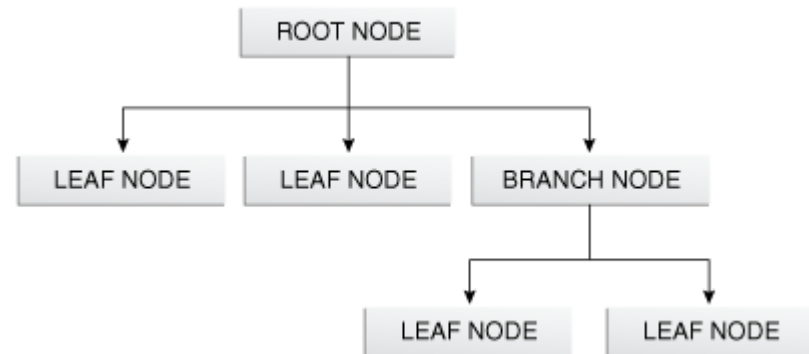


Key concepts in JavaFX

- ▶ **Stage:** where the application will be displayed (e.g., a Windows' window)
- ▶ **Scene:** one container of Nodes that compose one “page” of your application
- ▶ **Node:** an element in the Scene, with a visual appearance and an interactive behavior. Nodes may be hierarchically nested



Essential Reference

▶ JavaFX JavaDoc API

▶ <http://docs.oracle.com/javase/8/javafx/api/>

▶ JavaFX Class Diagrams

▶ <http://www.falkhausen.de/JavaFX>

The screenshot shows the JavaFX 8 JavaDoc API overview page. The left sidebar lists all classes and packages. The main content area displays a table of packages with their descriptions.

Package	Description
<code>javafx.animation</code>	Provides the set of classes for ease of use transition based animations.
<code>javafx.application</code>	Provides the application life-cycle classes.
<code>javafx.beans</code>	The package <code>javafx.beans</code> contains the interfaces that define the most generic form of observability.
<code>javafx.beans.binding</code>	Characteristics of Bindings
<code>javafx.beans.property</code>	The package <code>javafx.beans.property</code> defines read-only properties and writable properties, plus a number of implementations.
<code>javafx.beans.property.adapter</code>	
<code>javafx.beans.value</code>	The package <code>javafx.beans.value</code> contains the two fundamental interfaces <code>ObservableValue</code> and <code>WritableValue</code> and all of its sub-interfaces.
<code>javafx.collections</code>	Contains the essential JavaFX collections and collection utilities
<code>javafx.collections.transformation</code>	
<code>javafx.concurrent</code>	Provides the set of classes for <code>javafx.task</code> .
<code>javafx.css</code>	Provides API for making properties styleable via CSS and for supporting pseudo-class state.
<code>javafx.embed.swing</code>	Provides the set of classes to use JavaFX inside Swing applications.
<code>javafx.embed.swt</code>	Provides the set of classes to use JavaFX inside SWT applications.
<code>javafx.event</code>	Provides basic framework for FX events, their delivery and handling.
<code>javafx.fxml</code>	Contains classes for loading an object hierarchy from markup.
<code>javafx.geometry</code>	Provides the set of 2D classes for defining and performing operations on objects related to two-dimensional geometry.
<code>javafx.print</code>	Provides the public classes for the JavaFX Printing API
<code>javafx.scene</code>	Provides the core set of base classes for the JavaFX Scene Graph API.
<code>javafx.scene.canvas</code>	Provides the set of classes for canvas, an immediate mode style of rendering API.
<code>javafx.scene.chart</code>	The JavaFX User Interface provides a set of chart components that are a very convenient way for data visualization.
<code>javafx.scene.control</code>	The JavaFX User Interface Controls (UI Controls or just Controls) are specialized Nodes in the JavaFX Scenegraph especially suited for reuse in many different application contexts.
<code>javafx.scene.control.cell</code>	The <code>javafx.scene.control.cell</code> package is where all cell-related classes are located, other than the core

