Práctico de Deployment

GeneXus™ 16

Setiembre 2019

CONTENIDO

OBJETIVO

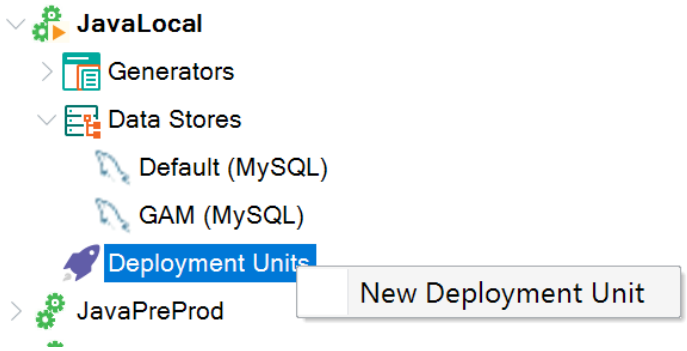
Luego que tenemos configurado y armado todo el proceso de build y test de nuestra aplicación, el siguiente paso es configurar todo lo correspondiente para que se haga el deployment de la misma a el/los servidor/es correspondientes: test, pre-producción, producción.

Para realizar esta tarea vamos a agregar pasos en la configuración de nuestro proyecto Jenkins en el cual venimos trabajando utilizando DevOps.

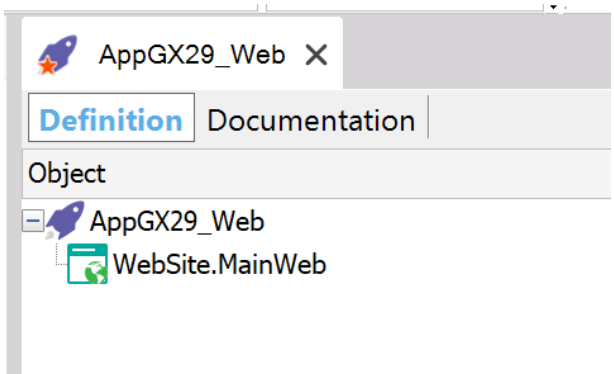
CREACIÓN DE LA DEPLOYMENT UNIT

Previo a configurar el deployment en el proyecto Jenkins vamos a crear una *Deployment Unit*. La Deployment Unit va a ser bien sencilla y simplemente va a incluir el Web Panel llamado *MainWeb*, este Web Panel tiene llamadas a todos los componentes de la aplicación Web, por lo que al hacer deploy del mismo irá todo lo necesario para la ejecución del sitio del GX29.

- Vamos a *Preferences* y en el Environment actual, en *Deployment Units*, hacemos botón derecho -> *New Deployment Unit*.



- Ponemos un nombre a la Deployment Unit, por ejemplo, AppGX29_Web y luego en el diálogo de edición de la misma agregamos el Web Panel *MainWeb*



