> 由 KiOii (*_EM_Cpper*)整理。 (KiOii (*_EM_Cpper*) makes this note.)



# Managing Stages

**In this chapter**

- 如何获得屏幕的详细信息，如数字、分辨率( resolution )和尺寸( dimension )
- 什么是stage，如果设置stage的边界( bound )和风格( style )
- 如何移动一个 未装饰的( undecorated )stage
- 如何设置stage的模式( modality )和不透明度( opacity )
- 如何调整 stage大小和全屏显示( full-screen mode )

## 4.1 Knowing the Details of Your Screeens

- **Screen** class :
  - package : javafx.stage
  - purpose : get details
    - such as : dots-per-inch (DPI) setting and dimensions of user screens

      > 每一英寸长度中，取样、可显示或输出点的数目
      >
      > 1英寸(in)=2.54厘米(cm)
  - 分类:
    - primary screen
    - noprimary screen

      > 如果多个屏幕被连接到电脑上，其中一个屏幕被称为主屏幕( pirmary)，而其他屏幕则被称为非主屏幕( noprimary )
  - static method
    - **getPrimary** : get the reference of the Screen object for the primary monitor

> 获取主监视器（主屏幕）的对象引用

```
Screen primaryScreen = Screen.getPrimary();
```

- **getScreens** : returns an ObservableList of Screen objects

```
ObservableList<Screen> screenList = Screen.getScreens();
```

- instance method
  - **getDpi** : get the resolution of a screen in DPI

```
Screen primaryScreen = Screen.getPrimary();
double dpi = primaryScreen.getDpi();
```

  - **getBounds** and **getVisualBounds** : get the bounds and visual bounds

> return a Rectangle2D object

# 4.2 What Is a Stage?

- **Stage** class
  - package : javafx.stage
  - define : a top-level container that hosts a scene

> scene cinsists of visual elements

  - 分类
    - primary stage : created by the platform and passed to start method
    - You can create additional stages as needed
  - super class : **Window** class

> x, y, width, height, opcity properties
>
> show ()、hide()

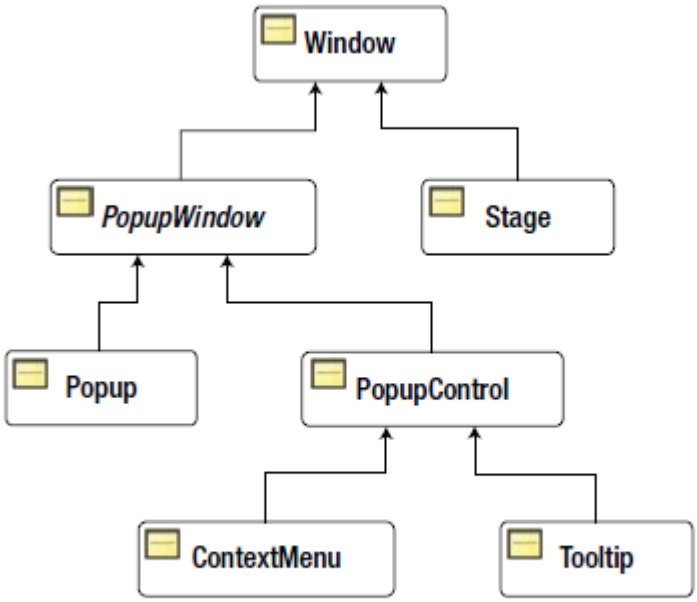



*Figure 4-1. The class diagram for the Stage class*

- method
  - close : the same effect as hide() of **Window** class
  - show : show stage

> 开始详细介绍处理stage

## 4.3 Showing the Primary Stage

- method
  - show : show it
  - close : close a showing stage

    > 前提必须是showing

  - setTitle : 设置标题

## 4.4 Setting the Bounds of a Stage

- the bounds of a stage
  - consist of four properties : x, y, width, height
- stage size and position
  - 当一个stage没有一个scene时，他的position和size取决于platform
- method
  - setScene : 设置scene