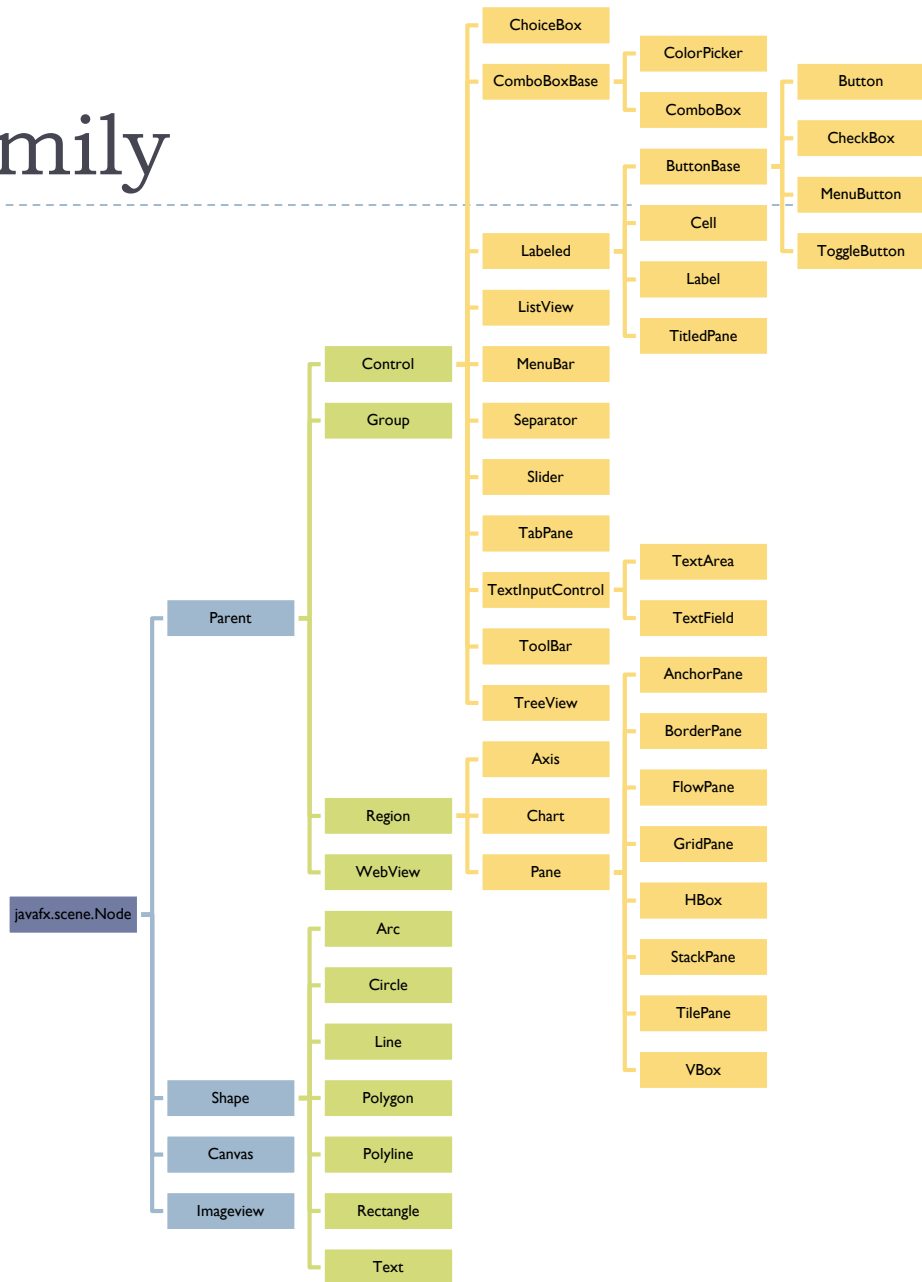
The screenshot shows the scene.control website, which provides class diagrams for JavaFX classes. The site has a navigation menu on the left and a grid of class diagrams on the right.

- Home
- About
- JavaFX
 - animation
 - application
 - beans
 - collections
 - concurrent
 - css
 - embed
 - event
 - fxml
 - geometry
 - print
 - scene
 - scene.chart
 - scene.control
 - cell
 - Control Hierarchy
 - Choice, Combo
 - Container
 - Dialog
 - FocusModel
 - Labeled
 - ListView
 - Menu
 - Progress, Slider, Separator
 - Scrolling
 - SelectionModel
 - Spinner
 - TableColumn
 - TableView
 - TextInputControl
 - Tree
 - TreeTable

The grid of class diagrams includes:

- cell
- Control Hierarchy
- Choice, Combo
- Container
- Dialog
- FocusModel
- Labeled
- ListView
- Menu
- Progress, Slider, Separator
- Scrolling
- SelectionModel

