mat|r

Bienvenidos a la Clase N°05

En unos minutos comenzamos...

Clase 05

Introducción a mat|r script

mat|r project

Ecosistema 360 para la creación de experiencias digitales

2020

Expositor

Agustina Dinamarca

Capacitadora mat|r

Se dedica a representar el producto en actividades que tienen como objetivo educar a usuarios sobre qué es mat|r, para qué sirve y cómo usarlo a través del dictado de webinars y capacitaciones, y el desarrollo de contenido educativo.

DESAFÍO MAT|R DE LA SEMANA (:

¡PON A PRUEBA TUS CONOCIMIENTOS E INGENIO!

DESARROLLA UNA APP QUE SEA UN CONVERSOR DE UNIDADES.

Ayuda:

- Un primer layout en el cual el usuario pueda elegir, presionando un botón, el tipo de conversor. Ej. : metro-pulgadas, libra-kilogramo, etc.
- Un segundo layout en el cual el usuario indique una cantidad a convertir de acuerdo al tipo de conversor elegido en el primer layout y que al presionar un botón se obtenga el resultado de la conversión.

CONTINUEMOS...

OBJETIVOS DE LA CLASE

- Estudiar qué son las reglas en mat|r script, dónde se las define y cómo usarlas en una aplicación.
- Aprender a asociar acciones a los distintos eventos que ocurren en los componentes visuales de un layout o acciones que actúen modificando algún valor de un atributo de un modelo.

HOY APRENDERÁS

- Contextos de reglas y reglas. Definiciones. Cláusulas; Listen y When en reglas.
- Cómo acceder y usar reglas en mat|r script en cualquier lugar del código de una aplicación.
- Cómo vincular reglas a eventos en los componentes visuales de un layout.

REGLAS Y CONTEXTOS DE REGLAS

¿QUÉ ES UNA REGLA Y CÓMO SE DEFINE?

ES UNA FORMA DE AGRUPAR SENTENCIAS DE CÓDIGO A EJECUTARSE COMO REACCIÓN A DIFERENTES EVENTOS EN LA UI.

- Se definen mediante la palabra reservada: `Rule`
- Deben estar declaradas dentro de un contexto que las agrupa, el cual se define mediante la palabra reservada: `RuleContext`

¿QUÉ ES UNA REGLA Y CÓMO SE DEFINE?

ES UNA FORMA DE AGRUPAR SENTENCIAS DE CÓDIGO A EJECUTARSE COMO REACCIÓN A DIFERENTES EVENTOS EN LA UI.

La sintaxis de una regla, dentro de un contexto de reglas, es la siguiente:

```
RuleContext nombreDelContexto {

    Rule nombreDeRegla [cláusula listen] [cláusula when] {

        //cuerpo de la regla
    }
}
```

¿QUÉ ES UNA REGLA Y CÓMO SE DEFINE?

El proyecto de app puede contener tantos RuleContexts como se quiera, y cada uno con tantas Rules como se requiera.

```
RuleContext contextoReglas1 {

    Rule regla11 {

        //cuerpo de la regla
    }

    ...

    Rule regla1N {

        //cuerpo de la regla
    }


}
```

```
RuleContext contextoReglas2 {

    Rule regla21 {

        //cuerpo de la regla
    }

    ...

    Rule regla2N {

        //cuerpo de la regla
    }


}
```

¿QUÉ EVENTOS DISPARAN LA EJECUCIÓN DE UNA REGLA?

1. **Tap en un botón** en la pantalla por interacción del usuario (regla usando `Decision`).
2. **Tap en un componente visual**, al que se le asoció una acción de ejecución de regla desde la “Configuración de Eventos” en “UI Builder”.
3. **Por activación de un listen**: el modelo y/o atributo al que la regla está observando fue modificado.

CLÁUSULA LISTEN DE REGLAS

PERMITE CONFIGURAR UNA REGLA, PARA QUE OBSERVE UNO O MÁS ATRIBUTOS DE UN MODELO, O CADENA DE ATRIBUTOS Y SE EJECUTE SI ALGUNO DE SUS VALORES CAMBIA.

- La sintaxis de una **regla con cláusula listen** es la siguiente:

```
Rule nombreRegla listen(nombreAtributo from modelo as nombreVariable) {

    // cuerpo de la regla
}
```

CLÁUSULA LISTEN DE REGLAS

PERMITE CONFIGURAR UNA REGLA, PARA QUE OBSERVE UNO O MÁS ATRIBUTOS DE UN MODELO, O CADENA DE ATRIBUTOS Y SE EJECUTE SI ALGUNO DE SUS VALORES CAMBIA.