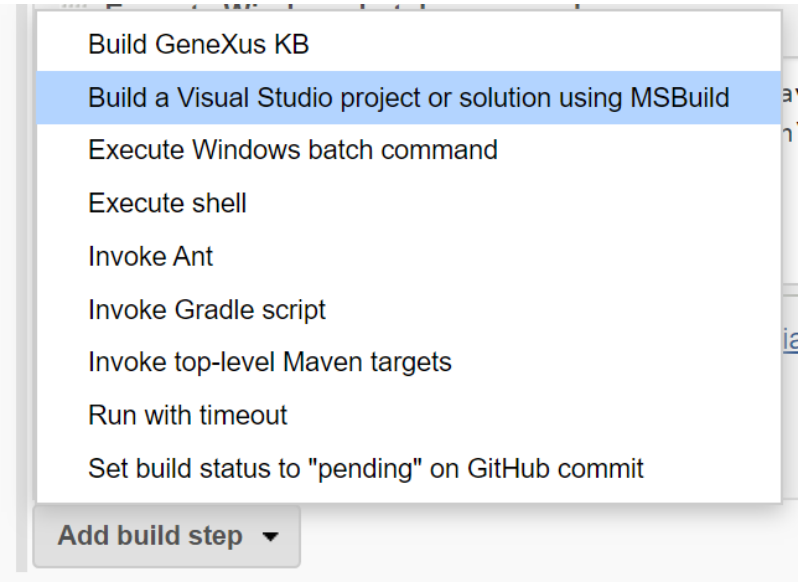
CONFIGURACIÓN EN JENKINS

Ahora sí, vamos a configurar las tareas de deployment en Jenkins. Vamos a crear un paquete war con la aplicación y hacer deploy del mismo en lo que sería el servidor de "producción" (en el caso del práctico, a modo de ejemplo vamos a usar el Tomcat local de la máquina de desarrollo).

Este proceso se realiza en 3 pasos:

1- CREACIÓN DEL ARCHIVO GXDPROJ CON MSBUILD

El primer paso es crear un archivo *gxdproj*, a través de una tarea MsBuild, que contendrá la información necesaria para crear el paquete de la aplicación a deployar. Para eso vamos a ir a nuestro proyecto Jenkins y agregar un nuevo paso en el build con el botón *Add Build Step*, que sea de tipo *Build Visual Studio project or solution using MsBuild*.



Cuando creamos un step de tipo MsBuild debemos indicar 3 cosas:

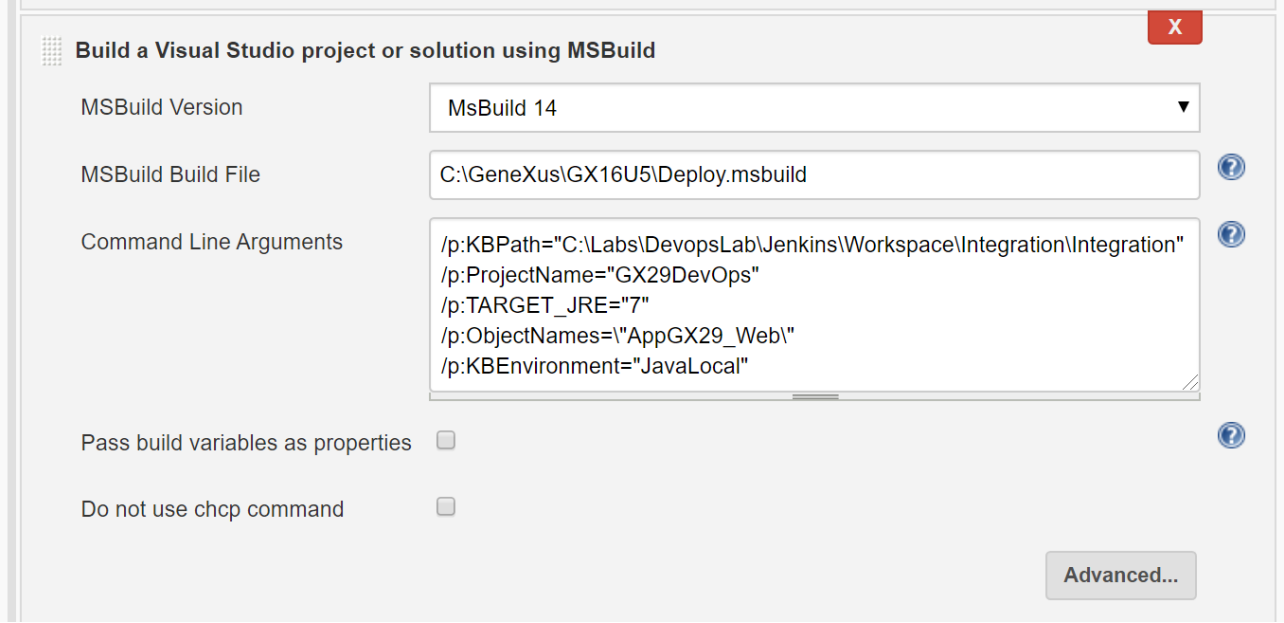
a) Qué MsBuild vamos a usar.
b) Que archivo vamos a ejecutar.
c) Que parámetros se van a enviar en la ejecución.

