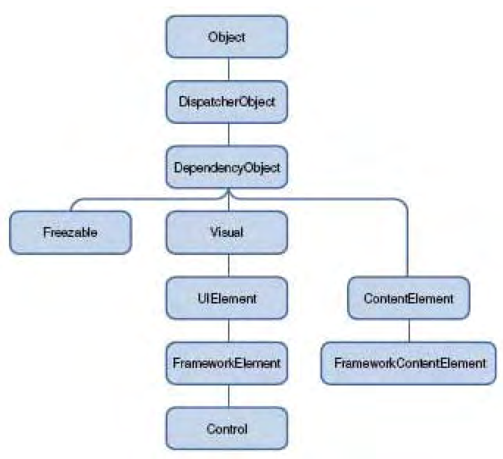
Lectures on WPF
Controls

Controls, Panels and other FrameworkElements

FrameworkElement extends `UIElement` and provides support for many scenarios of the user interface.
Most classes dealing with the user interface derive from `FrameworkElement` either directly or through the intermediate base classes **Control** or **Panel**.
The `Control` class is the most important base class for many elements of the user interface.
See: **Control_Classes.pdf** which has been cut out from the WPF class hierarchy handbill from T.C. Huber's book: **www.galileocomputing.de**.

Class Hierarchy above the Control Class



`Object`: The base class for all .NET classes.
`DispatcherObject`: The base class for any object that wishes to be accessed only on the thread that created it.
A Dispatcher maintains a prioritized queue of work items for a specific thread. Most WPF classes derive from `DispatcherObject`, and are therefore inherently thread-unsafe.
`DependencyObject`: The base class for any object that can support dependency properties. It defines the `GetValue` and `SetValue` methods of dependency properties.
`Visual`: The base class for all objects that have their own visual representation. Its primary role is to provide rendering support.
`UIElement`: The base class for all visual objects with support for routed events, command binding, layout, and focus.
`FrameworkElement`: The base class that adds support for styles, data binding, resources, and a few common mechanisms for Windows-based controls such as tooltips and context menus.
`Control`: The base class for familiar controls such as `Button` and `ListBox` adds many properties to its `FrameworkElement` base class, such as `Foreground`, `Background`, and `FontSize` and the Template property which defines the complete appearance. See: **Guided tour of the WPF class hierarchy**

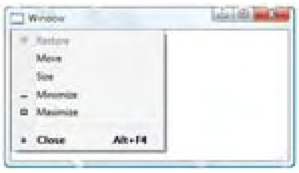
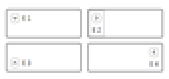
`Freezable`: The base class for objects that can be "frozen" into a read-only state for performance reasons. `Freezables`, once frozen, can even be safely shared among multiple threads, unlike all other `DispatcherObjects`. Frozen objects can never be unfrozen, but you can clone them to create unfrozen copies.
`ContentElement`: A base class similar to `UIElement`, but for pieces of content that don't have rendering behavior on their own. Instead, ContentElements are hosted in a `Visual`-derived class to be rendered on the screen.
`FrameworkContentElement`: The analog to `FrameworkElement` for content.

The term *element* is often used to refer to an object that derives from `UIElement` or `FrameworkElement`.

ContentControls

The most important (and most simple) subclass of class `Control` is ContentControl. It holds and displays just one single element = one piece of content (which can be deeply nested). This one-child limit is what differentiates `ContentControls` from `ItemControls` and other `FrameworkElements` such as `Panel`, `Page`, `TextBlock` etc.
List of ContentControls:
1. `Window` with its child `NavigationWindow`,
2. `ButtonBase` with its children `Button`, `RepeatButton`, `RadioButton`, `CheckBox`,
3. `HeaderedContentControl` with its children `Groupbox`, `Expander`, `TabItem`,
4. `Label` 5. `Frame` and 6. `ScrollViewer`.
All `Controls` having more than one child are no `ContentControls`.

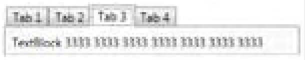
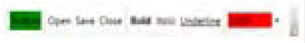
Table of important ContentControls:

Window	The point of interaction between a user and a standalone application. A window has two distinct areas: 1. A non-client area, which hosts the windows adornments, including an icon, title, System menu, minimize button, maximize button, restore button, close button, and a border. 2. A client area, which hosts application-specific content.	
Button	Is one of the most basic element of a user interface. A Button inherently reacts to a MouseClick event.	
RadioButton	Is usually used as an item in a group of RadioButton controls. However, it is possible to create a single RadioButton. When a RadioButton is selected, it cannot be cleared by clicking it. When RadioButton elements are grouped, the buttons are mutually exclusive. A user can select only one item at a time within a RadioButton group. Important difference to CheckBox: When a RadioButton is selected all other RadioButtons will be automatically deselected.	
CheckBox	A binary button that can be checked and unchecked independently of other CheckBoxes in a group.	
GroupBox	The `GroupBox` is the simplest of the `HeaderedContentControls`: a box with rounded corners and a title. It is often used to group small sets of related controls such as `Buttons` or `CheckBoxes` under a common title. → Sample	
Expander	The `Expander` is a `HeaderedContentControl` which wraps a region of content as `TabControl` does but shows or hides the content by clicking a small arrow button. `Expanders` are used in online help and on web pages. → Sample	
Label	Mostly used to show short text. Provides support for quick keyboard access = mnemonics.	
Frame	Provides the ability to navigate to content as `Page` does. A `Source` property allows to set the URI for the desired content and `Frame` returns an object that contains the content.	

