	UNIVERSIDAD FRANCISCO DE PAULA SANTANDER **FACULTAD DE INGENIERÍA** **PROGRAMA DE INGENIERIA DE SISTEMAS** **Registro Actividades Independiente**	

Asignatura	**Programación Web**
Código	**1155606**
Tipo de asignatura:	**Obligatoria – 6 Semestre**
Nombre del Profesor	**Carlos René Angarita Sanguino** (carlosreneas@ufps.edu.co)

- Semestre: Definida en la plataforma
- Fecha máxima de Entrega: Definida en la plataforma
- Horas de trabajo: 4 horas

INFORMACIÓN DE LA ACTIVIDAD

Competencias a desarrollar:

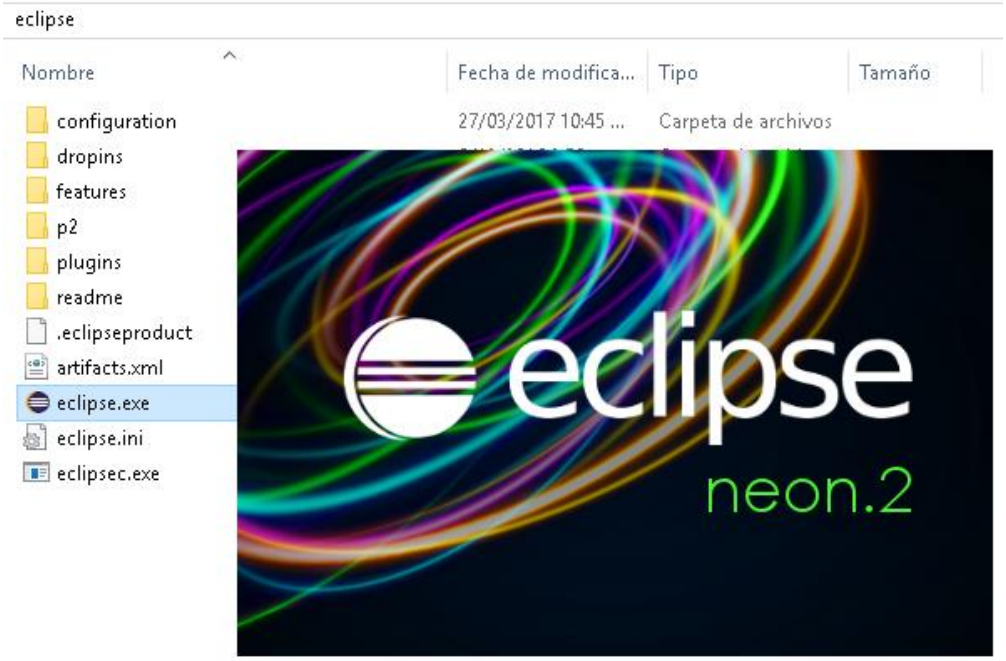
- Conocer el entorno de eclipse con la perspectiva Web
- Desarrolla páginas JSP.

Tipo de Actividad: Teórica ( ) Práctica (X)

Vamos a crear un proyecto nuevo en Eclipse.

Iniciando el entorno

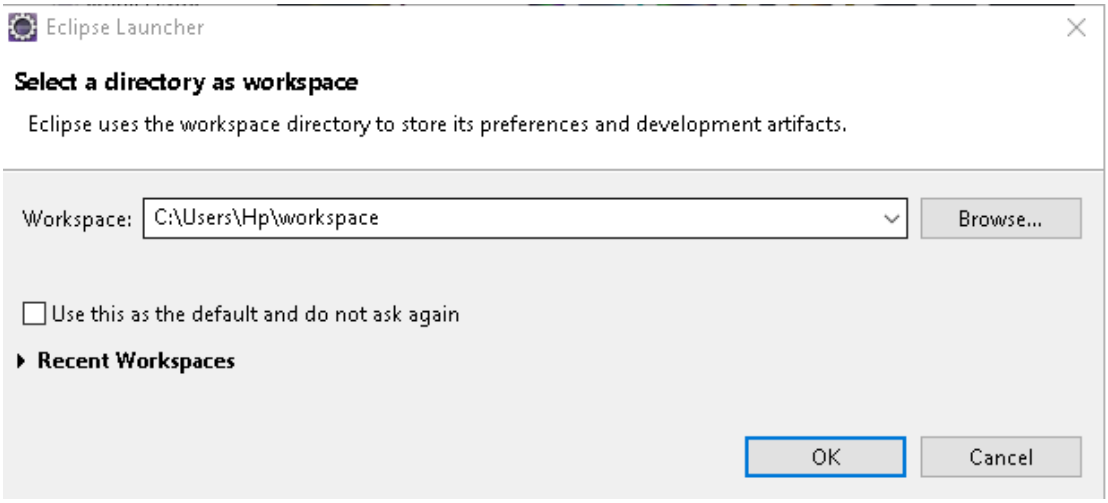
Ingresemos a Eclipse, en este caso es la versión NEON.