Entonces para crear el proyecto de deploy vamos a usar la versión de MsBuild que tenemos catalogada en Jenkins, ejecutar el archivo *Deploy.msbuild* que está en la raíz de nuestra instalación de GeneXus y le vamos a pasar los siguientes parámetros:

- */p:KBPath* : Path de la KB

- */p:ProjectName* : Un nombre para la tarea de deploy, podemos elegir cualquiera. Se va a usar en el siguiente paso.
- */p:TARGET_JRE* : Versión de Java que se va a utilizar
- */p:ObjectNames* : Objeto u objetos que se van a deployar. Vamos a utilizar el nombre de la Deployment Unit que creamos al principio.
- */p:KBEnvironment* : Environment de la KB desde donde vamos a deployar

Como ejemplo la configuración en Jenkins quedará de la siguiente manera:

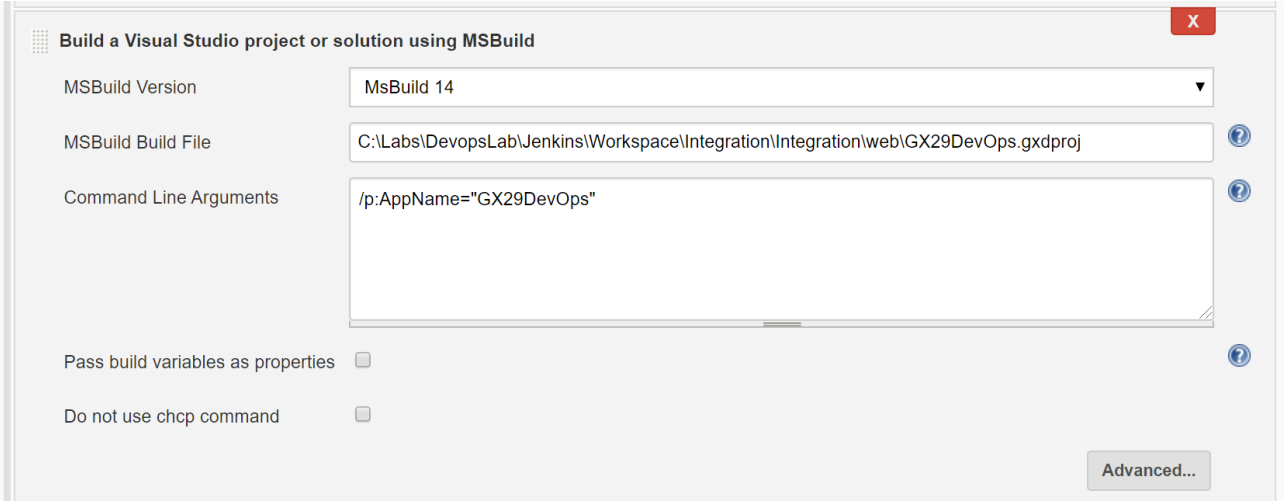


2- CREACIÓN DEL PAQUETE WAR DE LA APLICACIÓN

El archivo gxdproj creado en el paso 1, es en sí un archivo MsBuild, que debemos ejecutarlo para que se genere el paquete war de nuestra aplicación. Así que cómo en el paso 1, debemos agregar un nuevo paso en el proyecto Jenkins de tipo *Build Visual Studio project or solution using MsBuild* y usando la siguiente configuración:

- Usamos el MsBuild catalogado en Jenkins
- El archivo a ejecutar será el creado en el paso 1, que fue generado en el directorio web del environment de la KB seleccionado, con el nombre del proyecto indicado (en el parámetro */p:ProjectName*) y extensión gxdproj
- Se necesita configurar un sólo parámetro */p:AppName* donde indicamos un nombre para la aplicación, que puede ser cualquiera y será usado en el siguiente paso.

