Quick recap

A reminder of what we have done during Lecture 03

Last time…

- Dependencies
  - What is required to develop Vaadin apps?
  - What is optional?
- Setting up Eclipse
  - How to install Vaadin plugin?
- Writing Vaadin applications
  - What is the main class?
  - How to capture events with listeners?
  - How to debug server-side code?
  - How to deploy the application?

Web application development with Vaadin

Lecture 04

GUI Components in Vaadin: Overview

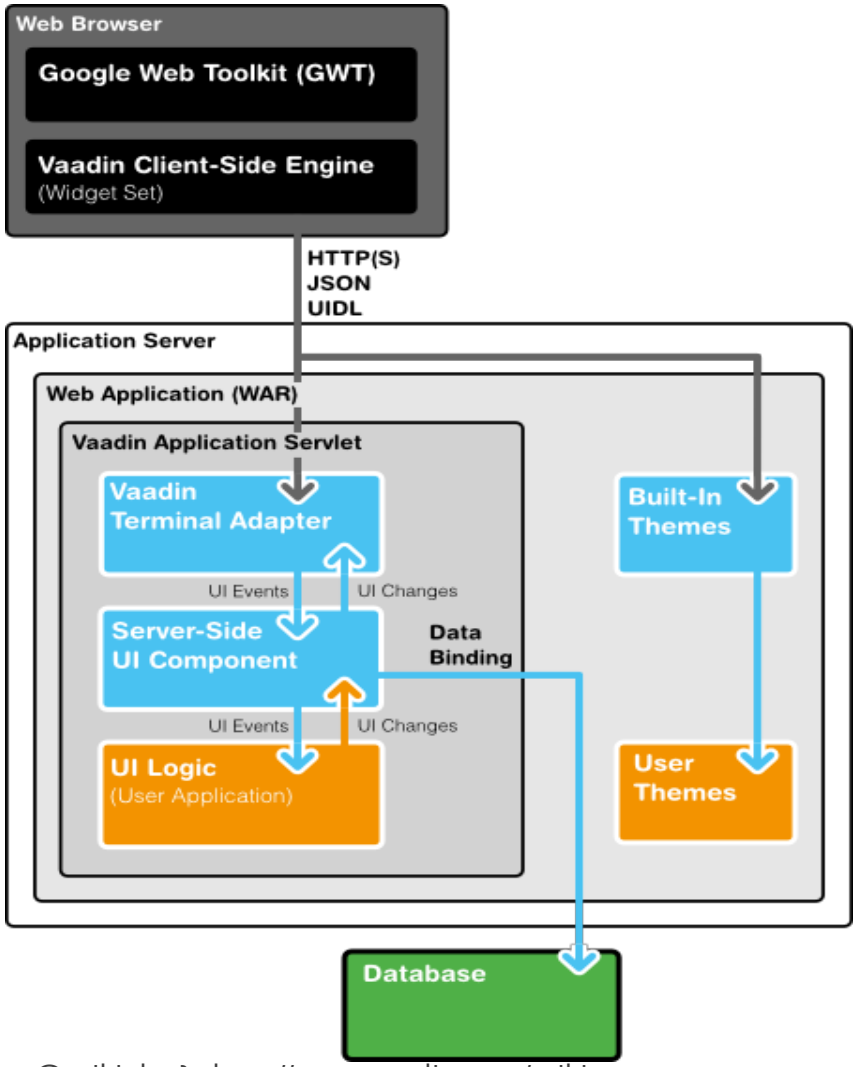
Contents

- Vaadin Architecture
- Sampler
- Common properties
- Components
- Summary

Vaadin Architecture

The bigger picture

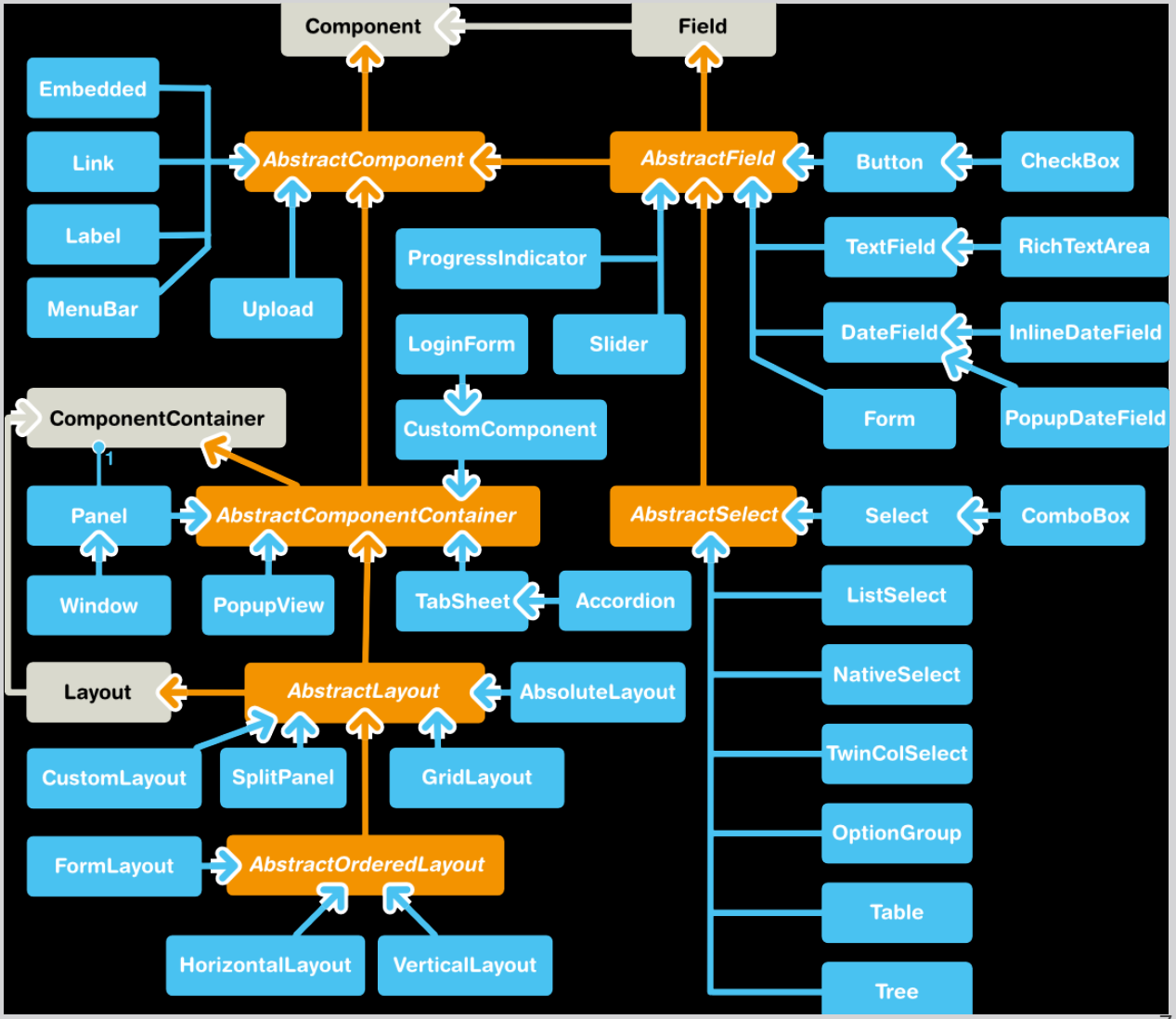
