**一**.**javaFX Preloader**: JavaFX在Application加载的时候提供了Proloader应用，可以展现loading process bar或者others notification。

**二.** [**Model-View-Controller** (MVC)](http://zh.wikipedia.org/wiki/MVC)是一个非常重要的软件设计原则。按照MVC模式可以将我们的应用程序划分成3个部分，然后为这每一部分建立自己的包 (在源代码文件夹上右键， 选择 新建 | 包):

* ch.makery.address - 放置所有的控制器类(也就是应用程序的业务逻辑)
* ch.makery.address.model - 放置所有的模型类
* ch.makery.address.view - 放置所有界面和控件类

**注意:** view包里可能会包含一些控制器类，它可以直接被单个的view引用，我们叫它 **视图-控制器**。

**三**. 有两种方式来创建用户界面，一终是能过XML文件来定义，另外一种就是直接通过java代码来创建. 我们这里将使用XML的方式来创建大部分的界面，这种方式将会更好的将业务逻辑和界面分开，以保持代码的简洁。使用Scene Builder(所见即所得)来编辑我们的XML布局文件，它可以避免我们直接去修改XML文件。

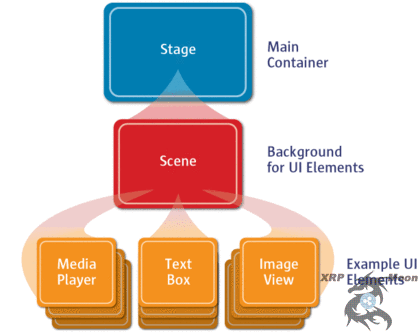
**四**. 点击菜单中的 Preview | Show Preview in Window 可以预览你设计好的界面，试着缩放预览的界面，你会发现TableView会随着窗口的缩放而变化。

**五**. 选择这个 TableView ，在右边面板中将它的 Column Resize Policy（列调整策略） 修改成 constrained-resize（约束调整） (同样是在属性设置项里面)。确保这个TableView的列能够铺满所有的可用空间。

**六**. 注意: 添加一个控件到已经存在的行里面去，你可在这行的行号上右键选择 “Add Row”。

**七**. 你可能注意到了IDE生成的 MainApp.java 继承自 Application 同时包含了两个方法， 这是一个JavaFX应用程序的最基本的代码结构，这里最重要的方法是 start(Stage primaryStage) ，它将会在应用程序运行时通过内部的 main 方法自动调用。

正如你所看到的，这个start(...) 方法会接收一个 Stage 类型的参数，下面的图向你展示了一个JavaFX应用程序的基本结构。

  
\*Image Source: http://www.oracle.com\*

**一切看起来象是剧场里表演**: 这里的 Stage 是一个主容器，它就是我们通常所认为的窗口（有边，高和宽，还有关闭按钮）。在这个 Stage 里面，你可以放置一个 Scene，当然你可以切换别的 Scene，而在这个 Scene 里面，我们就可以放置各种各样的控件。

**八.**对整个项目进行clean build，否则会找不到加载的fxml的路径

**九**. 在JavaFX中,对一个模型类的所有属性使用 [Properties](http://docs.oracle.com/javase/8/javafx/api/javafx/beans/property/Property.html)是很常见的. 一个 Property 允许我们, 打个比方, 当 lastName 或其他属性被改变时自动收到通知, 这有助于我们保持视图与数据的同步，阅读 [Using JavaFX Properties and Binding](http://docs.oracle.com/javase/8/javafx/properties-binding-tutorial/binding.htm) 学习更多关于Properties 的内容。

[LocalDate](http://docs.oracle.com/javase/8/docs/api/java/time/LocalDate.html), 我们使用了 birthday 类型, 这是一个新的部分在 [Date and Time API for JDK 8](http://docs.oracle.com/javase/tutorial/datetime/iso/).

### 十. ObservableList

我们处理JavaFX的view classes需要在人员列表发生任何改变时都被通知.  
这是很重要的,不然视图就会和数据不同步.为了达到这个目的,JavaFX引入了一些新的[集合类](http://docs.oracle.com/javase/8/javafx/collections-tutorial/collections.htm).

在这些集合中, 我们需要的是ObservableList. 将以下代码增加到MainApp类的开头去创建一个新的ObservableList. 我们也会增加一个构造器去创建一些样本数据和一个公共的getter方法:  
**十一**. 所有fxml文件需要访问的属性和方法必须加上 @FXML 注解.实际上,只有在私有的情况下才需要, 但是让它们保持私有并且用注解标记的方式更好!

