "Bill Burke")
      .request().async()
      .get(callback);
```

RESTful Web Services (JSR 339)

- **Llamadas Asincrónicas vía Http (Server-side Asincrónico Http)**

```
@Path("/listener")
public class ChatListener{

   List<AsyncResponse> listeners = ...some global list...;


   @GET
   public void listen(@Suspended AsyncResponse res) {
      list.add(res);
   }

}
```

RESTful Web Services (JSR 339)

- **Filtros**

```
public class HeaderLoggingFilter implements ClientRequestFilter,
ClientResponseFilter {
    // from ClientRequestFilter
    public void filter(ClientRequestContext crc) throws
IOException {
        for (Entry e : crc.getHeaders().entrySet()) {
            … = e.getKey();
            … = e.getValue();
        }
    }

    // from ClientResponseFilter
    public void filter(ClientRequestContext crc,
ClientResponseContext crc1) throws IOException {
        ...
    }
}
```

RESTful Web Services (JSR 339)

- **Interceptores**

```
@Provider
//Codifica una salida con GZIP.
public class GZIPEndoer implements WriterInterceptor {
   public void aroundWriteTo(WriterInterceptorContext ctx) throws IException,
WebApplicationException {
      GZIPOutputStream os = new GZIPOutputStream(ctx.getOutputStream());
      try {
         ctx.setOutputStream(os);
         return ctx.proceed();
      } finally {
         os.finish();
      }
   }
}
```

RESTful Web Services Demo

Java API for JSON Processing 1.0 (JSR 353)

JSR 353 es el API de Java para procesamiento JSON (JSON-P) y define un API para el proceso (por ejemplo, análisis, generar, transformar y consulta) JSON.

Este JSR forma parte de Java EE 7.

El API permite producir y consumir JSON de manera secuencial (equivalente a StAX en el mundo XML) y construir un modelo de objetos de Java para JSON (equivalente a DOM en el mundo XML)

Java API for JSON Processing 1.0 (JSR 353)

Puntos importantes del API.

Basados en DOM.

JsonBuilder - Construye un objeto JSON o un arreglo JSON

JsonReader - Lee un objeto JSON o un arreglo

JsonWriter - Escribe un objeto JSON o un arreglo

Streaming APIs

JsonGenerator

JsonParser

Java API for JSON Processing 1.0 (JSR 353)

Sintaxis JSON

```
{
   "firstName": "Duke",
   "lastName": "Java",
   "age": 18,
   "streetAddress": "100 Internet Dr",
   "city": "JavaTown",
   "state": "JA",
   "postalCode": "12345",
   "phoneNumbers": [
      { "Mobile": "111-111-1111" },
      { "Home": "222-222-2222" }
   ]
}
```

Java API for JSON Processing 1.0 (JSR 353)

Construir objeto JSON con JsonBuilder

```
JsonObject value = new JsonBuilder()
    .beginObject()
      .add("firstName", "John")
      .add("lastName", "Smith")
      .add("age", 25)
        .beginObject("address")
          .add("streetAddress", "21 2nd Street")
          .add("city", "New York")
          .add("state", "NY")
          .add("postalCode", "10021")
        .endObject()
        .beginArray("phoneNumber")
          .beginObject()
            .add("type", "home")
            .add("number", "212 555-1234")
          .endObject()
          .beginObject()
            .add("type", "home")
            .add("number", "646 555-4567")
          .endObject()
        .endArray()
    .endObject()
.build();
```

Java API for JSON Processing 1.0 (JSR 353)

Ejemplo de uso de JsonReader

```
String json = "...";
JsonReader reader = new JsonReader(new StringReader(json));
JsonValue value = reader.readObject();
reader.close();
```

Java API for JSON Processing 1.0 (JSR 353)

Ejemplo de uso de JsonWriter

```
JsonWriter jsonWriter = new JsonWriter(new FileWriter(...));
JsonObject jsonObject = new JsonBuilder()
 .beginObject()
 . . .
 .endObject()
 .build();
jsonWriter.writeObject(jsonObject);
jsonWriter.close();
```

Java API for JSON Processing 1.0 (JSR 353)

Escribiendo en un archivo usando JsonGenerator