Nodes family



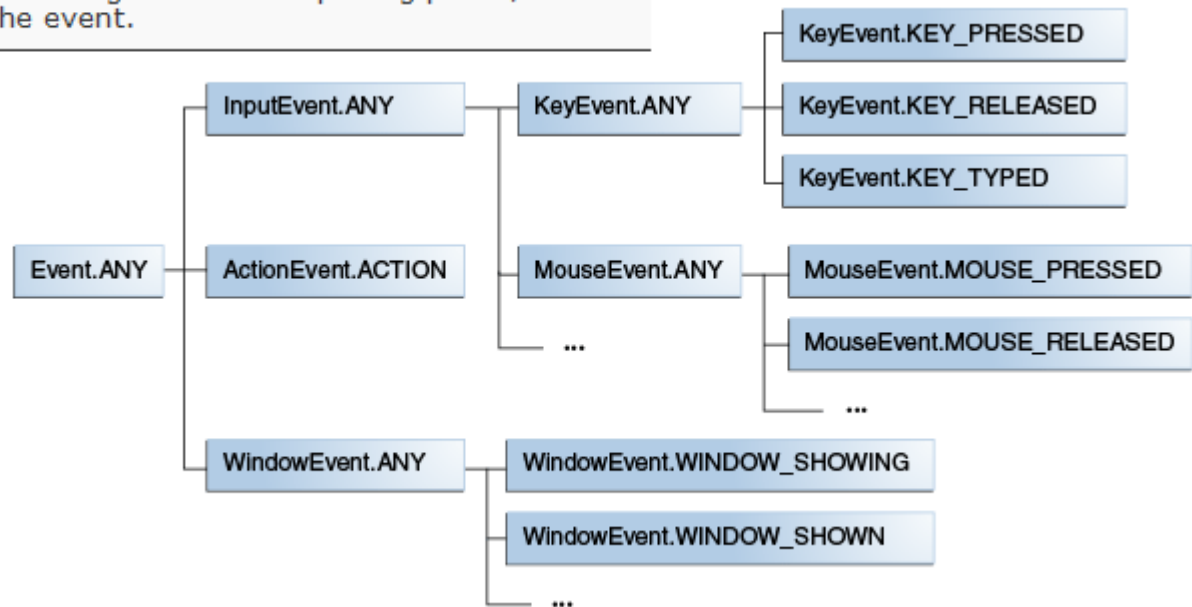
Focus on
Panels
and
Controls

Key concepts in JavaFX

- ▶ **Property:** attributes of the Nodes, may specify content, size, color, ... Can be read and written by the application
- ▶ **Event:** every user action on one element of the GUI generates a different *event*. Events can be captured and *handled* by our code
- ▶ **Controller:** the Java class that contains
 - ▶ References to interesting Nodes
 - ▶ Event Handlers

What is an event?

Property	Description
Event type	Type of event that occurred.
Source	Origin of the event, with respect to the location of the event in the event dispatch chain. The source changes as the event is passed along the chain.
Target	Node on which the action occurred and the end node in the event dispatch chain. The target does not change, however if an event filter consumes the event during the event capturing phase, the target will not receive the event.



