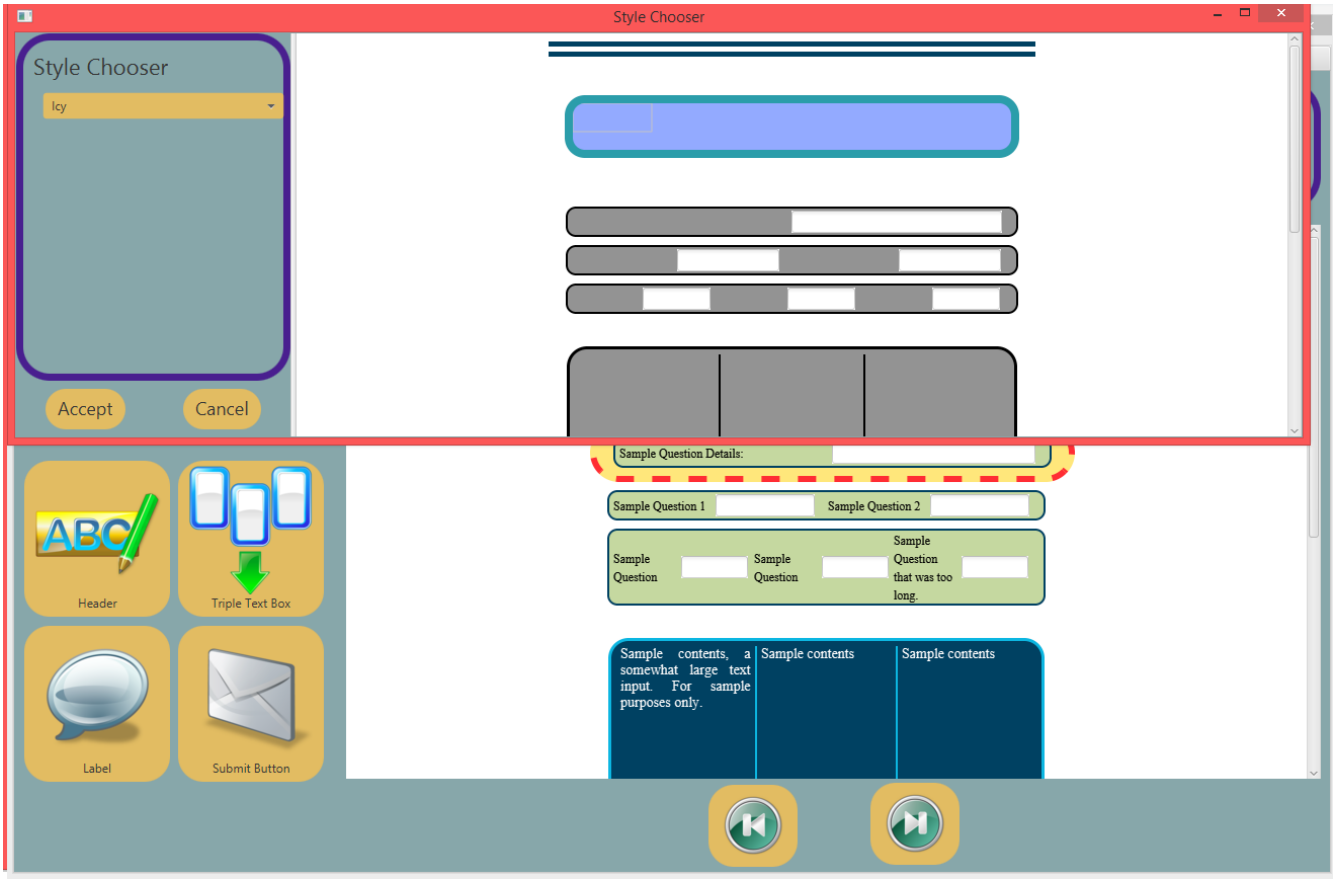
Manual del Desarrollador:



Descripción de las Clases del Programa Java:

parkhon package: En este paquete se encuentran los controladores de JavaFX y algunas clases de lógica básica del programa. Adicionalmente este paquete es el que contiene la clase main que comienza la ejecución del programa.

Classes:

ElementStyleTuple: Esta clase contiene una tupla de un FormSymbol y de un StyleSymbol. Esencialmente así agrupa el programa el elemento HTML con su estilo CSS. La lógica central del programa es un ArrayList de objetos de esta clase que rastrea la posición de un elemento HTML junto con el estilo que le fue aplicado.

ErrorNotifier: Esta clase es prácticamente de naturaleza puramente estática, sus métodos y variables estáticas sirven para poder presentarle errores al usuario final. Cuando algún error es ocasionado, se le notifica a esta clase de manera publica y estática, y la clase al ser invocada por el controlador central (FormatCreatorController) le avisa a este de todos los errores que fueron provocados durante la ejecución.