- ***nombreAtributo***: nombre del atributo a observar. El caracter * puede ser utilizado para observar todos los atributos del modelo.
- ***modelo****:* nombre del modelo que contiene el atributo a observar.
- ***nombreVariable***: nombre de una variable local que se usará dentro del cuerpo de la regla para referenciar a la instancia del modelo cuyo atributo fue modificado.

CLÁUSULA WHEN DE REGLAS

PERMITE AGREGAR CONDICIONES PARA LA EJECUCIÓN DE LA REGLA, DE MODO QUE LAS MISMAS SERÁN EVALUADAS Y DEPENDIENDO DEL RESULTADO, SE EJECUTARÁ O NO LA REGLA.

La sintaxis de una **regla con cláusula when** es la siguiente:

```
Rule nombreRegla when(expresion booleana) {

    // cuerpo de la regla
}
```

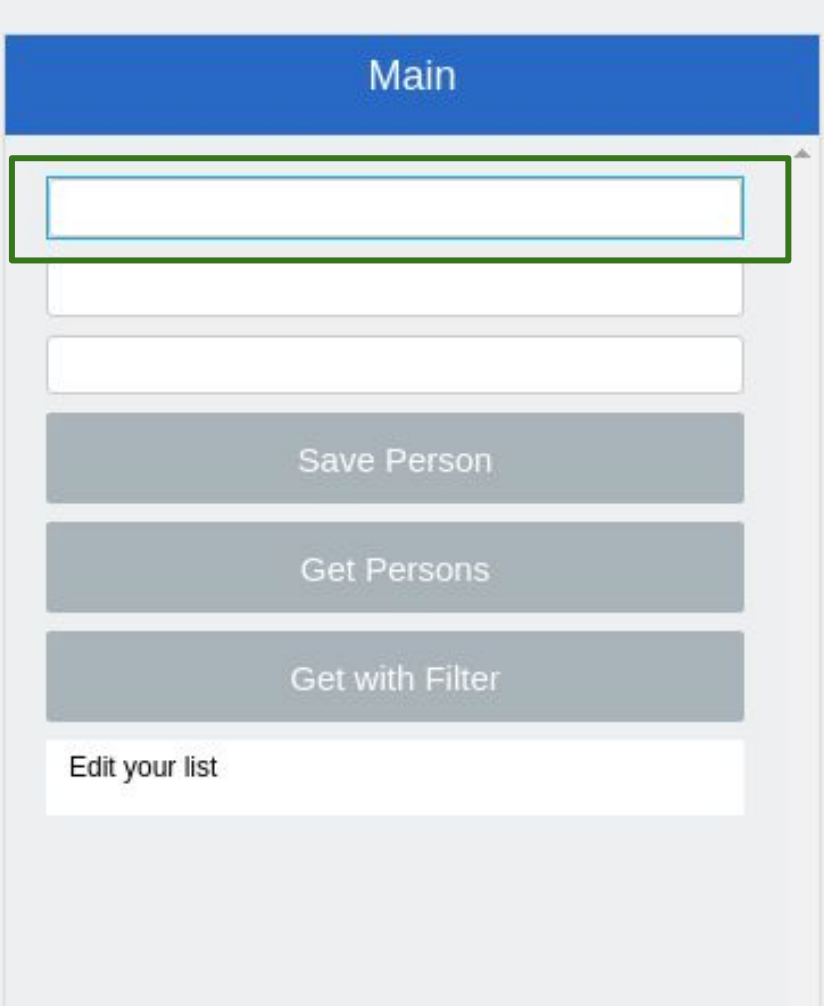
VINCULAR REGLAS A EVENTOS EN LA UI

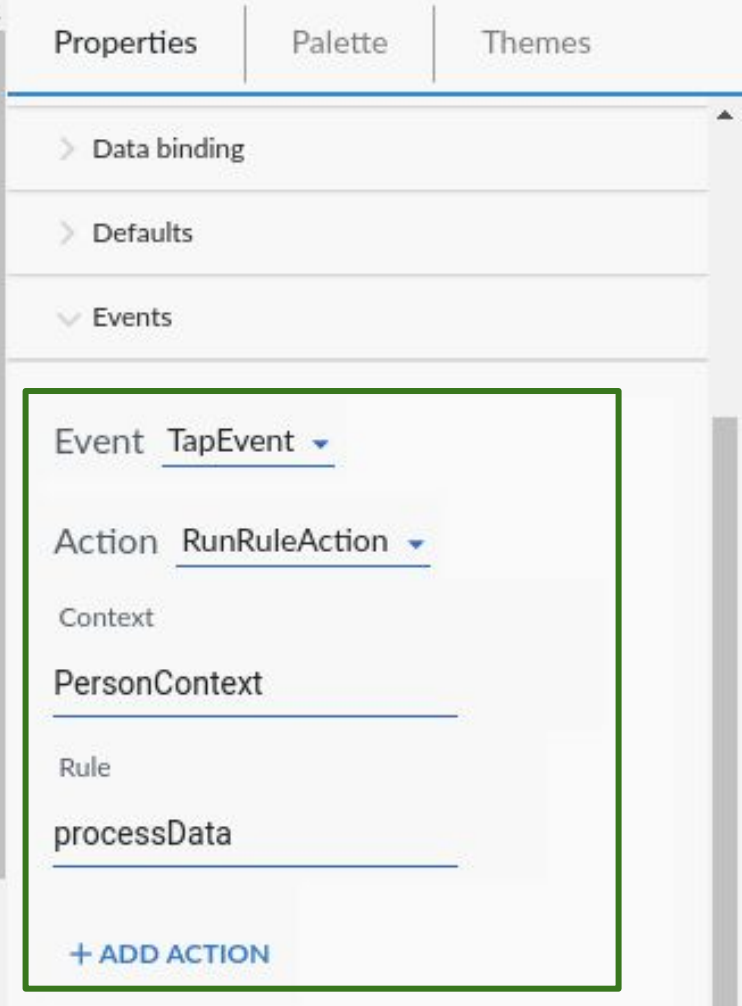
¿CÓMO ASOCIARLOS?

1. Ir al UI Build, al layout de interés, y agrega o selecciona un componente visual.
2. Al hacer click en el componente, ir a “Properties”, “Events”.

Según el componente se tendrán habilitados ciertos eventos, y según estos, el tipo de acción.

3. Seleciona un tipo de evento.
4. Selecciona un tipo de acción.
5. Completa los campos que aparezcan.
6. Click en “+ADD ACTION” y listo (:


Main
Save Person
Get Persons
Get with Filter
Edit your list


Properties
Palette
Themes
Data binding
Defaults
Events
Event TapEvent
Action RunRuleAction
Context
PersonContext
Rule
processData
+ ADD ACTION

TIPOS DE EVENTOS

Tipos de Eventos
TapEvent
LongPressEvent
OnSelectEvent
OnSelectBubbleEvent (sólo en Map View)

- Video y PDF no admiten eventos.