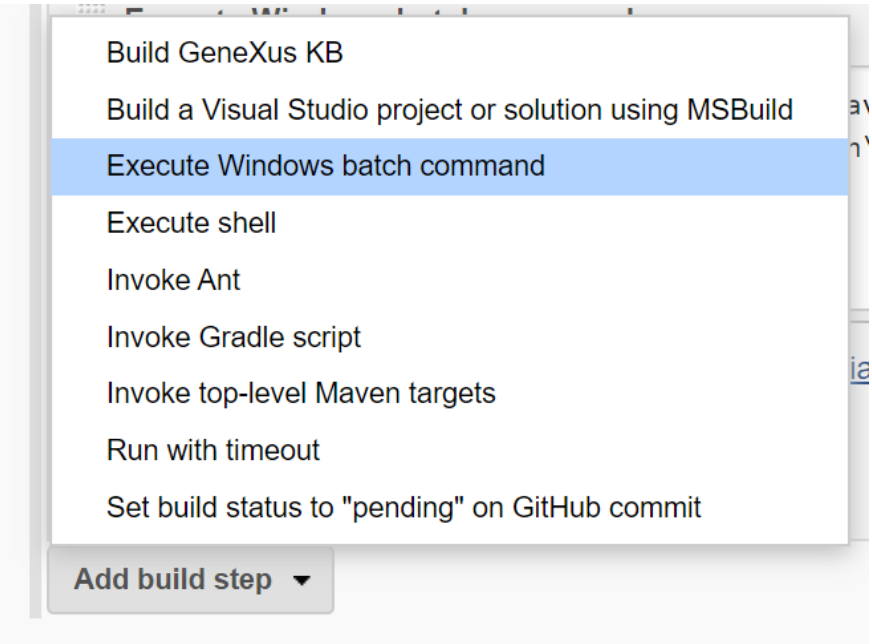
Como ejemplo la configuración en Jenkins quedará de la siguiente manera:



3- COPIA DEL PAQUETE WAR AL TOMCAT

Luego que tenemos el paquete war creado necesitamos llevarlo al servidor correspondiente. Aquí depende de que servidor estamos utilizando, lo que necesitamos ejecutar. GeneXus provee tareas MsBuild para enviar paquetes war a servidores de Amazon, Azure, IBM, SAP, etc. que deberían ser ejecutadas en este momento si fuera el caso.

Como a los efectos del práctico, vamos a usar un Tomcat local, lo que tenemos que hacer es simplemente copiar el archivo war al servidor.



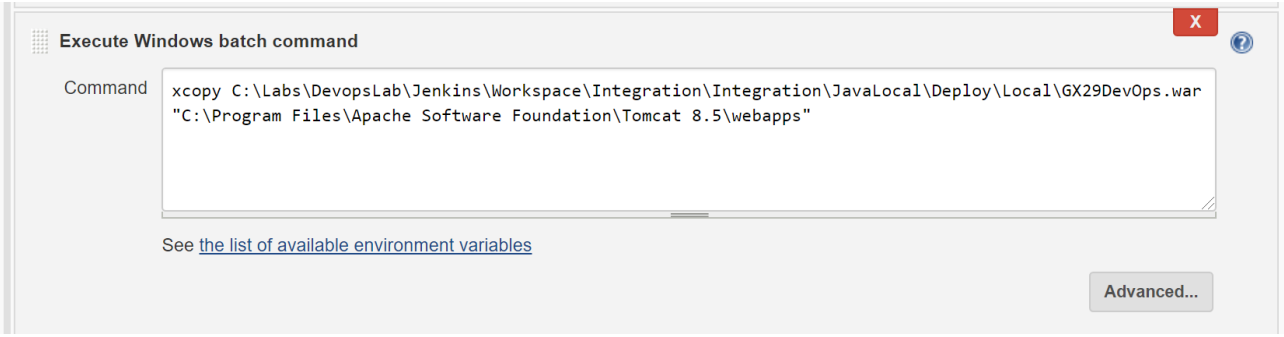
Esto lo configuramos en Jenkins, agregando un nuevo paso de build de tipo Execute Windows batch command en el cual vamos a configurar que se copie el archivo al directorio de las webapps del Tomcat con el siguiente comando:

xcopy <KBPath>\<EnvironmentName>\Deploy\Local\<AppName>.war <TomcatPath>\webapps

Donde:

- *<KBPath>* es el path donde está nuestra KB
- *<EnvironmentName>* es el nombre del environment de la KB donde estamos trabajando. Es el mismo que indicamos en el paso 1 (en el parámetro */p:KBEnvironment*)
- *<AppName>* es el nombre de la aplicación que configuramos en el paso 2 con el parámetro */p:AppName*
- *<TomcatPath>* es el path donde está instalado el Tomcat

Como ejemplo la configuración en Jenkins quedará de la siguiente manera:



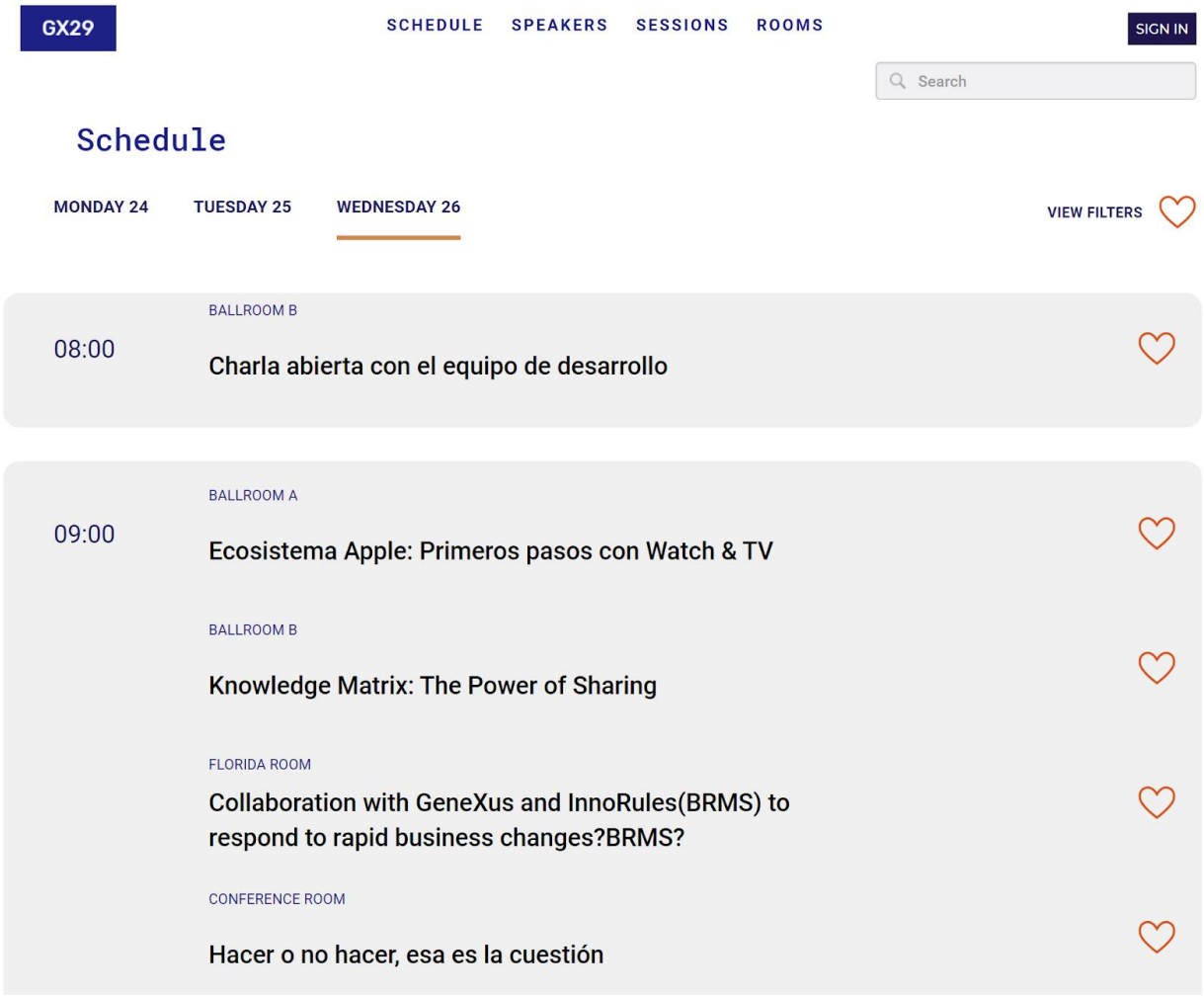
Nota: Si alguno de los path indicados contiene espacios, el valor debe ser encerrado entre comillas. El valor del path de Tomcat en la imagen es de ejemplo, verificar que versión se tiene instalada y utilizar el correcto.