Architecture



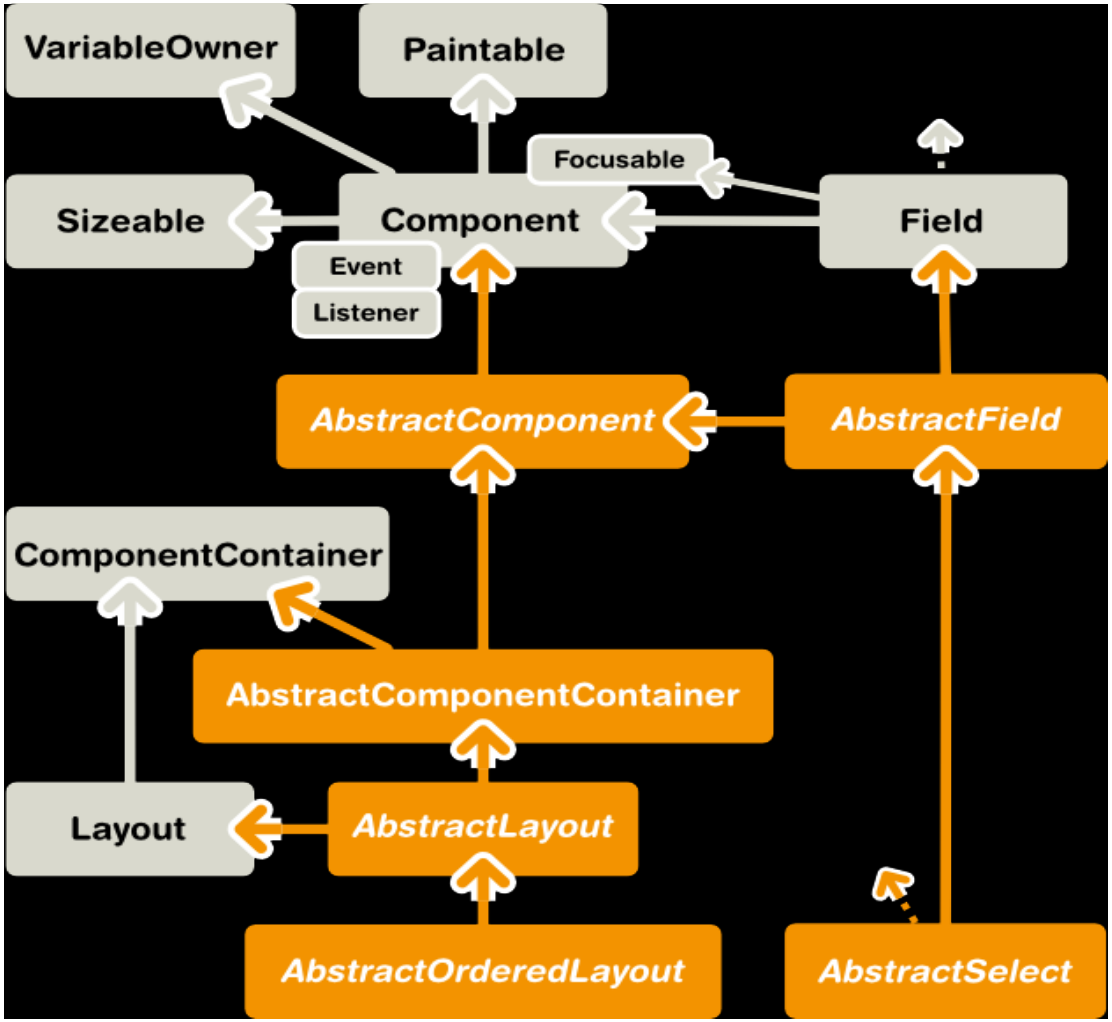
Component hierarchy

2013-01-17

- Legend
  - Gray: interface
  - Orange: abstract class
  - Blue: concrete class
  - **Note: Vaadin 6**
- Two subhierarchies
  - Components
  - Fields
- Five groups of components
  - Simple components
  - Containers
  - Layouts
  - Fields
  - Selects