Empty JavaFX window

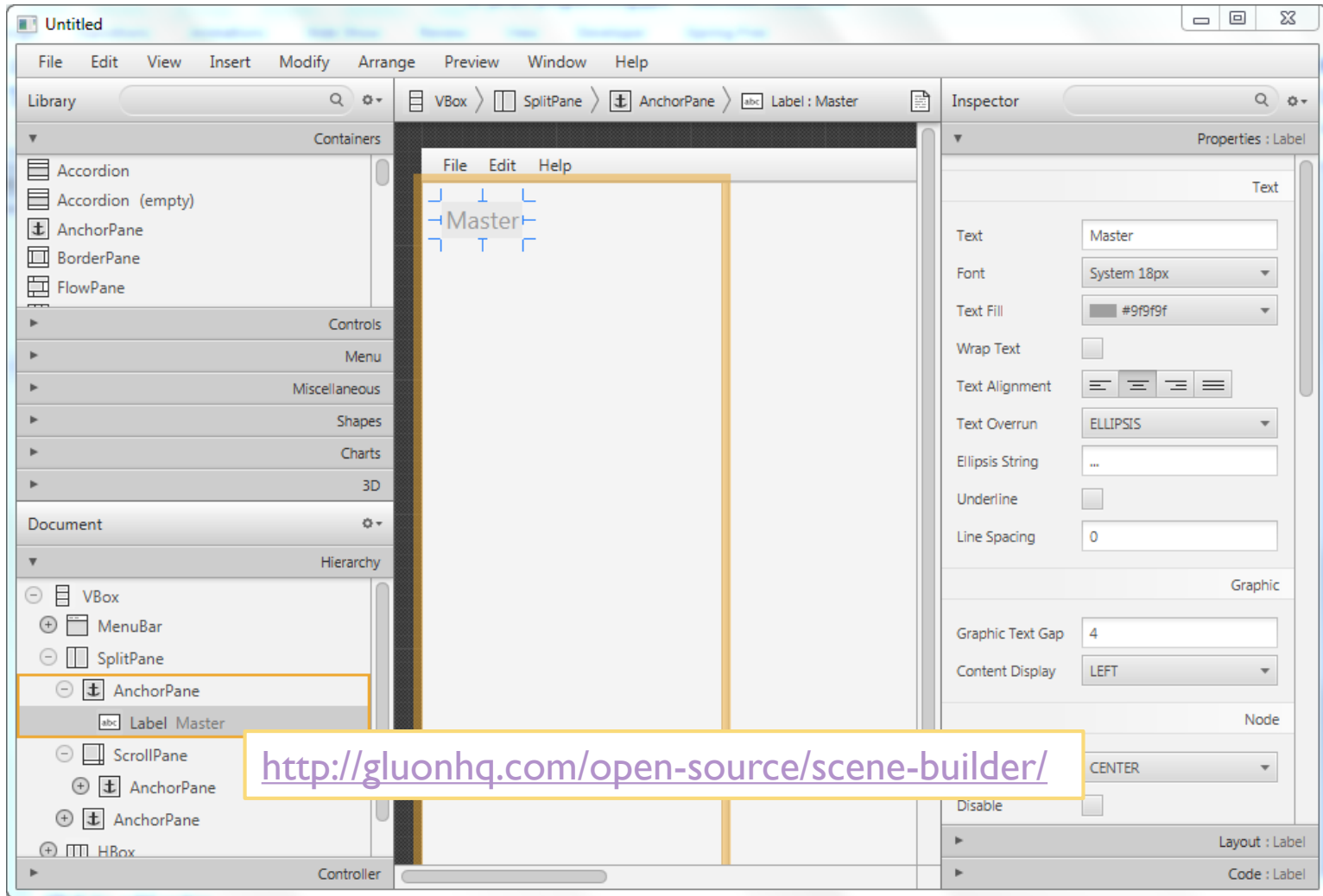
```
public class Main extends Application {  
  
    @Override  
    public void start(Stage stage) {  
        Group root = new Group(); // the root is Group or Pane  
        Scene scene = new Scene(root);  
        stage.setTitle("JavaFX Demo");  
        stage.setScene(scene);  
        stage.show();  
    }  
  
    public static void main(String[] args) {  
        launch(args);  
    }  
}
```

General rules

- ▶ A JavaFX application extends `javafx.application.Application`
- ▶ The `main()` method should call `Application.launch()`
- ▶ The `start()` method is the main entry point for all JavaFX applications
 - ▶ Called with a `Stage` connected to the Operating System's window
- ▶ The content of the scene is represented as a hierarchical scene graph of `Nodes`
 - ▶ `Stage` is the top-level JavaFX container
 - ▶ `Scene` is the container for all content