FormatCreatorController: Esta clase es el controlador de la ventana principal del programa. Todas las funcionalidades del programa que existen son invocadas según el paradigma orientado a eventos a través de este controlador. En esencia es un controlador de JavaFX. Todas las ventanas que pueden ser creadas surgen desde aquí. Todas las operaciones del usuario de la ventana principal están contenidos en este controlador. Se podría decir que es la raíz del programa desde la perspectiva de la interfaz gráfica.

Main: Esta clase contiene la función main del programa java. Como operación previa al lanzamiento carga todos los estilos y luego cede el control de la aplicación al FormatCreatorController atravez de la interfaz gráfica.

StyleChooserController: Esta clase es el controlador de la ventana de elección de estilo. Su componente mas complejo es el visor de la parte izquierdo de la ventana, y para generarlo escribe sus propios archivos, siendo estos preview.html y sample.css. Para generar el código de estos dos archivos, le pide al FormProcessor el código de cada elemento HTML y al StyleSymbol electo por la interfaz gráfica. su código de CSS para cada elemento de la forma.

UserErrorController: Esta clase es el controlador de la ventana de muestra de errores. Su única función es poder recibir el código String de error. Es creada par el FormatCreatorController con información adquirida del ErrorNotifier.

parkhon.data_structures package: Este paquete es un contenedor para clases cuya principal misión es representar datos de manera interna. Fue creado cuando empezaron a haber demasiadas clases de esta naturaleza, por la que esta algo incompleto en cuanto a que varias clases de esta naturaleza se encuentran en otros paquetes.

Classes:

SymbolWord: SymbolWord es una clase que representa un fragmento de código HTML con un “input” o entrada que puede dar el usuario de la aplicación. Básicamente cualquier elemento que sea personalizable por la interfaz gráfica. esta compuesto de varios symbol words: uno por parte personalizable, y una parte final para el código de cerrada.

CustomizablePart: Esta clase esta contenida dentro de SymbolWord puramente, y básicamente solo es una estructura de datos para simplificar la estructura de SymbolWord. Contiene el input del usuario y también la explicación que proporcionara el programa al usuario para que este personalize el elemento en la ventana de personalizacion.

FormOutputManager: Esta clase sirve para escribir y leer output de manera dinámica. Gobierna la carpeta “tmp”, que contiene todos los archivos generados durante la ejecución del programa. Estos archivos sirven a los visores dentro del programa, pero también el archivo final generado esta dentro de esta carpeta.

parkhon.logic package: Este paquete contiene las clases que dan la lógica de negocios al programa. Desde rastrear la posición de todos los elementos de una forma en la cual se esta trabajando, hasta la lectura de formas ya creadas y de archivos CSS de estilo. Básicamente todo el comportamiento interno del programa esta contenido dentro de este paquete.

Classes:

FormatCreatorLogic: Este es el sistema nervioso central de la aplicación. Todas las modificaciones que se pueden hacer sobre una forma están implementadas en esta clase:

- Posición
- Estilo
- Creación
- Eliminación

La clase es de naturaleza estática, y no se debe instanciar yo que todas sus funcionalidades son estáticas.

Adicionalmente esta clase es la encargada de la generación de los documentos HTML y CSS del visor “preview” pero también de la versión final del programa.

FormProcessor: Esta clase es encargada de lectura y cargado de formas previamente creadas (HTML). Pero también es responsable de la creación de elementos nuevos (HTML). El código de cada elemento esta especificado en esta clase atravez de variables String.

FormSymbol: Un elemento de una forma (encabezado, etiqueta, pregunta, botón de submision, etc.) se maneja de manera interna en el programa como un FormSymbol. Cuando se crea un elemento de Forma se crea un objeto de esta clase, igualmente cuando se carga una forma desde disco, se crean los objetos de esta clase para representar al archivo cargado. Todo el trabajo que se hace con personalizar los elementos de una forma son hechos sobre objetos de esta clase.

Finalmente, esta clase también es responsable de generar el código HTML final de su propio elemento, considerando tanto el estilo como el texto personalizado que le dio el usuario de la aplicación.

StyleReader: Esta clase se encarga de encontrar y cargar archivos de CSS dentro de la carpeta res/styles y cargarlos. Si se logra cargar un archivo CSS se convierte en un StyleSymbol.

StyleSymbol: Esta clase representa un archivo CSS en la lógica interna del programa, atravez de el objeto es posible solicitar el código CSS de cualquier elemento HTML que haya sido registrado como una categoría. las categorías se crean en el archivo CSS atravez de los meta-datos que están contenidos como comentarios de CSS.