ItemControls

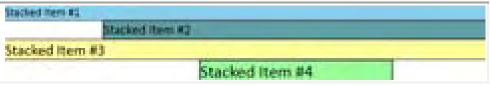
An **ItemsControl** is a type of `Control` that contains a collection of multiple items, such as strings, objects, or other elements. Adding a child to an `ItemsControl` object adds it to an `ItemCollection`.
List of ItemControls:
1. `Selector` with its children `ListBox`, `ComboBox` and `TabControl`,
2. `MenuBase` with its children `Menu` and `ContextMenu`,
3. `HeaderedItemsControl` with its child `ToolBar`,
4. `StatusBar` and 5. `TreeView`.

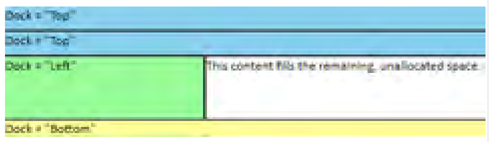
Table of important ItemControls:

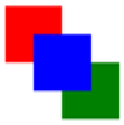
ListBox	Contains a list of selectable items. All items of a `ListBox` are visible (unlike the `ComboBox`). The `SelectionMode` property determines whether more than one item in the `ListBox` is selectable at a time → Properties: `Single` (the default), `Multiple`, or `Extended`. → **Sample**	Item 1 Item 2 Item 3 Item 4 Item 5
ComboBox	Selection in drop-down list form that can be shown or hidden by clicking the arrow on the control. Otherwise it's very similar to a `ListBox`. → **Sample**	
TabControl	The `TabControl` is a `HeaderedContentControl` useful for minimizing screen space usage while an application wants to expose a large amount of information. A `TabControl` consists of multiple `TabItem` objects that share the same screen space. Only one `TabItem` in a `TabControl` is visible at a time. When a user selects the tab of a `TabItem`, the contents of that `TabItem` become visible and the contents of the other `TabItem` objects are hidden. → **Sample**	 Tab 1 Tab 2 Tab 3 Tab 4 TextBlock 3333 3333 3333 3333 3333 3333 3333
Menu	Presents a list of items that specify commands or options for an application. Typically, clicking an item on a menu opens a submenu or causes an application to carry out a command. An item in a menu can be anything that can be added to an ItemCollection. → **Sample**	
ToolBar	Container for a horizontal or vertical group of controls with an overflow menu in case the ToolBar doesn't fit in its window. → **Sample**	
StatusBar	Container for a horizontal group of noninteractive elements such as TextBlocks, Images and a ProgressBar. → **Sample**	Some Text More text Much more text

Panels

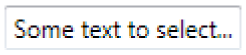
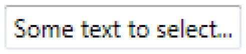
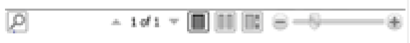
A Panel is a type of `FrameworkElement` that positions and arranges one or more child objects.
List of panel controls: `StackPanel, DockPanel, UniformGrid, Grid, Canvas, TabPanel, ToolBarOverflowPanel, ToolBarPanel, VirtualizingPanel, VirtualizingStackPanel, WrapPanel`
Important Panel Controls are:

StackPanel	Allows to stack elements both vertically, which is the default setting, or horizontally.	 Stacked Item #1 Stacked Item #2 Stacked Item #3 Stacked Item #4

DockPanel	Arranges child elements on top, left, right and bottom within the client area = Dock property. A set of child elements with the same Dock property values are positioned differently depending on the order. The last child element fills the remaining space if `LastChildFill` is `true`.	 Dock = "Top"
		Dock = "Top"
		Dock = "Left"
		This content fills the remaining, unallocated space.
		Dock = "Bottom"

UniformGrid	Arranges content in a matrix of columns and rows where all the cells have the same size.	
Grid	Same as UniformGrid but by default, rows and columns take up the least amount of space necessary to accommodate the largest content within any cell contained in a given row or column.	
Canvas	Child elements are positioned by coordiantes. Child elements of a Canvas are never resized, they are just positioned at their designated coordinates. Canvas is the only panel element that has no inherent layout. It has default Height and Width properties of zero.	

Text Containers

TextBlock	Lightweight control for displaying small amounts of flow content.	
TextBox	Display and edit unformatted text.	
RichTextBox	A TextBox which operates on `FlowDocument` objects.	
FlowDocumentReader	Provides a control for viewing flow content, with built-in support for multiple viewing modes. Other name: `SinglePageViewer`.	

Miscellaneous Containers

Page	Content that can be navigated to and hosted by a browser. An application typically has two or more pages, which can be navigated between using a Hyperlink or NavigationService or a browser. See: Navigation Overview.	
Slider	Enables the user to select from a range of values by moving a `Thumb` control along a track. .	
ViewBox	A child is automatically stretched and scaled to fill the available client area.	
Border	Draws a border and a background around another element.