JavaFX Scene Builder 8.3



Building a scene from FXML

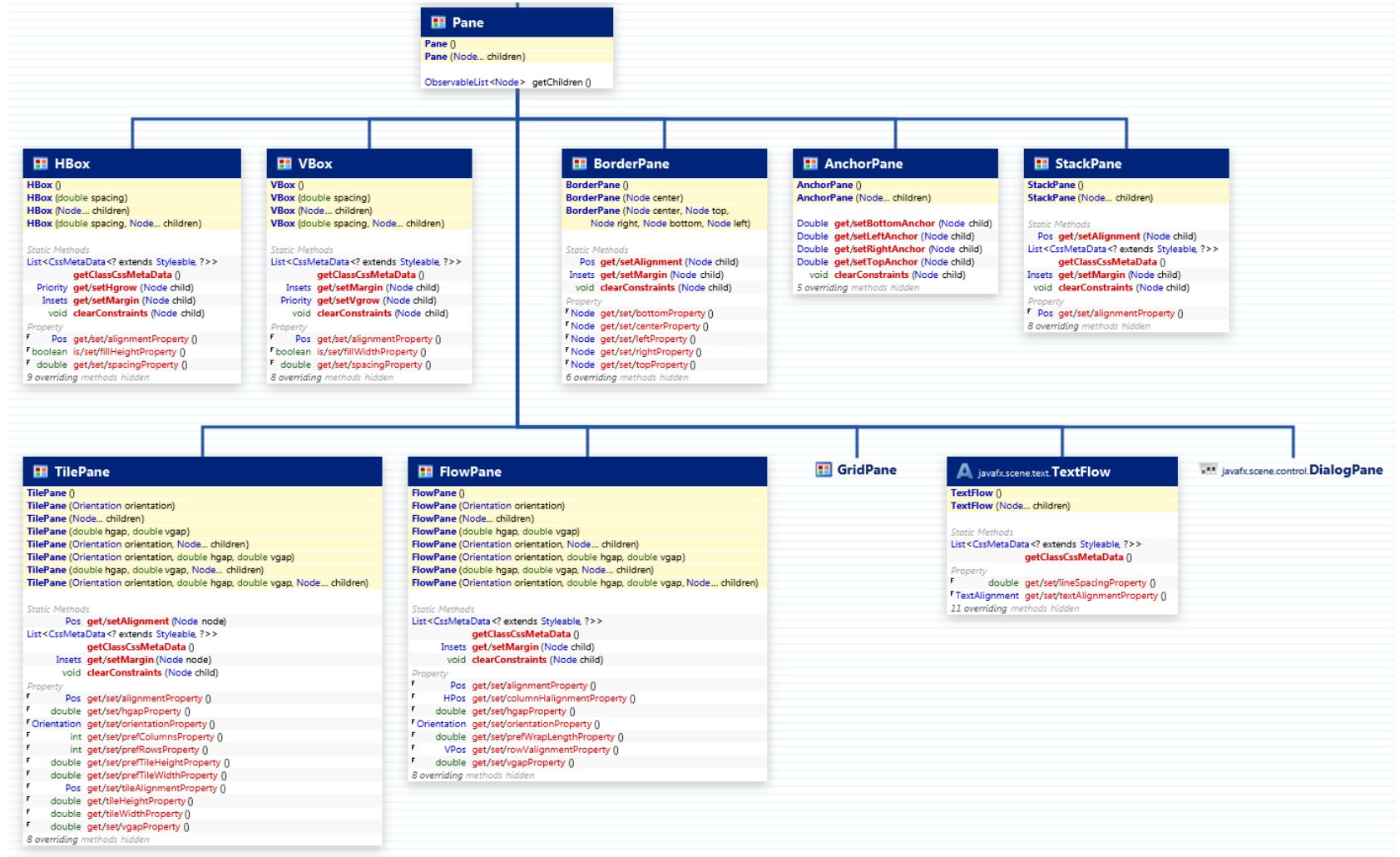
```
public void start(Stage stage) throws Exception {
    Parent root = FXMLLoader.Load(
        getClass().getResource("circle.fxml"));

    stage.setTitle("Circle Demo");
    stage.setScene(new Scene(root));
    stage.show();
}
```