Finalmente vamos a probar que todo quedó bien deployado ejecutando un Web Panel de la aplicación. Ya que usamos nuestro Tomcat local, vamos a ejecutar la siguiente URL y veremos en pantalla el programa del evento.

http://localhost:8080/<AppName>/servlet/com.gx29.website.schedule

Donde:

- *<AppName>* es el nombre de la aplicación que configuramos en el paso 2 con el parámetro */p:AppName*



TEST DE INTEGRACIÓN

Vamos a realizar una modificación en nuestra aplicación y luego ejecutar el proyecto Jenkins para verificar que ahora se ejecuta:

- Build de la aplicación con el cambio
- Test
- Deploy a producción

Vamos a editar el Web Panel ScheduleData, que tiene la información de cada día del programa y vamos a descomentar el código que está en el evento *GridSessions.Load* entre las líneas 56 a 75:

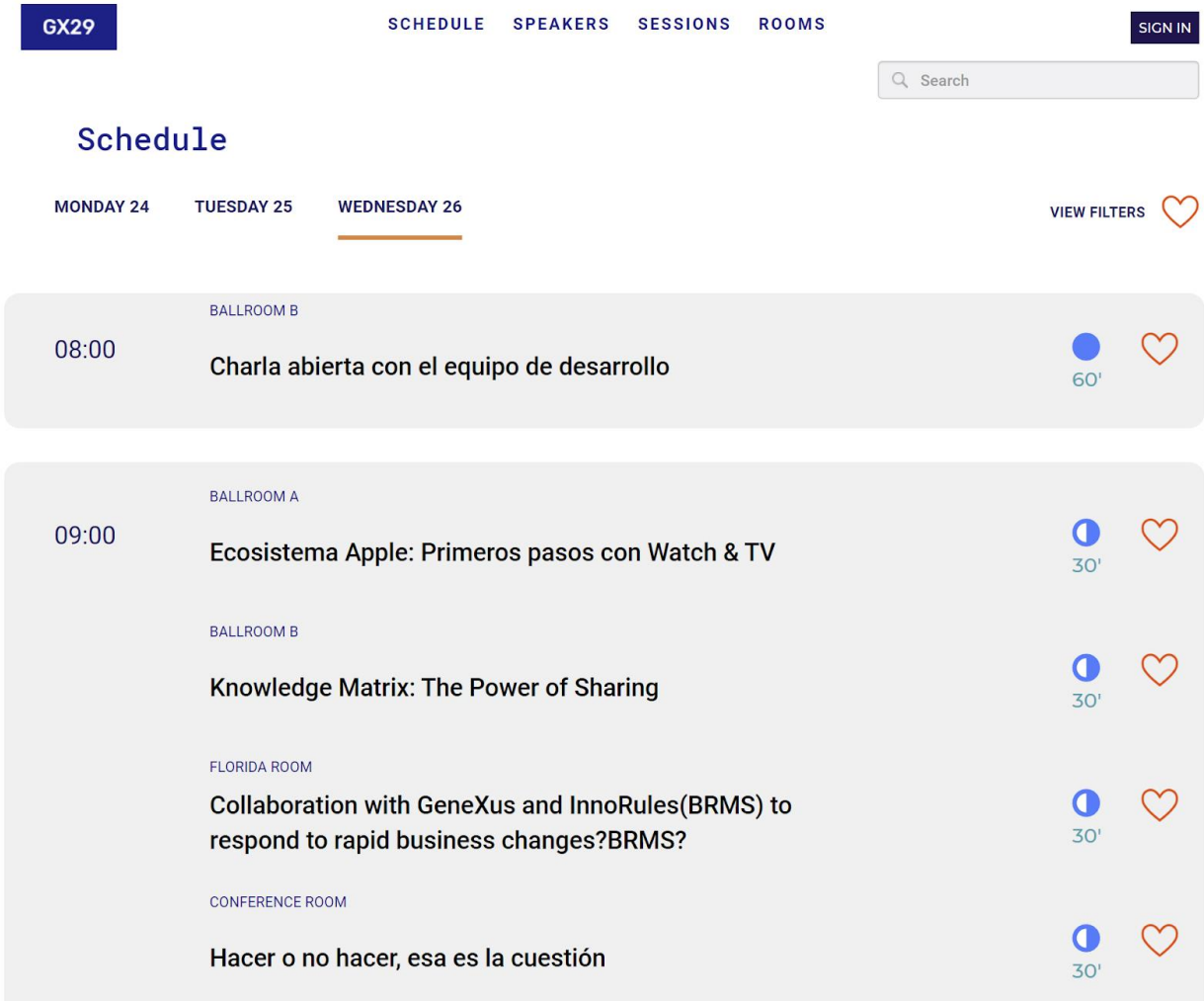
```
TableDuration.Class = ThemeClass:TableTime
Do Case
    Case &Session.SessionDuration = 30
        txtDuration.Caption = !"30'"
        txtDurationExtraSmall.Caption = !"30'"
        ImageDuration.FromImage(Time30)
        ImageDurationExtraSmall.FromImage(Time30)
    Case &Session.SessionDuration = 45
        txtDuration.Caption = !"45'"
        txtDurationExtraSmall.Caption = !"45'"
        ImageDuration.FromImage(Time45)