* initialize() 方法在fxml文件完成载入时被自动调用. 那时, 所有的FXML属性都应已被初始化.
* 我们在表格列上使用setCellValueFactory(...) 来确定为特定列使用Person对象的某个属性. 箭头 -> 表示我们在使用Java 8的 *Lambdas* 特性. (另一个选择是使用 [PropertyValueFactory](http://docs.oracle.com/javase/8/javafx/api/), 但它不是类型安全的).=

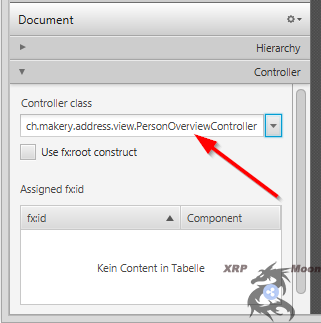
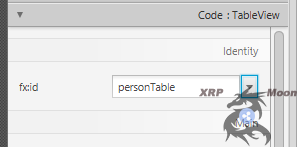
**十二.**

### 连接 MainApp 和 PersonOverviewController

setMainApp(...) 必须被 MainApp 类调用. 这让我们可以访问MainApp对象并得到Persons的列表和其他东西. 用以下代码替换showPersonOverview() 方法. 它包含了新增的两行:

## 将View与Controller挂钩

我们快要完成了! 但是有件小事被遗漏了: 至今没有告诉 PersonOverview.fxml使用的是哪个控制器以及元素与控制器中的属性的对应关系.

1. 使用\*SceneBuilder\* 打开 PersonOverview.fxml.
2. 打开左边的 Controller 组选择PersonOverviewController 作为 **controller class**.  
   
3. 在 Hierarchy 组选择TableView 并选择 Code 组将 personTable 作为 **fx:id**.  
   
4. 对列做相同的事并且将 firstNameColumn and lastNameColumn分别作为 **fx:id** .
5. 对在第二列的 **each label** , 选择对应的 **fx:id**.  
   
6. 重要事项: 回到eclipse并且 **refresh the entire AddressApp project** (F5). 这是必要的因为有时候eclipse并不知道在Scene Builder中作出的改变.

**十三.**

## 删除按钮

我们的用户接口已经包含一个删除按钮，但是没有任何功能。我们能在\*SceneBuilder\*中的按钮上选择动作。在我们控制器中的任何使用@FXML（或者它是公用的）注释的方法都可以被\*Scene Builder\*访问。因此，让我们在PersonOverviewController类的最后添加一个删除方法。

**十四.**

### 为什么是XML？

持久性数据的一种最常用的方法是使用数据库。数据库通常包含一些类型的关系数据（例如：表），当我们需要保存的数据是对象时。这称[object-relational impedance mismatch](http://wikipedia.org/wiki/Object-relational_impedance_mismatch)。匹配对象到关系型数据库表有很多工作要做。这里有一些框架帮助我们匹配（例如：[Hibernate](http://www.hibernate.org/)，最流行的一个）。但是它仍然需要相当多的设置工作。

对于简单的数据模型，非常容易使用XML。我们使用称为[JAXB](https://jaxb.java.net/)（\*\*J\*\*ava **A**rchitecture for **X**ML **B**inding）的库。只需要几行代码，JAXB将允许我们生成XML输出，如下所示：

**十五.**

### 使用JAXB

JAXB已经包含在JDK中。这意味着我们不需要包含任何其它的库。

JAXB提供两个主要特征：\*\*编列(marshal)\*\*Java对象到XML的能力，\*\*反编列(unmarshal)\*\*XML到Java对象。

为了让JAXB能够做转换，我们需要准备我们的模型。

**十六.**

现在，再次测试。试着保存和加载XML文件。在重启之后，它应该自动加载最后使用的文件。

## 它如何工作

让我们看下它是如何一起工作的：

1. 应用程序使用MainApp中的main(…)方法启动。
2. 调用public MainApp()构造函数添加一些样例数据。
3. 调用MainApp的start(…)方法，调用initRootLayout()从RootLayout.fxml中初始化根布局。fxml文件有关于使用控制器的信息，连接视图到RootLayoutController。
4. MainApp从fxml加载器中获取RootLayoutController，传递自己的引用到控制器中。使用这些引用，控制器随后可以访问MainApp的公开方法。
5. 在initRootLayout方法结束，我们试着从Perferences中获取\*最后打开的人员文件\*。如果Perferences知道有这样一个XML文件，我们将从这个XML文件中加载数据。这显然会覆盖掉构造函数中的样例数据。

#### 日历控制器如何工作

1. 控制器需要从FXML文件中访问两个元素:

* barChar：它有String和Integer类型。String用于x轴上的月份，Integer用于指定月份中人员的数量。
* xAxis：我们使用它添加月字符串

1. initialize() 方法使用所有月的列表填充x-axis。
2. setPersonData(…)方法将由MainApp访问，设置人员数据。它遍历所有人员，统计出每个月生日的人数。然后它为每个月添加XYChart.Data到数据序列中。每个XYChart.Data对象在图表中表示一个条形。