Abstractions



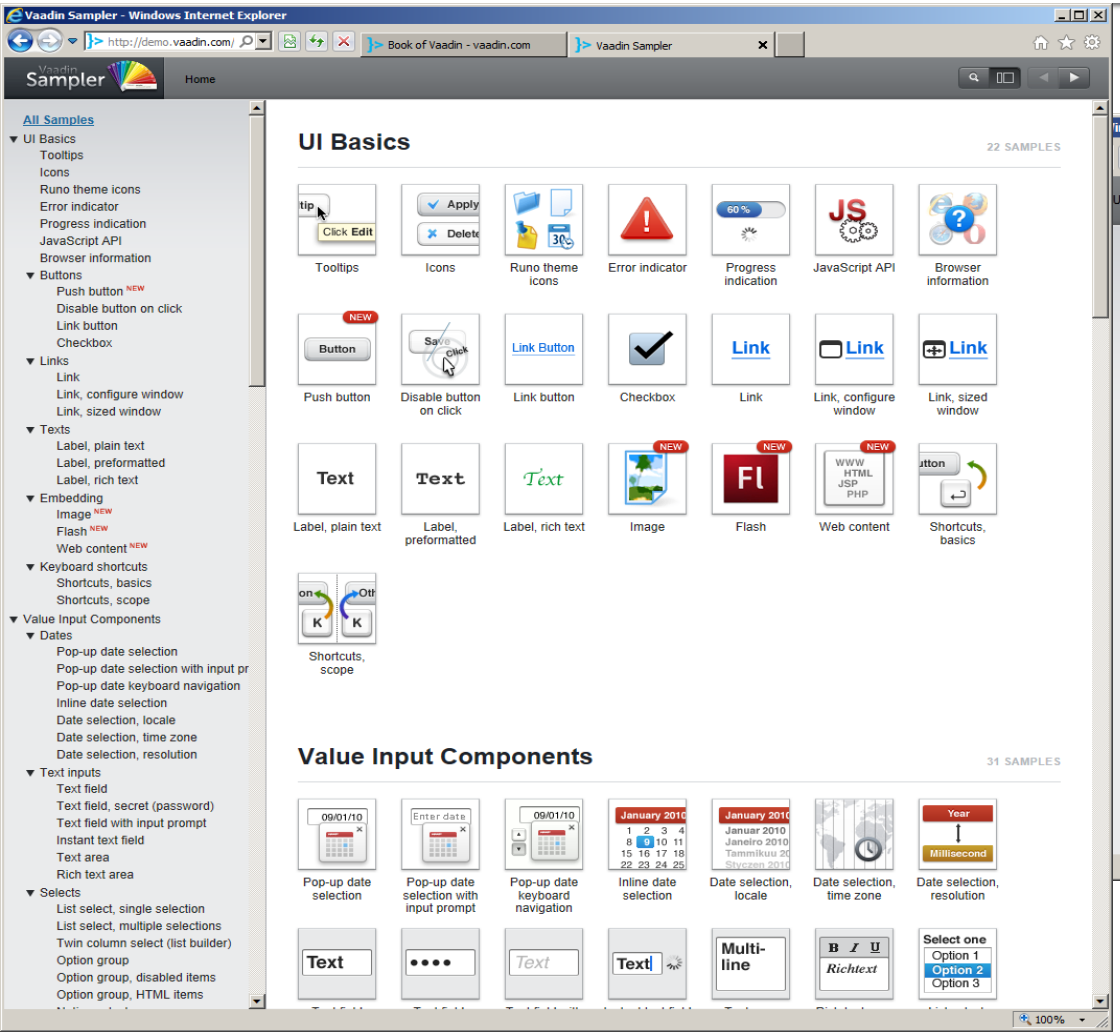
Component

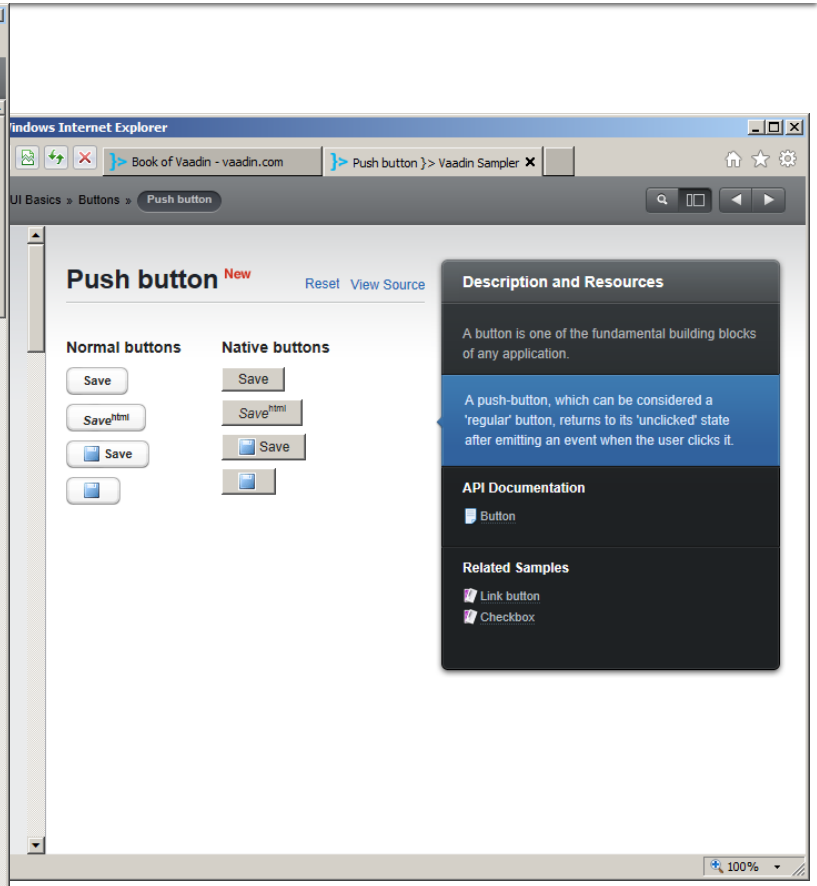
- Base interface for components
  - `Paintable`
    - Paints component at the client side
  - `VariableOwner`
    - User interaction from the client side
  - Bunch of useful properties
- Runtime hierarchy
  - Parent-children relationship
  - Dynamic
    - Not known at construction time
- *Paired* with `AbstractComponent`
  - Implements serialisation
  - And useful properties

Sampler

Try before you code

http://demo.vaadin.com/sampler





Features

- Up-to-date version
- All the components
  - Nicely organised
  - With sample source code

Common properties

Everything is a component

Property: Caption

- An explanatory text accompanying a user interface component
  - Displayed near
  - Describes role
  - Plain text only
    - HTML escaped
- Handling varies
  - Some display themselves
  - Some rely on parents