Al ingresar a Eclipse el sistema nos solicita un entorno sobre el cual trabajaremos nuestros proyectos.

A este entorno sobre el cual trabajamos se le denomina "workspace". Y tiene la raíz en un

directorio del filesystem. El eclipse entonces se puede abrir en diferentes momentos para trabajar sobre diferentes workspaces. Por ejemplo, podrían tener un workspace para cada materia, y sin embargo utilizar el mismo eclipse. Y hasta pueden ejecutar el mismo eclipse simultáneamente sobre distintos workspaces. El workspace entonces contiene proyectos. Que pueden ser proyectos java, o de cualquier tipo que esté soportado por su eclipse (recordar que decíamos que el eclipse se puede extender para soportar otros lenguajes).

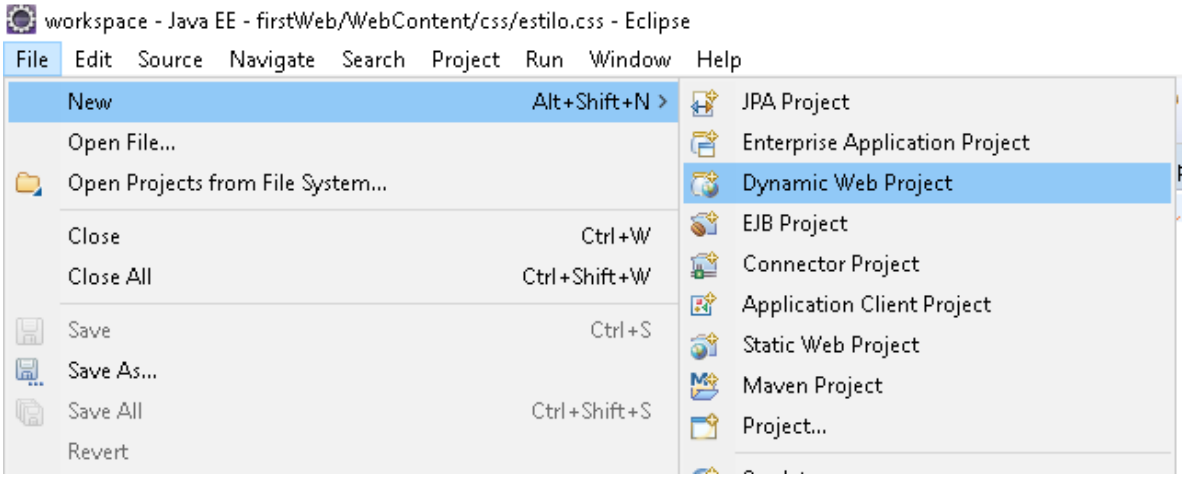


Creando el primer proyecto

Ahora ya con el IDE abierto podemos proceder a realizar nuestro primer ejercicio. Como primera opción debemos crear el proyecto. Para este caso que es orientado a la Web, crearemos un proyecto *Dynamic Web Project*.

→File →New →Dynamic Web Project

Como se observa en la siguiente imagen.

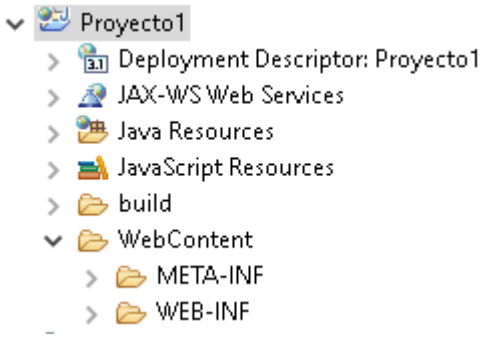


¿A que corresponden las Servlet API Versión?

Al crear el proyecto se tienen tres ventadas. La primera permite configurar el Proyecto. Dentro de las opciones importantes en la siguiente ventana se tiene el nombre del proyecto (*Project Name*), el Target Runtime y Dynamic Web Module. El primero corresponde al nombre, el segundo al contenedor de servlets que se encargara de desplegar las peticiones y el último corresponde a la versión del API Servlet.

La siguiente ventana es de Java y me permite configurar y vincular distintos folder.

La última configura las opciones del módulo web. LA raíz de la aplicación, directorio raíz. Por ahora todos los datos son por defecto.



A continuación podemos ver la estructura básica creada en nuestro proyecto:

/WebContent - Directorio raíz de la aplicación web, en el cual se colocan todos los archivos (JSP, HTML, imágenes, hojas de estilo, etc.) que utiliza la aplicación. Se pueden crear subdirectorios adicionales para mantener cualquier otro recurso de tipo estático que forme parte de la aplicación web y constituyan la parte de acceso público desde cualquier navegador.