        ImageDurationExtraSmall.FromImage(Time45)
    Case &Session.SessionDuration = 60
        txtDuration.Caption = !"60'"
        txtDurationExtraSmall.Caption = !"60'"
        ImageDuration.FromImage(Time60)
        ImageDurationExtraSmall.FromImage(Time60)
    Otherwise
        TableDuration.Class = ThemeClass:TableTime + !" " +
ThemeClass:TableVisibilityHidden
EndCase
```

Y comentamos la línea 77

```
// TableDuration.Class = ThemeClass:TableTime + !" " +
ThemeClass:TableVisibilityHidden
```

Con ese código nuevo, lo que pretendemos es que al lado de cada charla salga una imagen y un texto indicando la duración de la misma.

Ejecutamos nuestro proyecto Jenkins y luego de finalizado el mismo, verificamos en la URL del sitio de producción (*http://localhost:8080/<AppName>/servlet/com.gx29.website.schedule*) que el cambio fue armado, testeado y deployado.



GeneXus™
www.genexus.com

MONTEVIDEO - URUGUAY	Av. Italia 6201- Edif. Los Pinos, P1	(598) 2601 2082
CIUDAD DE MÉXICO - MÉXICO	Hegel N° 221, Piso 2, Polanco V Secc.	(52) 55 5255 4733
MIAMI - USA	7300 N Kendall Drive, Suite 470	(1) 201 603 2022
SÃO PAULO - BRASIL	Rua Samuel Morse 120 Conj. 141	(55) 11 4858 0300
TOKYO - JAPAN	2-27-3, Nishi-Gotanda Shinagawa-ku, Tokyo, 141-0031	(81) 3 6303 9381 (81) 3 6303 9980