Property: Caption

SERVER SIDE

```
foo.setCaption("bar");

String bar =
  foo.getCaption();

// nothing fancy there
```

CSS RULES

- .v-caption
  - Caption element
- .v-captiontext
  - Caption itself
- .v-caption-clearelem
  - Clears floating elements

Property: Icon

- An explanatory graphic accompanying a user interface component
  - Displayed near
  - Depicts role
    - Accessibility is important
- Handling varies
  - Some display themselves
  - Some rely on parents

Property: Icon

SERVER SIDE

```
// sets icon based on
// current theme

foo.setIcon(
 new ThemeResource(
  "icons/user.png"
));

// there are several
// types of resources
// (lecture 06)

Resource icon =
  foo.getIcon();
```

CSS RULES

- .v-icon
  - May be inside .v-caption

Property: Description

- More elaborated text about the component
  - Detailed information, e.g.
    - Component role
    - Allowed values
  - Rich text
    - (X)HTML allowe
- Usually a tooltip
- Strangely, **not in the interface Component**
  - Present in `AbstractComponent`
    - All Vaadin components are `AbstractComponents`
    - Reasons for this decision: unknown
  - Made public in the interface `Field`

Property: Description

SERVER SIDE

```
foo.setDescription(
  "<h2>Hello, world!</h2>");

String bar =
   foo.getDescription();

// notice (X)HTML
// can be anything, e.g. <img>
```

CSS RULES

- No CSS rules
- Painted by the client

Property: Locale

OVERVIEW

- Inheritable
  - Parent
  - Application
  - System
- Not available in the constructor
  - No use for straightforward i18n
- No CSS rules

SERVER SIDE

```
// language ISO 639-1
// country ISO 3166-1-
//           -alpha-2

foo.setLocale(
  new Locale(
    "pl", "PL"
 ));

Locale bar =
  foo.getLocale();
```

Properties: Component flags

ENABLED

- Concerns user interaction
  - Affects all contained components
  - Disabled
    - Visible
    - Accepts no input
  - Enabled
    - Default state
- Code

```
foo.setEnabled(boolean);
boolean bar =
  foo.isEnabled();
```

- CSS rule
  - .v-disabled
    - In addition to normal CSS
    - No rule for enabled components

VISIBLE

- Concerns component visibility
  - Affects all contained components
  - Invisible
    - Not passed to the browser
    - As if never existed
  - Visible
    - Default state
- Code

```
foo.setVisible(boolean);
boolean bar =
  foo.isVisible();
```

- No CSS
  - Empty element
  - Inline style
    - `display: none;`

Properties: Component flags

READ-ONLY

- Concerns user interaction
  - Does not apply to contained components
  - Made visible by the interface `Field`
    - Makes sense, fields take values from the user
  - Read-only
    - Value modifications not communicated by the browser
    - Server does not accept any changes
      - `Property.ReadOnlyException`
  - Non read-only
    - Default state

SERVER SIDE

```
foo.setReadOnly(boolean);
boolean bar =
   foo.isReadOnly();
```

CSS RULE

- .v-readonly
  - In addition to normal CSS
  - No rule for default state

Property: Style names

- Custom CSS style class names
  - Name your things!
  - Formatting done in app's CSS
- Rendered in two forms
  - .v-{component}-{style_name}
  - .{style_name}
- Cannot cause conflicts
- Must be valid CSS class name

Property: Style names

SERVER SIDE

```
Label foo =
  new Label("foo");

// sets style name or names
foo.setStyleName(
 "style1 style2 style3"
);

// add and remove
// one-by-one
foo.addStyleName("style1");
foo.addStyleName("style2");
foo.removeStyleName("bar");

// space-separated string
String bar =
   foo.getStyleName();
```

CSS RULES

- .v-label-style1
- .v-label-style2
  - Different for other components
- .style1
- .style2
  - The same for any component

Property: Width and Height

- Defined in interface **`Sizeable`**
  - Not every component can be resized
  - Each **`Field`** is sizeable
- Can be undefined
  - The component will take as little space as possible
- Default behaviour varies

Property: Width and Height

SERVER SIDE

```
Button foo =
  new Button("foo");

// sets width with CSS string
foo.setWidth("100px");

// set height another way
foo.setHeight(
      50, Sizeable.Unit.PERCENTAGE);

// sets height undefined
foo.setHeight(Sizeable.SIZE_UNDEFINED);

// shortcuts for setting both dimensions
foo.setSizeUndefined();
foo.setSizeFull();

// getters
float width = foo.getWidth();
Sizeable.Unit widthUnit =
    foo.getWidthUnit();

// similar for height, of course
```