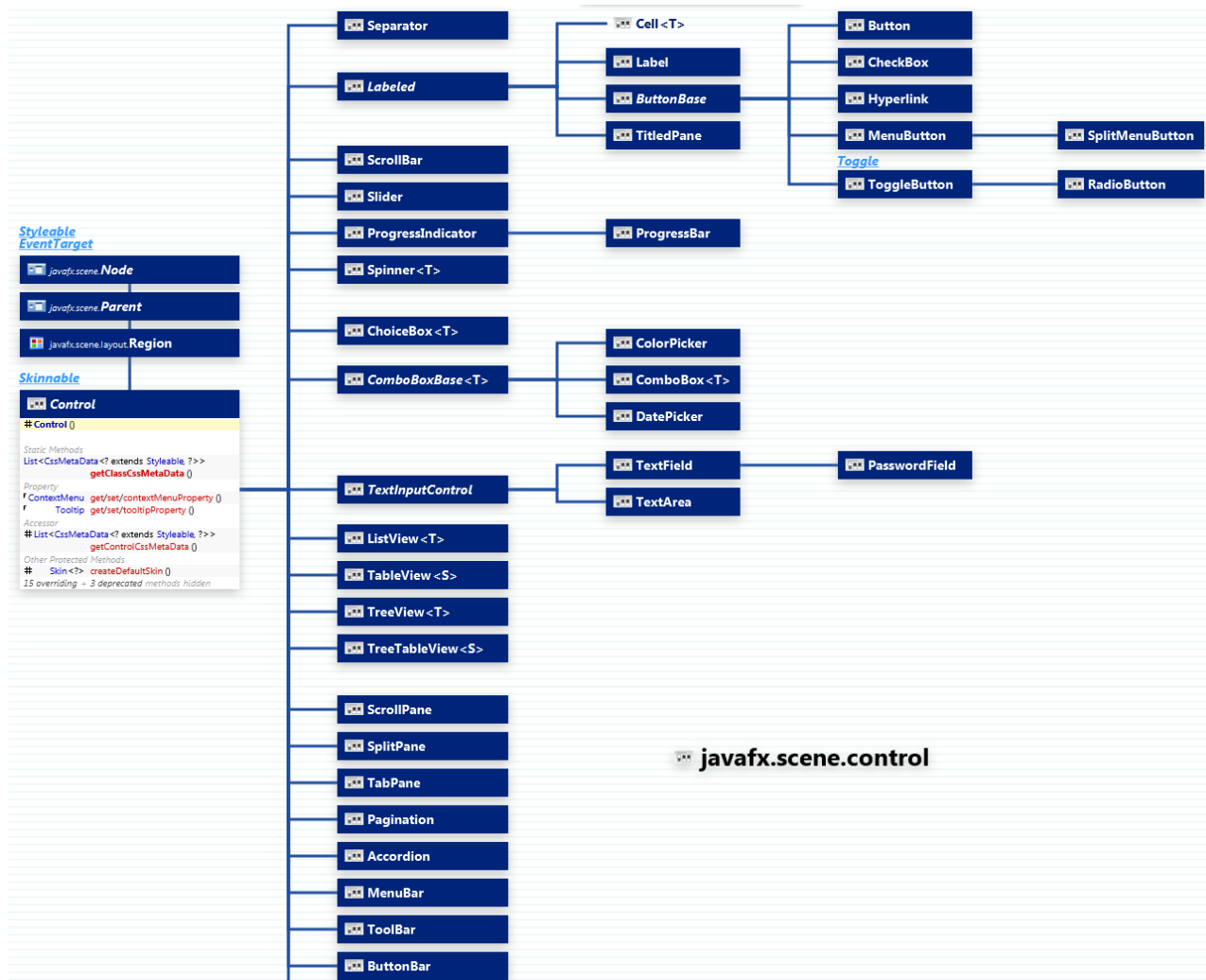
Nodes

- ▶ **The Scene is populated with a tree of Nodes**
 - ▶ Layout components (Panels)
 - ▶ UI Controls
 - ▶ Charts
 - ▶ Shapes
- ▶ **Nodes have Properties**
 - ▶ Visual (size, position, z-order, color, ...)
 - ▶ Contents (text, value, data sets, ...)
 - ▶ Programming (event handlers, controller)
- ▶ **Nodes generate Events**
 - ▶ UI events
- ▶ **Nodes can be styled with CSS**

Panes



Controls



Properties

- ▶ Extension of the Java Beans convention
 - ▶ May be used also outside JavaFX
- ▶ Encapsulate properties of an object
 - ▶ Different types (string, number, object, collection, ...)
 - ▶ Set/Get
 - ▶ Observe changes
 - ▶ Support lazy evaluation
- ▶ Each Node has a large set of Properties

Properties	
Type	Property and Description
BooleanProperty	cancelButton A Cancel Button is the button that receives a keyboard VK_ESC press, if no other node in the scene contains a button that receives a keyboard VK_ESC press.
BooleanProperty	defaultButton A default Button is the button that receives a keyboard VK_ENTER press, if no other node in the scene contains a button that receives a keyboard VK_ENTER press.

Properties inherited from class <code>javafx.scene.control.ButtonBase</code>
armed, onAction

Properties inherited from class <code>javafx.scene.control.Labeled</code>
alignment, contentDisplay, ellipsisString, font, graphic, graphicTextGap, labelPadding, mnemonicParsing, textFill, textOverrun, text, underline, wrapText

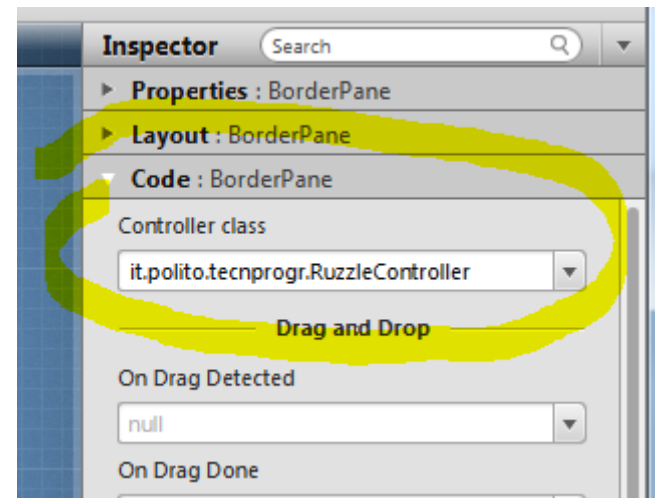
Properties inherited from class <code>javafx.scene.control.Control</code>
contextMenu, height, maxHeight, maxWidth, minHeight, minWidth, prefHeight, prefWidth, skinClassName, skin, text