```
FileWriter writer = new FileWriter("test.txt");
JsonGenerator gen = Json.createGenerator(writer);
gen.writeStartObject()
   .write("firstName", "Duke")
   .write("lastName", "Java")
   .write("age", 18)
   .write("streetAddress", "100 Internet Dr")
   .write("city", "JavaTown")
   .write("state", "JA")
   .write("postalCode", "12345")
   .writeStartArray("phoneNumbers")
      .writeStartObject()
         .write("type", "mobile")
         .write("number", "111-111-1111")
      .writeEnd()
      .writeStartObject()
         .write("type", "home")
         .write("number", "222-222-2222")
      .writeEnd()
   .writeEnd()
.writeEnd();
gen.close();
```

Java API for JSON Processing 1.0 (JSR 353)

Leyendo un archivo con formato JSON utilizando JsonParser

```
JsonParser parser = Json.createParser(new StringReader(jsonData));
while (parser.hasNext()) {
   JsonParser.Event event = parser.next();
   switch(event) {
      case START_ARRAY:
      case END_ARRAY:
      case START_OBJECT:
      case END_OBJECT:
      case VALUE_FALSE:
      case VALUE_NULL:
      case VALUE_TRUE:
         System.out.println(event.toString());
         break;
      case KEY_NAME:
         System.out.print(event.toString() + " " +
                          parser.getString() + " - ");
         break;
      case VALUE_STRING:
      case VALUE_NUMBER:
         System.out.println(event.toString() + " " +
                            parser.getString());
         break;
   }
}
```

JSON API Demo

Java API for WebSocket 1.0 (JSR 356)

WebSocket:
Es un nuevo protocolo derivado de HTTP.

HTTP es el protocolo estándar para la Web, es muy efectivo para una gran cantidad de casos de uso pero, sin embargo, tiene algunos inconvenientes en el caso de aplicaciones Web interactivas.

Java API for WebSocket 1.0 (JSR 356)

Desventajas de HTTP:

Half-duplex: basado en el modelo de solicitud / respuesta.

Verbose: una gran cantidad de información se envía en las cabeceras HTTP asociados con el mensaje.

Con el fin de añadir un modo de inserción en el servidor, es necesario utilizar solución (**poll**, **long poll**, **Comet / Ajax**), ya que no existe un estándar.

Este protocolo no está optimizado a escala en aplicaciones de gran tamaño para comunicación en tiempo real bidireccional.

Java API for WebSocket 1.0 (JSR 356)

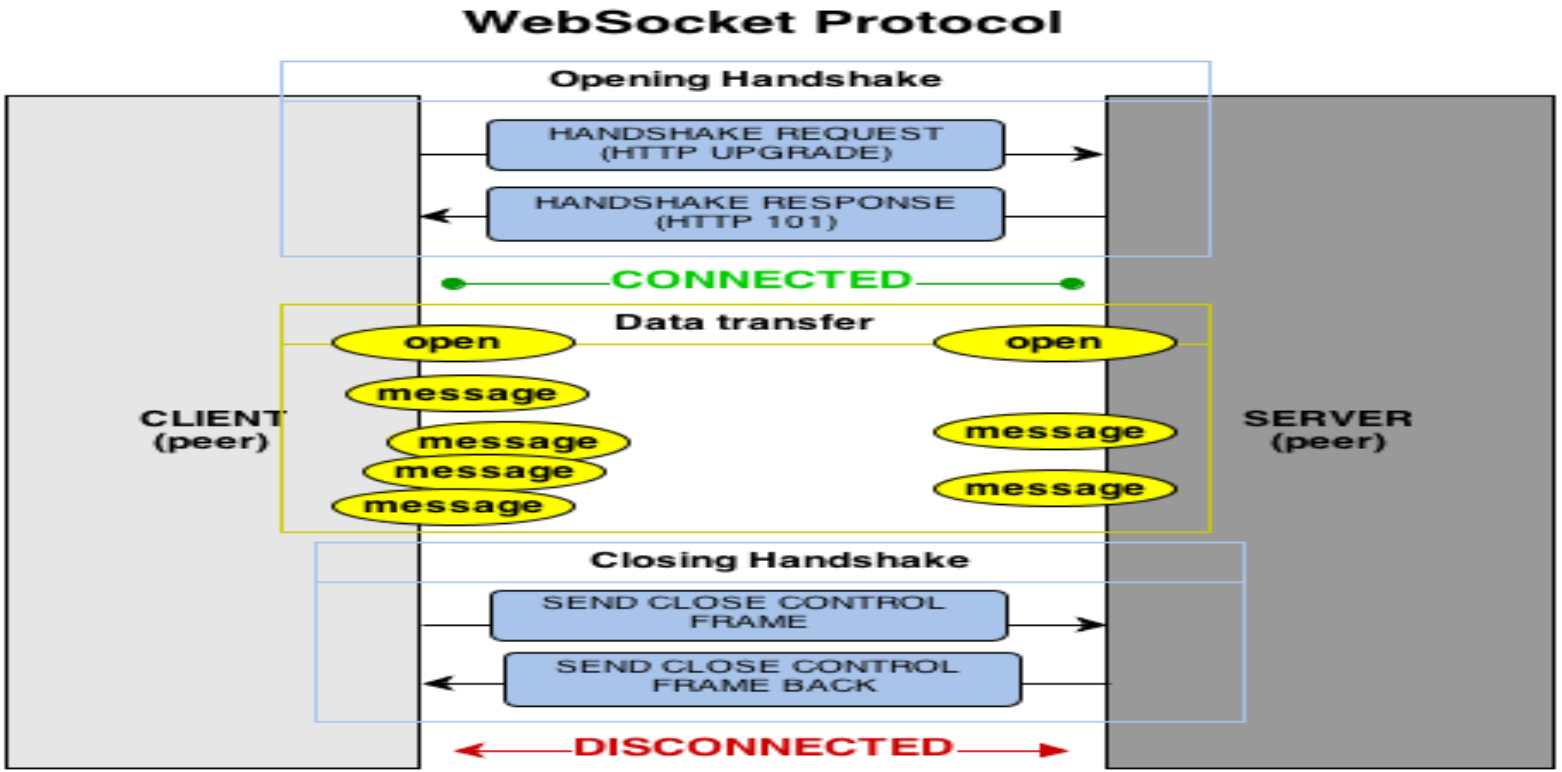
El nuevo protocolo WebSocket ofrece funciones más avanzadas de HTTP porque es:

Basado en 1 conexión TCP única entre 2 pares (mientras que cada solicitud / respuesta HTTP necesita una nueva conexión TCP)

Bidireccional: cliente puede enviar el mensaje al servidor y el servidor también puede enviar un mensaje al cliente

Full-duplex: cliente puede enviar varios mensajes al servidor, así como el servidor al cliente sin esperar una respuesta de sí

Java API for WebSocket 1.0 (JSR 356)



Java API for WebSocket 1.0 (JSR 356)

WebSocket Server Endpoint