CSS RULES

- No CSS rules
- Rendered as inline style
  - Cannot be overriden
  - Unless undefined

Property: Focusable and TabIndex

- Defined in **`Component.Focusable`**
  - Not every component can be focused
  - Each **`Field`** is focusable
- No way to find currently focused component
  - Some (but not all) fields broadcast
    - **`FocusEvent`** when receiving focus
    - **`BlurEvent`** when losing focus
- Focus order of components can be set

Property: Focusable and TabIndex

SERVER SIDE

```
TextField foo =
  new TextField("foo");

// server-side focusing
foo.focus();

// tab-order
foo.setTabIndex(1);
int index =
  foo.getTabIndex();
```

CSS RULES

- .v-{component}-focus
  - In addition to normal CSS
  - No rule for no focus

Components

The simplest components

com.vaadin.ui.Label

ContentMode.TEXT

- Default mode
- Plain text
- < > & are escaped

ContentMode.(X)HTML

- Rendered as
- Should be XHTML 1.1 Strict
- Can lead to cross-site scripting

ContentMode.PREFORMATTED

- Rendered with monospaced font
- Can contain \n \t
- < > & are escaped

com.vaadin.ui.Label

SERVER SIDE

```
Label foo =
  new Label("foo");

foo.setCaption(
  "A link to <a href=
  \"http://vaadin.com\">
  Vaadin home page
  </a> in a label");

// switching mode
foo.setContentMode(
  ContentMode.XHTML);
```

CSS RULES

- .v-label

com.vaadin.ui.Link

- Clickable link to an external resource
  - Target resource can be changed
- Can have an icon
- Does not broadcast events
- There are other ways
  - **`Label`** in **`ContentMode.XHTML`**
  - **`Button`** styled as link
    - Broadcasts events

com.vaadin.ui.Link

SERVER SIDE

```
// constructor is simple
Link link =
 new Link(
  "Vaadin forum",
  new ExternalResource(
   "http://vaadin.com/forum"
));

// there are also getters
link.setIcon(
  new ThemeResource(
    "img/user.png„
));

// resource is not fixed
link.setResource(...);
```

CSS RULE

- .v-link
  - a
  - .v-icon
    - span

com.vaadin.ui.MenuBar

- Horizonal menu bar
  - With submenus and submenus…
  - **`.addItem(String, Resource, MenuBar.Command)`**

    → **`MenuBar.MenuItem`**
    - Item text
    - Item icon (optional)
    - Command executed upon clicking (optional)
    - Returned menu item can contain more items
- Just like in desktop apps
  - But positioned anywhere

com.vaadin.ui.MenuBar

SERVER SIDE

```
final Label clicked =
  new Label("Nothing clicked.");
MenuBar.Command cmd =
  new MenuBar.Command() {
   public void menuSelected(
                          MenuItem item) {
     clicked.setCaption(
             "Clicked: "+item.getText());
}};

MenuBar menu = new MenuBar();
for(String main:
           new String[]{"foo", "bar"}) {
  MenuBar.MenuItem sub =
    menu.addItem(main, null, cmd);
  sub.addItem(main+"_sub", null, cmd);
  sub.addItem(main+"_omg", null, null);
}

//will produce menu:
// foo       | bar
//  -foo_sub |  -bar_sub
//  -foo_omg |  -bar_omg
```

CSS RULES

- .v-menubar
  - .gwt-MenuItem
    - Menu items
  - .gwt-MenuItem-selected
    - Selected menu items

com.vaadin.ui.Embedded

PURPOSE

- Embeds a resource
  - PDF, applet, stream…
- Three common cases excluded
  - All inherit from **`AbstractEmbedded`**
    - **`Embedded`** does not

com.vaadin.ui.Image

- Embeds an image
- Broadcasts click events

com.vaadin.ui.Flash

- Embeds Flash content
  - For those that support it

com.vaadin.ui.BrowserFrame

- Embeds web page
- Rendered as <iframe>

Basic fields

Components that accept a simple value

Overview

- Interaction with the user
  - Broadcast events that can be listened to
    - **`ClickEvent`**
    - **`FocusEvent`**
    - **`BlurEvent`**
    - **`ValueChangeEvent`**
    - **`ReadOnlyStatusChangeEvent`**
  - Contain user-assignable value
- Designed to fit Vaadin Data Model
  - Explained later in the course
  - Can be used without a data source

com.vaadin.ui.Button

- Clickable thingy
  - Finalises user input
  - Initialises an action
  - Can be automatically disabled on click
  - Can contain HTML caption
- Clickable in a number of ways
  - Mouse
  - Keyboard shortcut
  - Method call

com.vaadin.ui.Button

SERVER SIDE

```
// straightforward way to
// create a button with a
// listener in one go
final Button btn =
  new Button(
    "Click me!",
      new Button.ClickListener() {
        public void buttonClick(
          Button.ClickEvent event) {
            btn.setCaption(
              "Can't touch this!");
  }});

// disable the button on click
btn.setDisableOnClick(true);

// run this
// try clicking the button
```