TIPOS DE ACCIONES

Tipos de Acciones sobre los Eventos
ShowAlertAction
NavigateAction
RunRuleAction
InvokeFunctionAction

RESUMEN

RESUMEN

- Se estudió que las reglas se definen en contextos de reglas y agrupan código a ser ejecutado cuando ocurren eventos. Además, pueden usarse cláusulas when y listen.
- Se pueden asociar la acción de reglas a los distintos eventos que ocurren en los componentes visuales de un layout o frente a modificaciones de valores en los atributos de modelos.

EXTRA :D

TIPO DE DATO: DATE

PERMITE LA REPRESENTACIÓN DE UN INSTANTE DE TIEMPO ESPECÍFICO

CONSTRUCTORES	DESCRIPCIÓN ES EL MÁS COMÚN
`Date()`	Crea una instancia configurada con el instante en el que se llamó al constructor.
`Date(timeIntervalSinceNow: Double)`	Crea una instancia configurada con el instante en el que se llamó al constructor sumando la cantidad de segundos indicados en el parámetro 'timeIntervalSinceNow'.

TIPO DE DATO: DATE

PERMITE LA REPRESENTACIÓN DE UN INSTANTE DE TIEMPO ESPECÍFICO

CONSTRUCTORES	DESCRIPCIÓN
`Date (timeIntervalSince1970: Double)`	Crea una instancia configurada con el instante resultante de sumar la cantidad de segundos indicados en el parámetro 'timeIntervalSince1970' al instante '1 de enero de 1970 a las 00:00:00 UTC'.
`Date (year: Integer, month: Integer, day: Integer, hour: Integer, minutes: Integer, seconds: Integer, UTCOffset: Integer)`	Crea una instancia configurada con un instante en formato UTC utilizando los parámetros año, mes, dia, hora, minutos. El parámetro UTCOffset permite sumarle al instante la diferencia en horas respecto a las 0:00hs UTC.