```
@ServerEndpoint(value="/chat", encoders=ChatMessageEncoder.class,
decoders=ChatMessageDecoder.class)
public class ChatServer {
    @OnMessage
    public String receiveMessage(ChatMessage message, Session client)
{
       for (Session s : client.getOpenSessions()) {
           s.getBasicRemote().sendText(message);
       }
    }
}
```

Java API for WebSocket 1.0 (JSR 356)

WebSocket Client

```
WebSocketContainer container = ContainerProvider.
getWebSocketContainer();
String uri = "ws://localhost:8080/chat/websocket";
container.connectToServer(ChatClient.class, URI.create(uri));
```

Websocket Demo

Código fuente de los ejemplos en GitHub

https://github.com/ecabrerar/codecampsdq40-javaee7

Java EE 7 como toda tecnología, esta es tan buena como la disposición que tenga el usuario a hacer el mejor uso de ella.

Anónimo.

¡Gracias por acompañarnos!

Referencias

Todas las marcas registradas, así como todos los logotipos, imágenes, fotografías, audio y vídeos mostrados en esta presentación son propiedad de sus respectivos propietarios y/o representantes.

Su utilización es solamente para fines ilustrativos.

Enlaces:

http://www.java.com/es/download/faq/techinfo.xml
https://javaee7.zeef.com/arjan.tijms
http://mgreau.com/posts/2013/11/11/javaee7-websocket-angularjs-wildfly.html
http://java.dzone.com/articles/whats-new-jax-rs-20
http://www.infoq.com/news/2013/06/Whats-New-in-JAX-RS-2.0

@eudriscabrera

@eudris

@ecabrerar

@eudriscabrera

Eudris Cabrera Rodríguez

Ingeniero Telemático

Desarrollador de Software / Consultor Informático

eudris@gmail.com