Properties inherited from class <code>javafx.scene.Parent</code>
needsLayout

Properties inherited from class <code>javafx.scene.Node</code>
blendMode, boundsInLocal, boundsInParent, cacheHint, cache, clip, cursor, depthTest, disabled, disable, effect, eventDispatcher, focused, focusTraversable, hover, id, inputMethodRequests, layoutBounds, layoutX, layoutY, localToParentTransform, localToSceneTransform, managed, mouseTransparent, onContextMenuRequested, onDragDone, onDragDropped, onDragEntered, onDragExited, onDragOver, onInputMethodTextChanged, onKeyPressed, onKeyTyped, onMouseClicked, onMouseDragEntered, onMouseDragExited, onMouseDragged, onMouseDragOver, onMouseEntered, onMouseExited, onMouseMoved, onMousePressed, onMouseReleased, onRotate, onRotationFinished, onRotationStarted, onScrollFinished, onScroll, onScrollStarted, onSwipeDown, onSwipeLeft, onSwipeRight, onSwipeUp, onTouchMoved, onTouchPressed, onTouchReleased, onTouchStationary, onZoomFinished, onZoom, onZoomStarted, opacity, pickOnBounds, pressed, rotate, rotationAxis, scaleX, scaleY, scaleZ, scene, style, translateX, translateY, visible

Defining a Controller class

- ▶ The Root element of the scene graph may specify a **fx:controller** attribute
 - ▶ `<BorderPane id="BorderPane" xmlns:fx="http://javafx.com/fxml" fx:controller="it.polito.tecnprogr.RuzzleController">`



Injection of Node references

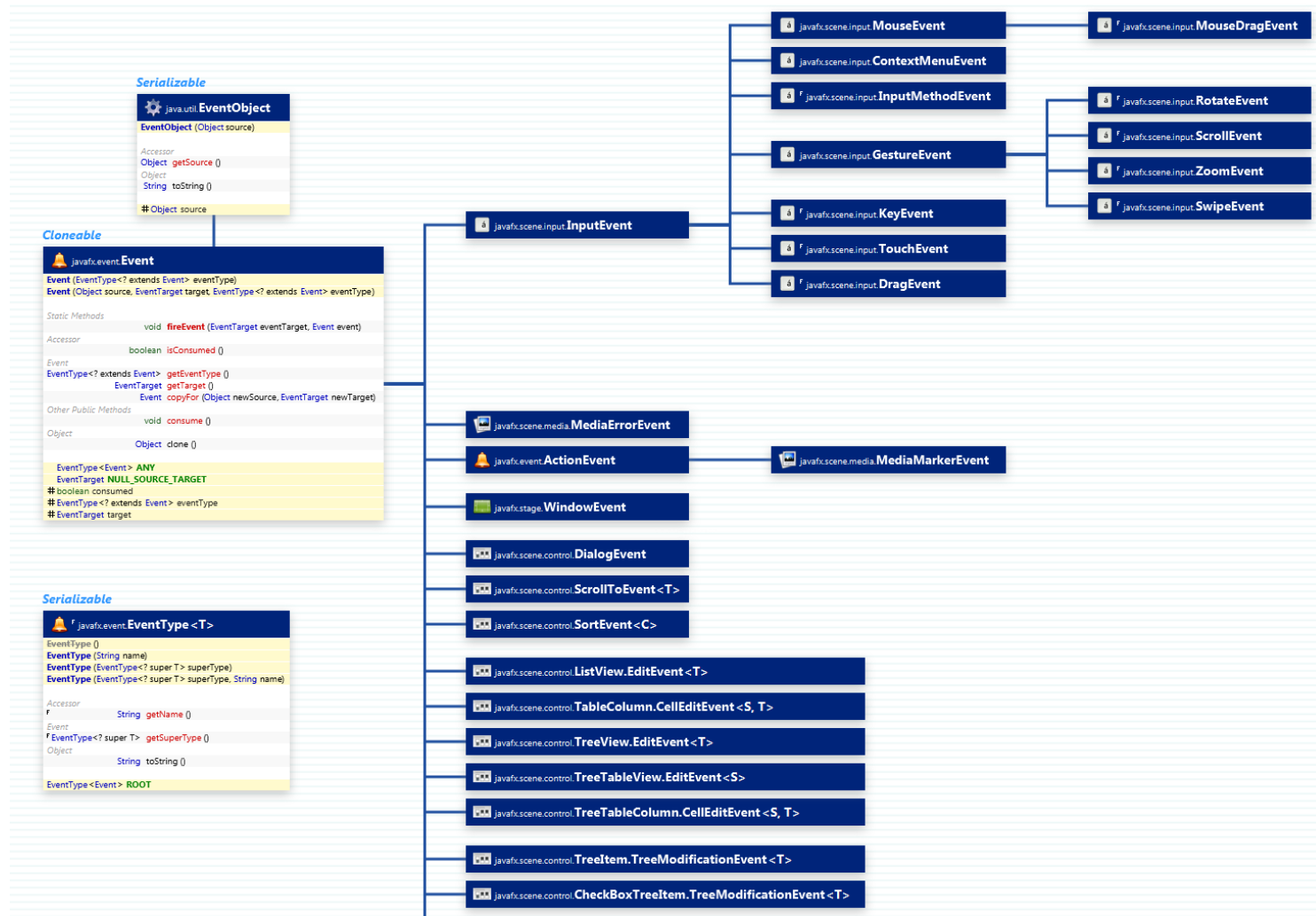
- ▶ The controller code may directly access various Nodes in the associated scene graph
- ▶ The attribute `@FXML` associates a Node variable with the corresponding node, with the same `fx:id` value as the variable name
- ▶ Try:View | Show Sample Controller Skeleton on the Scene Builder!

```
@FXML // fx:id="theTitle"  
private Label theTitle;
```