/WebContent/WEB-INF - Directorio que contiene todos los recursos relacionados con la aplicación web que no se han colocado en el directorio raíz y que no deben servirse al cliente. Esto es importante, ya que este directorio no forma parte del documento público, por lo que ninguno de los ficheros que contenga va a poder ser enviado directamente a través del servidor web. En este directorio se coloca el archivo web.xml, donde se establece la configuración de la aplicación web.

/WebContent/WEB-INF/clases - Directorio que contiene todos los servlets y cualquier otra clase de utilidad o complementaria que se necesite para la ejecución de la aplicación web. Normalmente contiene solamente archivos .class.

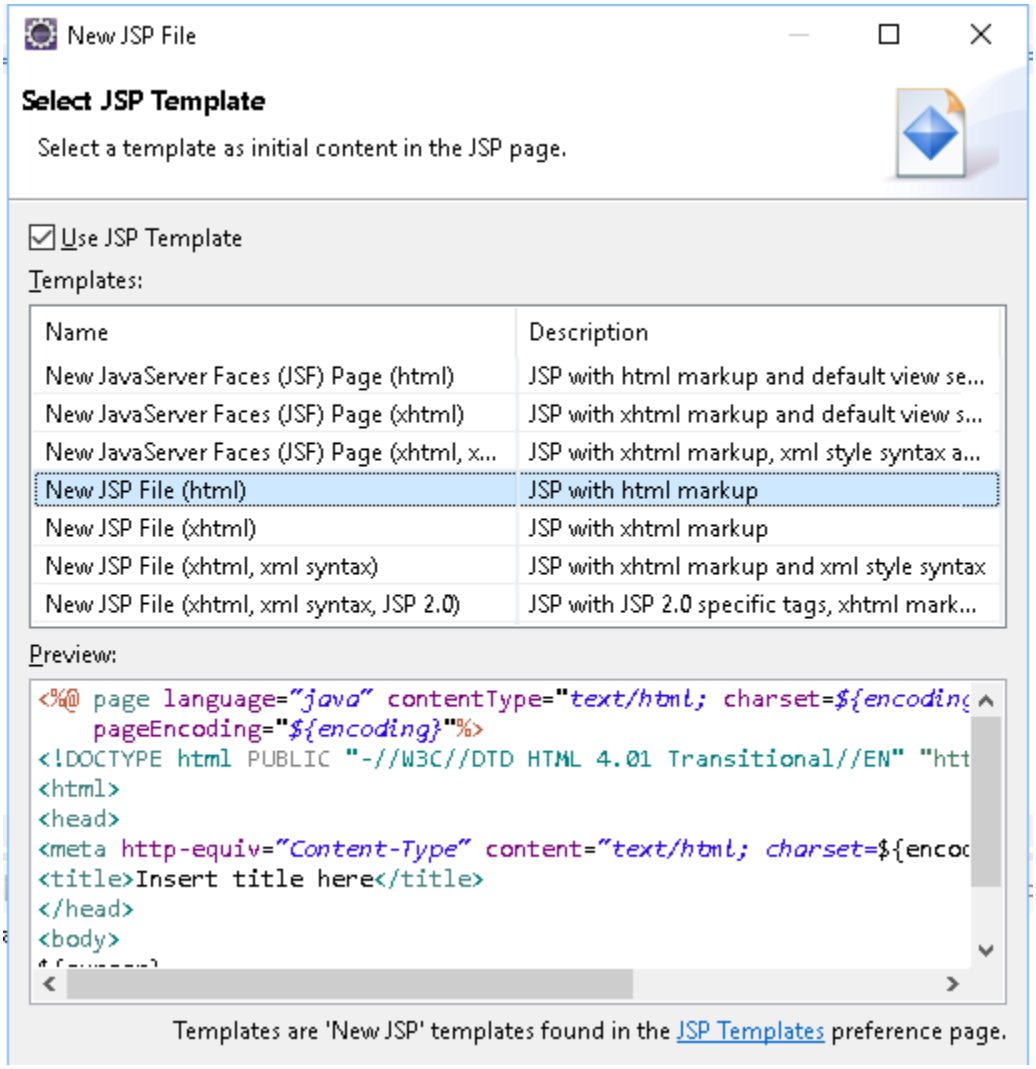
/WebContent/WEB-INF/lib - Directorio que contiene los archivos Java de los que depende la aplicación web. Por ejemplo, si la aplicación web necesita acceso a base de datos a través de JDBC, en este directorio es donde deben colocarse los ficheros JAR que contengan el driver JDBC que proporcione el acceso a la base de datos. Normalmente contiene solamente archivos .jar.