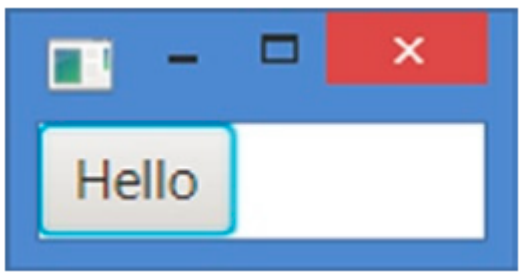
    > 你不能创建一个没有root node的scene

```
package com.javafx.test;

import javafx.application.Application;
import javafx.stage.Stage;
import javafx.scene.Group;
import javafx.scene.Scene;

public class TestStageClass extends Application
{
    public static void main(String[] args)
    {
        Application.launch(args);
    }
    @Override
    public void start(Stage stage)
    {
        // Write code here...
        stage.setTitle("Stage with an Empty Scene");
        Scene scene = new Scene(new Group());
        stage.setScene(scene);
        stage.show();
    }
}
```

- default for scene
  - white background
- example
  - add a button to the scene

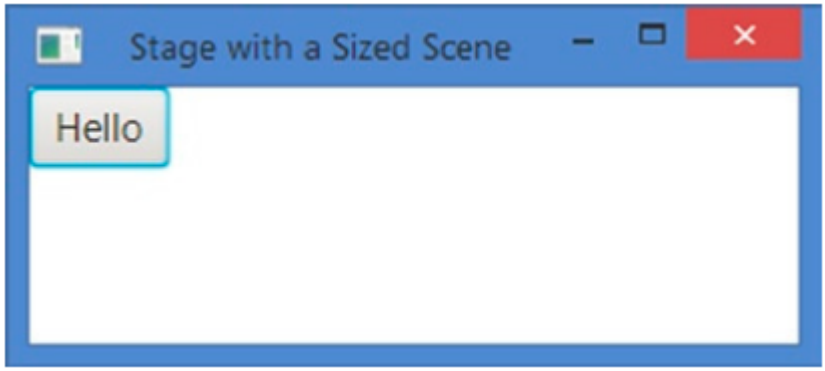
```
import javafx.scene.control.Button;
...
Group root = new Group(new Button("Hello"));
Scene scene = new Scene(root);
...
```

> 一旦有了node elements，默认的scene大小被设置为能显示node的最小标准



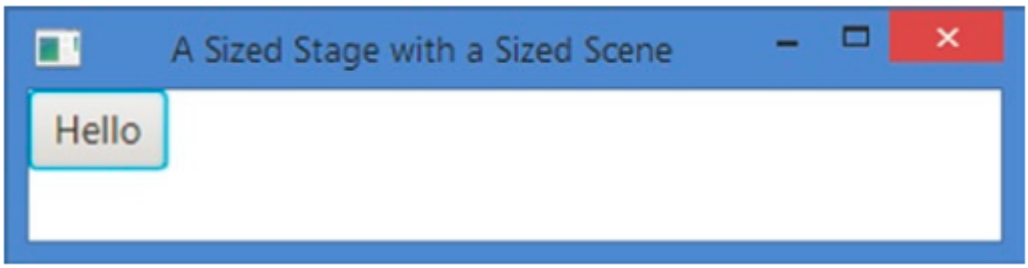

  - add a button to the scene and set the scene width and height (300 and 100)

```
...
Group root = new Group(new Button("Hello"));
Scene scene = new Scene(root,300,100);
...
```




- add button,set scene size and stage size

```
Group root = new Group(new Button("Hello"));
Scene scene = new Scene(root,300,100);
stage.setScene(scene);
stage.setWidth(400);
stage.setHeight(100);
stage.show();
```




- sizeToScene() : 充满整个scene

  > 当同步scene和stage时，这将是非常有效的操作

- centerOnScreen() : center the stage on the screen
- wraning : 只有当show之后，stage的x和y才会被确认下来！！！

  > show之后也可以设置x和y

# 4.5 Intializing the Style of a Stage

- area of a stage : two parts
  - content area ( 内容 ) : displays the visual content of its scene
  - decorations( 装饰 ):: title bar and borders ( 表现形式取决于平台)
    - provide additional features