MÉTODOS DE DATE

MÉTODO	DESCRIPCIÓN
`Integer` **`year`**`()`	Retorna el año, considerando calendario gregoriano y timezone UTC.
`Integer` **`month`**`()`	Retorna el número de mes, considerando calendario gregoriano y timezone UTC.
`Integer` **`day`**`()`	Retorna el día, considerando calendario gregoriano y timezone UTC.
`Integer` **`hour`**`()`	Retorna la hora, considerando calendario gregoriano y timezone UTC.

MÉTODOS DE DATE

MÉTODO	DESCRIPCIÓN
`Integer` **`minutes`**`()`	Retorna la cantidad de minutos, considerando calendario gregoriano y timezone UTC.
`Integer` **`seconds`**`()`	Retorna la cantidad de segundos, considerando calendario gregoriano y timezone UTC.
`Integer` **`dayOfWeek`**`()`	Retorna el índice del día de la semana, considerando calendario gregoriano y timezone UTC. Retorna 0 para el día Lunes, 1 para el Martes, … 6 para el Domingo.

MÉTODOS DE DATE

MÉTODO	DESCRIPCIÓN
`Double` **`timeIntervalSinceNow`**`()`	Retorna el intervalo de segundos entre el instante en que fue construida la instancia y el instante de la llamada al método.
`Double` **`timeIntervalSince1970`**`()`	Retorna el intervalo de segundos entre el instante en que fue construida la instancia y el instante '1 de enero de 1970 a las 00:00:00 UTC'.

MÉTODOS DE DATE

MÉTODO	DESCRIPCIÓN
`Double` **`timeIntervalSinceDate`**`(Date fecha)`	Retorna el intervalo de segundos entre el instante en que fue construida la instancia y el instante de la instancia del parámetro.
`Date` **`dateByAddingTimeInterval`**`(Double segundos)`	Retorna una nueva instancia Date resultante de sumar el instante en que fue construida la instancia y los segundos del parámetro.
`String` **`toString`**`()`	Retorna la representación del valor de Date asignado como String.

TIPO DE DATO: DATE FORMATTER

PERMITE EL MANEJO DE LAS DISTINTAS REPRESENTACIONES Y FORMATOS DE FECHAS. CONFIGURA UNA FORMA Y ESTILO DE CÓMO SE VISUALIZARÁ UNA FECHA.

- El formato se define mediante un parametro String que adopta el estándar unicode de fechas.

- Se soportan la mayor parte de las configuraciones a excepción de formatos de las siguientes combinaciones de localización: MMM, MMMM, MMMMM, zzzz y los caracteres: Y, D, q, Q, W, e, E, A, k, K, j.

TIPO DE DATO: DATE FORMATTER

CONSTRUCTORES	DESCRIPCIÓN
DateFormatter(format: String, utcTimeZoneOffset: Integer)	Inicializa el DateFormatter con el formato de fecha del argumento y un desplazamiento en horas igual al argumento utcTimeZoneOffset con respecto a la zona horaria UTC.
DateFormatter(format: String)	Inicializa el DateFormatter con el formato de fecha del argumento y desplazamiento 0 (zona horaria UTC).
DateFormatter()	Inicializa el DateFormatter con el formato “yyyy-MM-dd’T’HH:mm:ss Z” y desplazamiento 0 (zona horaria UTC).

MÉTODOS DE DATE FORMATTER

MÉTODO	DESCRIPCIÓN
`Date` **`dateFromString`**`(String fecha)`	Parsea y convierte el argumento fecha, retornando una nueva instancia Date utilizando el formato con el cual fue configurada la instancia DateFormatter.
`String` **`stringFromDate`**`(Date fecha)`	Genera la representación en String de la fecha argumento, en base al formato con el cual fue configurada la instancia DateFormatter.

¿CONSULTAS O DUDAS?

COMUNÍCATE NOSOTROS

support@matrproject.com

http://forum.matrproject.com

mat|r

matrproject.com

Contacto

Soledad Peñaloza
Community Manager
soledad.p@matrproject.com

Sergio Farias
Product Development
sergio.f@matrproject.com

Agustina Dinamarca
Capacitadora mat|r
agustina.d@matrproject.com

mat|r

¡Muchas Gracias!

mat|r project

Ecosistema 360 para la creación de experiencias digitales