Events

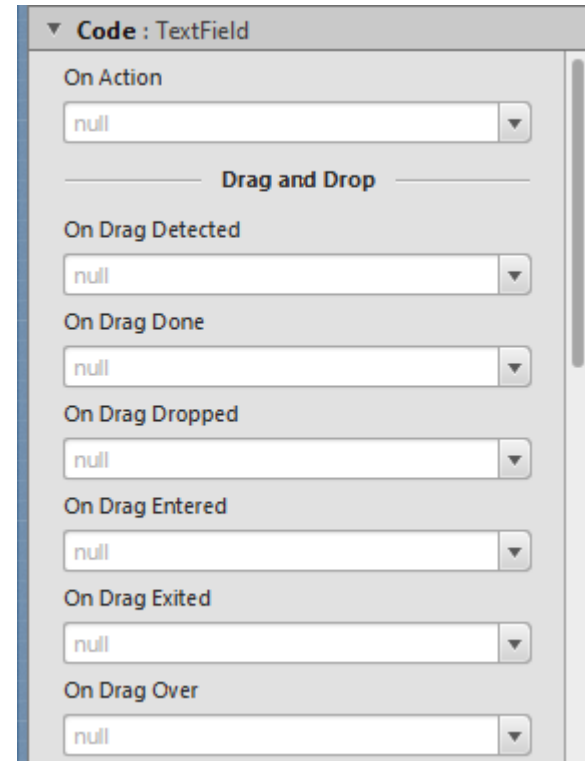
- ▶ FX Event (`javafx.event.Event`):
 - ▶ Event Source => a Node
 - ▶ Event Target
 - ▶ Event Type
- ▶ Usually generated after some user action
- ▶ Event Types
- ▶ You can define **event handlers** in your application

Event types



Registration of Event Handlers

- ▶ In FXML, you may set an event handler through attributes
 - ▶ `onAction`, `onKeyTyped`, `onMouseClicked`, ... hundreds more ...
- ▶ The value should be the `#name` of a method in the controller class
 - ▶ With the right signature for the event type








```
<Button fx:id="cercaBtn"  
onAction="#doCercaParola"  
text="Cerca" />
```

```
@FXML  
void doCercaParola (  
ActionEvent event ) {
```

Licenza d'uso



- ▶ Queste diapositive sono distribuite con licenza Creative Commons “Attribuzione - Non commerciale - Condividi allo stesso modo (CC BY-NC-SA)”
- ▶ Sei libero:
 - ▶ di riprodurre, distribuire, comunicare al pubblico, esporre in pubblico, rappresentare, eseguire e recitare quest'opera 
 - ▶ di modificare quest'opera 
- ▶ Alle seguenti condizioni:
 - ▶ **Attribuzione** — Devi attribuire la paternità dell'opera agli autori originali e in modo tale da non suggerire che essi avallino te o il modo in cui tu usi l'opera. 
 - ▶ **Non commerciale** — Non puoi usare quest'opera per fini commerciali. 
 - ▶ **Condividi allo stesso modo** — Se alteri o trasformi quest'opera, o se la usi per crearne un'altra, puoi distribuire l'opera risultante solo con una licenza identica o equivalente a questa. 
- ▶ <http://creativecommons.org/licenses/by-nc-sa/3.0/>