      > 提供了特性 而不仅仅是美观而已

- style attribute of a stage: 决定了背景颜色和装饰
  - five types of stages
    - Decorated 装饰
      - solid white background and **platform decorations**
    - Undecorated 未装饰
      - solid white background and **no decorations**
    - Transparent 透明的
      - transpartent background and **no decorations**
    - Unified 统一
      - **no border** between the client area and decoration, has **platform decorations**
      - 为了看到效果，scene的背景设置为Color.TRANSPARENT
    - Utility 实用
      - has a solid white background and **minimal platform decorations**
  - wraning : style 只能指定 decoration,背景颜色由scene指定( 默认白色 )

    > 如果你设置了stage的style为TRANSPARENT，stage将得到scene的背景
    >
    > 如果想得到真正的透明stage，需要通过setFill设置scene为null
    >
    > scene.setFill(null) + stage.initStyle(TRANSPARENT) : 完全看不到
    >
    > stage.initStyle(TRANSPARENT) : 一张白纸
  - five constants in the **StageStyle** enum
    - StageStyle.DECORATED
    - StageStyle.UNDECORATED
    - StageStyle.TRANSPARENT
    - StageStyle.UNIFIED
    - StageStyle.UTILITY
  - **initStyle**(StageStyle style) method : set the style of a stage

# 4.6 Moving an Undecorated Stage

> 未装饰的窗口（没有title bar）如何移动
>
> 如果您将stage更改为透明，那么您将需要通过将鼠标拖动到消息label来拖动舞台，因为透明区域将不会响应鼠标事件

```
package com.javafx.test;

import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.input.MouseEvent;
import javafx.scene.layout.VBox;
import javafx.stage.Stage;
import javafx.stage.StageStyle;
```

```
import javafx.scene.text.Text;

public class MovingUndecorated  extends Application
{
    private Stage stage;
    private double dragOffsetX;
    private double dragOffsetY;
    private VBox root;
    public static void main(String[] args)
    {
        Application.launch(args);
    }
    @Override
    public void start(Stage stage)
    {
        // Store the stage reference in the instance variable to
        // use it in the mouse pressed event handler later.
        this.stage = stage;
        Label msgLabel = new Label("Press the mouse button and drag.");
        Button closeButton = new Button("Close");
        closeButton.setOnAction(e -> stage.close());
        root = new VBox();
        root.getChildren().addAll(msgLabel, closeButton);
        Scene scene = new Scene(root, 300, 200);
        // Set mouse pressed and dragged even handlers for the scene
        scene.setOnMousePressed(e -> handleMousePressed(e));
        scene.setOnMouseDragged(e -> handleMouseDragged(e));
        stage.setScene(scene);
        stage.setTitle("Moving a Stage");
        stage.initStyle(StageStyle.UNDECORATED);
        stage.show();
    }
    protected void handleMousePressed(MouseEvent e)
    {
        // Store the mouse x and y coordinates with respect to the
        // stage in the reference variables to use them in the drag event
        this.dragOffsetX = e.getScreenX() - stage.getX();
        this.dragOffsetY = e.getScreenY() - stage.getY();
        root.getChildren().add(new Text("offx :"+this.dragOffsetX +
                " offy :"+this.dragOffsetY));
    }
    protected void handleMouseDragged(MouseEvent e)
    {
        // Move the stage by the drag amount
        stage.setX(e.getScreenX() - this.dragOffsetX);
        stage.setY(e.getScreenY() - this.dragOffsetY);
    }
}
```

> MouseEvent.getScreenX 获取鼠标相对屏幕的位置
>
> dragOffSetX存储了鼠标相对stage的位置，一旦发生drag事件

> 就用当前实时鼠标位置减去dragOffSetX得到实时的stage的x

# 4.7 Initailizing Modality of a Stage

> 初始化stage模式

- type of windows : modal and modeless
  - **Modality** class
    - NONE : 无模式窗口
    - WINDOW_MODAL： 用于父子窗口
    - APPLICATION_MODAL： 用于急需当前处理窗口
  - **initModality** : set modality of a stage