CSS RULES

- .v-button
  - Contents may vary

com.vaadin.ui.CheckBox

- Two-state selection component
  - Selected
  - Deselected
  - None of the above
- Two distinct roles
  - Confirs something when alone
  - Selects options when in group

com.vaadin.ui.CheckBox

SERVER SIDE

```
final Label foo = "checked";

// initial state can be specified
CheckBox box = new CheckBox(
                "Select me!", true);

box.addValueChangeListener(
 new Property.ValueChangeListener(){

  public void valueChange(
   Property.ValueChangeEvent event){

    foo.setCaption(
      ((Boolean)event.getProperty()
                          .getValue())
      .booleanValue()
        ? "checked„
        : "unchecked");
}});
```

CSS RULES

- .v-checkbox
  - input
    - For the clickable box
  - label
    - For the text

com.vaadin.ui.AbstractTextField

PURPOSE

- Common class for text inputs
  - Can restrict length
  - May have an input prompt
    - Aside from caption, description and icon
- Holds **`String`** as a value
- Broadcasts also **`TextChangeEvents`**

com.vaadin.ui.TextField

- Typical single-line text input field
- CSS rule
  - .v-textfield

com.vaadin.ui.TextArea

- Supports multi-line text
- Can wrap lines
- CSS rule
  - .v-textarea

com.vaadin.ui.PasswordField

- Single-line text input field with hidden text
- Does not encrypt its contents
- CSS rule
  - .v-textfield

com.vaadin.ui.RichTextArea

OVERVIEW

- WYSIWYG editor
- Stores **`String`** with HTML
  - Max length includes formatting
  - Possible cross-site scripting
- Extendable on client side
- No **`TextChangeEvent`**s

CSS RULES

- .v-richtextarea
  - .gwt-RichTextToolbar
    - Toolbar with options
  - .gwt-RichTextArea
    - The editor

com.vaadin.ui.DateField

- Holds **`java.util.Date`** as value
  - Which means also time
- Two available subclasses
  - **`InlineDateField`**
    - Displays inline editor
  - **`PopupDateField`**
    - Displays input with a button
    - Button shows date picker in a popup
- Many formats
  - Resolution = level of details
    - Year, month, day, hour, minute, second, milisecond
  - Validates input in the browser and on the server
  - Custom error message
  - Can have a *relaxed* policy
    - May 0th = April 30th

com.vaadin.ui.DateField

CSS RULES FOR INLINE

- .v-datefield
  - .v-datefield-calendarpanel
    - .v-datefield-calendarpanel-header
      - .v-datefield-calendarpanel-prevyear
      - .v-datefield-calendarpanel-prevmonth
      - .v-datefield-calendarpanel-month
      - .v-datefield-calendarpanel-nextmonth
      - .v-datefield-calendarpanel-nextyear
    - .v-datefield-calendarpanel-body
      - .v-datefield-calendarpanel-weekdays
      - .v-datefield-calendarpanel-weeknumbers
        - .v-first
        - .v-last
      - .v-datefield-calendarpanel-weeknumber
      - .v-datefield-calendarpanel-day
    - .v-datefield-calendarpanel-time
      - .v-datefield-time
        - .v-select
        - .v-label

CSS RULES FOR POPUP

- .v-datefield
  - .v-datefield-popupcalendar
    - The calendar
  - v-datefield-textfield
    - Input field
  - .v-datefield-button
    - Button that opens popup
  - .v-datefield-{resolution}
    - full, day, month, year
    - Depends on field resolution
  - .v-datefield-popup
    - .v-popupcontent
    - .v-datefield-calendarpanel {}
      - Once the popup is open

Component containers

Components that contain other components, but are not layouts

com.vaadin.ui.Panel

- Useful things
  - Caption
  - Icon
  - Border
  - Scrollbar
    - Scrollable programmatically
  - Broadcasts clicks
    - Unless a component inside ate it
- Can hold exactly one component
  - Though that component can contain other components which can contain components
    - We have to go deeper
- Adds complexity to layout
  - Abuse heavily increases page rendering time

com.vaadin.ui.Panel

SERVER SIDE

```
Image orly =
 new Image(
  "Ya rly!",
  new ExternalResource(
    "http://bit.ly/9o3Qd2")
  );

Panel panel =
  new Panel("No wai!");
panel.setContent(orly);

// if the image is too big
// scrollbars will appear
// srsly
```