Nota al desarrollador: Si se agrega un nuevo elemento de HTML completamente nuevo al programa, este debe tener código de estilo en el estilo CSS default “res/styles/fe_style_default.css”. Cambien hay que agregar esto a “bin/res/styles/fe_style_default.css”

Creación de Estilos:

Los estilos pueden ser utilizados para personalizar la apariencia de los elementos de una forma. El programa carga de manera automática todos les estilos que se encuentren en la carpeta res/styles. Es importante notar que esta carpeta existe tanto dentro de /bin como en el root del programa /. Por lo que es recomendable agregar los nuevos estilos en ambos directorios.

Encabezado del estile nuevo:

Todos los estilos creados necesitan un nombre para que el sistema los pueda registrar. Esto se hace al poner arriba del código CSS:

/* $FormatEditorStyle$<NOMBRE DEL ESTILO>$End */

Donde se remplaza <NOMBRE DEL ESTILO> por el nombre del estilo real, por ejemplo:

/* $FormatEditorStyle$Default$End */

Esto crearía el estilo con el nombre Default (las mayúsculas si importan, default no es igual a Default para el sistema).

Categoría nueva de estilo:

Cada elemento de HTML que se puede agregar a una forma necesita código CSS en un estilo. Para agregarlo se debe poner en el CSS meta-datos como este:

/* $FormatEditorMetadata$Header$End */

<CÓDIGO DE CSS PARA EL HEADER>

/* $FormatEditorEndTag */

Y así para cada elemento HTML diferente. De manera que para dar servicio de Styling a un elemento de tipo OpenQuestion por ejemplo, se haría código de esta manera:

```
/* $FormatEditorMetadata$OpenQuestion$End */
.buap_open_question.styled_Default {
                                border: 2px solid #003b5c;
                                border-radius: 10px;
                                background-color: #c2d69b;
                                width: 90%;
                                overflow: auto;
                                margin: 8px auto;
                                padding: 0px 2% 0px 0px;
}

.buap_open_question.styled_Default span{
                                margin: 0px;
                                margin-left:1%;
                                display: inline-block;
                                vertical-align:middle;
}

.buap_open_question.styled_Default p{
                                margin: auto 0px auto 0px;
                                display:inline;
                                font-size: 12px;
}
.buap_open_question.styled_Default input {
                                width: 95%;
}

.buap_open_question.styled_Default .last_element{
                                text-align: right;
                                margin-right: 5px;
```

```
}

.buap_one_liner_open_question.styled_Default span{
                                width: 49%;
}

.buap_two_liner_open_question.styled_Default span{
                                width: 23%;
}


.buap_two_liner_open_question.styled_Default .subject_end{
                                margin-right: 2.5%;
}

.buap_three_liner_open_question.styled_Default span{
                                width: 15%;
}

.buap_three_liner_open_question.styled_Default input{

}

.buap_three_liner_open_question.styled_Default .subject_end{
                                margin-right: 1%;
}
/* $FormatEditorEndTag */
```

El /* $FormatEditorMetadata$OpenQuestion$End */ notifica al lector que ahi comienza el código de estilo para un elemento OpenQuestion <O cualquier otro tipo especificado, solo hay que remplazarlo en el texto OpenQuestion>,

El /* $FormatEditorEndTag */ notifica al lector automático de CSS que un elemento debe de ser completamente leído hasta ahi, y le alista para buscar otro elemento para saber como estilizar.

Recomendación:

Para crear nuevos estilos es recomendable copiar el estilo Default y modificar la copia con las instrucciones nuevas de estilo. Es importante recordar dar un nombre nuevo a la copia en este caso.

Dependencias y Software de Desarrollo

Java 9: Este software fue hecho utilizando la versión 9 de java, que era la ultima versión disponible al tiempo de desarrollar el software. Para ejecutar el software es imperativo tener el runtime environment de java 9

JavaFX: La interfaz gráfica. y la programación orientada a eventos fue desarrollada atravez de JavaFX. Para compilar el programa es necesario tener JavaFX instalado.

SceneBuilder: Los documentos FXML de la interfaz gráfica. fueron creados con esta herramienta. Es opcional, ya que en teoría es posible editar los FXML sin utilizar ningún software adicional.

Eclipse: El provecto fue desarrollado utilizando el IDE de Eclipse, en el directorio del provecto se encuentran los meta-datos de eclipse para importar el provecto. Por lo que se recomienda utilizar eclipse de la misma manera para mas fácilmente trabajar en el proyecto.

Nota: Al compilar el programa en eclipse, es importante copiar la carpeta ./res y ponerla en ./bin/ de tal manera que exista ./bin/res/styles. Por defecto esta carpeta no es importada dentro de bin en la configuración del provecto.

launcher.bat: El launcher del programa es un script bat muy simple que solo manda la instrucción:

“java parkhon.Main” desde la carpeta bin. Si se desea correr la aplicación desde linea de comando solo es necesario hacer esta instrucción por la linea de comande en bin.