    > The modality of a stage must be set before it is shown
    >
    > 否则就抛出异常，primary stage设置Modality也会抛出异常

    ```
    Stage stage = new Stage();
    stage.initModality(Modality.WINDOW_MODAL);
    ```

- stage owner : a stage can have an owner

  > An owner of a Stage is another Window

  - **initOwner** : set an owner of a stage

    > 设置同样也是在show之前

  - 如果其所有者( 主人 owner )被最小化或隐藏，那么一个stage将被最小化或隐藏

# 4.8 Setting the Opacity of a Stage

- setOpacity : 0.0 ~ 1.0
- getOpacity

> 并不是所有platform都有透明效果

```
Stage stage = new Stage();
stage.setOpacity(0.5);
```

# 4.9 Resizing a Stage

- **setResizable** : set can resize or cannot resize (默认是 true )

  > 这边是限制user，即使设置false，我们也可以编程来设置size

- 限制变化
  - setMinWidth
  - setMinHeight
  - setMaxWidth
  - setMaxHeight

# 4.10 Showing a Stage in Full-Screen Mode

- fullScreen property : specified wherther a stage should be displayed in full-screen mode

  > 全屏幕模式的实现取决于平台和配置文件
  >
  > 如果平台不支持全屏模式，JavaFX运行时将通过显示最大化和未修饰的阶段来模拟它

- setFullScreen : enter full-screen mode
- isFullScreen : check if a stage is in full-screen mode

# 4.11 Showing a Stage and Wairing for It to Close

> 您通常希望显示一个对话框，并暂停进一步的处理，直到它被关闭

```
Option userSelection = messageBox("Close","Do you want to exit?", YESNO);
if(userSelction == YES)
{
    stage.close();
}
```

- **showAndWait** method : stop processing the current event,start a nested event loop to process other events

  > show 是立即返回，showAndWait则不是（只能被 JavaFX Application Thread调用）
  >
  > 不能被primary stage调用

  ```
  package com.javafx.test;

  import javafx.application.Application;
  import javafx.scene.Scene;
  import javafx.scene.control.Button;
  import javafx.scene.layout.VBox;
  import javafx.stage.Stage;

  public class ShowAndWaitApp extends Application
  {
      protected static int counter = 0;
      protected Stage lastOpenStage;
      public static void main(String[] args)
      {
          Application.launch(args);
      }
      @Override

      public void start(Stage stage)
  ```

```
    {
        VBox root = new VBox();
        Button openButton = new Button("Open");
        openButton.setOnAction(e -> open(++counter));
        root.getChildren().add(openButton);
        Scene scene = new Scene(root, 400, 400);
        stage.setScene(scene);
        stage.setTitle("The Primary Stage");
        stage.show();
        this.lastOpenStage = stage;
    }

    private void open(int stageNumber)
    {
        Stage stage = new Stage();
        stage.setTitle("#" + stageNumber);
        Button sayHelloButton = new Button("Say Hello");
        sayHelloButton.setOnAction(
        e -> System.out.println("Hello from #" + stageNumber));
        Button openButton = new Button("Open");
        openButton.setOnAction(e -> open(++counter));
        VBox root = new VBox();
        root.getChildren().addAll(sayHelloButton, openButton);
        Scene scene = new Scene(root, 200, 200);
        stage.setScene(scene);
        stage.setX(this.lastOpenStage.getX() + 50);
        stage.setY(this.lastOpenStage.getY() + 50);
        this.lastOpenStage = stage;
        System.out.println("Before stage.showAndWait(): " + stageNumber);
        // Show the stage and wait for it to close
        stage.showAndWait();
        System.out.println("After stage.showAndWait(): " + stageNumber);
    }
}
```

# Summary

- Screen : get detail of screen
- Scene : consists of visual elements
- Stage bound : pistion and size
- Stage area : content and decorations
    - decoration : **StageStyle** : decorated , undecorated, transparent, unified, utility
- type of window : modal and modeless
    - modal : **Modality** : none, window modal, application modal
- Stage opacity : how much you can see through the stage
- setFullScreen

- showAndWait