Creando la primera pagina JSP

Para crear nuestro primer archivo JSP debemos situarnos sobre la carpeta WebContent y hacer clic derecho.

→New →JSP File

La primera ventana además de pedir el nombre, el cual se debe guardar con extensión .jsp (para este caso index.jsp), pide la ubicación del archivo. (No era necesario ubicarnos en la carpeta WebContent ☺). La siguiente ventana me muestra un conjunto de templates de archivo JSF (Java Server Faces) o JSP (Java Server Pages).



Esta es la estructura básica del nuevo archivo creado.

```
1  <%@ page language="java" contentType="text/html; charset=ISO-8859-1"
2      pageEncoding="ISO-8859-1"%>
3  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://ww
4  <html>
5  <head>
6  <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
7  <title>Insert title here</title>
8  </head>
9  <body>
10
11 </body>
12 </html>
```

Si queremos ver cómo queda la página podemos hacerlo desde nuestro contenedor, para este caso Apache Tomcat. Se debe seleccionar el proyecto y hacer clic derecho.

→Run As →Run On Server o Alt + Shift + X, R

Esto me permite escoger el contenedor de Servlets si tuviera más configurados y quisiera cambiarlos.

Eclipse contiene un navegador interno, el cual muestra el proyecto en la dirección http://localhost:8080/nombreProyecto/

Codificando mi JSP

Vamos a digitar las siguientes sentencias, la cuales incluyen comentarios, declaraciones, expresiones y sentencias.

```
<%-- Nuestra primera pagina --%>

<%! int tabla = 5;
    String nombre = "Carlos"; %>

Primer Mensaje: <%= nombre %>

<% tabla = tabla + 5;
    out.println("El resultado nuevo es " + tabla); %>
```

Finalmente podemos ver nuevamente la página en el navegador. Después de guardar el debe compilar e interpretar para poder retornar nuevamente el HTML generado.

Creando un formulario

Vamos a crear un formulario de contacto con HTML5 y CSS 3. Inicialmente solo construiremos el formulario con HTML5 y luego nos preocuparemos por el CSS. Vamos a crear una nueva pagina formulario.jsp.

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>HTML5 y CSS Formulario de contacto</title>
<link href="css/estilo.css" rel="stylesheet" />
</head>
<body>
<form class="contact_form" action="#" id="contact_form" runat="server">
<div>
    <ul>
    <li> <h2>Contactos</h2> <span class="required_notification">* Datos requeridos</span>
        </li>
    <li> <label for="name">Nombre:</label> <input type="text"
        placeholder="Carlos Angarita" required /></li>
    <li> <label for="email">Email:</label> <input type="email" name="email"
        placeholder="info@dominio.com" required /> <span class="form_hint">Formato correcto: "name@dominio.com"</span></li>
    <li> <label for="website">Sitio web:</label> <input type="url" name="website"
        placeholder="http://dominio.com" required pattern="(http|https)://.*" />
        <span class="form_hint">Formato correcto: "http://dominio.com"</span></li>
    <li> <label for="message">Mensaje:</label> <textarea name="message" cols="40" rows="6" required></textarea>
        </li>
    <li> <button class="submit" type="submit">Enviar mensaje</button>
        </li>
    </ul>
    </div>
</form>
</body>
</html>
```

Igual del aula se puede descargar el archivo, tanto el CSS como el HTML.
Paso de parámetros
Una página permite el envío de parámetros a través de dos métodos, GET y POST, los cuales son definidor en el formulario.
Para este ejercicio vamos a crear una nueva página JSP que se llama recibir.jsp. Ahora vamos a modificar el action del formulario de la página formulario.jsp y vamos a poner recibir.jsp.

En la página JSP, para obtener el parámetro debemos usar la instrucción:

request.getParameter("nombrecampo")

Usa esta sentencia para obtener todos los datos del formulario. Como llegan los datos, intenta enviar por GET y por POST.

Preguntas Adicionales

1. Defina los siguientes contenedores Tomcat, Jetty, JBoss, Resin
2. Que es el ServletContext
3. Cuál es el método estándar para el empaquetado de aplicaciones Web
4. Que es JSF y cuál es la diferencia con JSP
5. Cuál es el puerto por defecto del Apache Tomcat ¿Se puede cambiar?
6. Diferencia en el envio entre GET y POST.

Recomendaciones de Entrega:

- El documento debe comprimirse en un archivo .ZIP o .RAR y se debe subir a la plataforma Uvirtual – Plataforma de apoyo al docente de la UFPS en la fecha máxima establecida.

Nota:

- Trabajos enviados fuera de la fecha límite de entrega no serán calificados.
- Cualquier intento de fraude motiva a la anulación del trabajo.
- Cualquier inquietud no duden en escribir al correo del docente (carlosreneas@ufps.edu.co).