CSS RULES

- .v-panel
  - .v-panel-caption
  - .v-panel-nocaption
    - The caption part
  - .v-panel-content
    - What is inside the panel
  - .v-panel-deco
    - Borders and shadows

com.vaadin.ui.AbstractSplitPanel

- Holds two components
  - Which can contain components which… etc.
    - Might be **`Panel`**s to magically allow scrollbars
- Two subclasses
  - **`HorizontalSplitPanel`**
    - Components arranged horizontally
    - The split is a vertical line
  - **`VerticalSplitPanel`**
    - The other way
- Split can be locked
  - User cannot change the position of the split

com.vaadin.ui.AbstractSplitPanel

SERVER SIDE

```
HorizontalSplitPanel ltr =
  new HorizontalSplitPanel();
VerticalSplitPanel ttb =
  new VerticalSplitPanel();

final TextField field =
  new TextField("Type something:");
final Label label =
  new Label("Nothing typed yet.");

Button btn = new Button(
  "Click me!",
  new Button.ClickListener() {
    public void buttonClick(
        Button.ClickEvent event) {
     label.setCaption(
            field.getValue());
}});

ttb.setFirstComponent(btn);
ttb.setSecondComponent(label);
ltr.setFirstComponent(field);
ltr.setSecondComponent(ttb);

// Type something:  |   [Click me!]
// [_______]        |-----------------
//                  | Nothing typed yet.
```

CSS RULES

- .v-splitpanel-horizontal
  - .v-splitpanel-hsplitter
  - .v-splitpanel-hsplitter-locked
    - For horizontal split panel
- .v-splitpanel-vertical
  - .v-splitpanel-vsplitter
  - .v-splitpanel-vsplitter-locked
    - For vertical split panel
  - .v-splitpanel-first-container
    - Top or left component
  - .v-splitpanel-second-container
    - Bottom or right component

com.vaadin.ui.TabSheet

- Multicomponent container
  - Each component on a separate tab
    - Caption
    - Icon
    - Description
    - Visibility
    - Availability
    - Ability to be closed
- Tabs are (currently) not loaded until selected
- Tab headers are arranged horizontally
  - Tab is displayed below tab headers
- One subclass, **`Accordion`**
  - Tab headers are arranged vertical
    - Tab is displayed between tab headers
- Broadcasts event when
  - Selected tab changes
    - Or when the first tab is added (it gets selected)
  - A tab is closed

com.vaadin.ui.TabSheet

SERVER SIDE

```
Accordion acc =
  new Accordion();
for(int count=1;
    count<=10;
    count++)
  acc.addTab(
    new Button(
      "Button" + count
    )).setCaption(
      "Page "+count
  ));
```

CSS RULES

- .v-tabsheet
  - .v-tabsheet-tabs
  - .v-tabsheet-content
    - There is more
- .v-accordion
  - .v-accordion-item
  - .v-accordion-item-caption
  - .v-accordion-item-content
    - There is more

Advanced fields

Components that allow to select option(s) from a data source

Overview

- Advanced interaction with the user
  - Broadcast events that can be listened to
    - **`ClickEvent`**
    - **`FocusEvent`**
    - **`BlurEvent`**
    - **`ValueChangeEvent`**
    - **`ReadOnlyStatusChangeEvent`**
    - **`ItemSetChangeEvent`**
    - **`PropertySetChangeEvent`**
- Contain user-assignable value
  - Selectable from a range of options
  - Fully configurable display
- Designed to fit Vaadin Data Model
  - Explained later in the course
  - Most have a meaningful behaviour without it

com.vaadin.ui.ComboBox

- Selects a single value
- Allows creating new values
- Dynamic item suggestion
  - As the user types in

com.vaadin.ui.NativeSelect

- Selects a single value
- As simple as it can be
  - No new values
  - No dynamic item suggestion
- CSS rules
  - .v-select
    - For the component
  - .v-select-select
    - For the <select>

com.vaadin.ui.ListSelect

- Can select multiple values
- As simple as it can be
  - No new values
  - No dynamic item suggestion
- CSS rules
  - .v-select
    - For the component
  - .v-select-select
    - For the <select>

com.vaadin.ui.OptionGroup

- Can select multiple values
  - Group of radio buttons in single-selection mode
  - Group of check boxes in multi-selection mode
- Supports disabling individual items
- CSS rules
  - .v-select-optiongroup
    - .v-select-option
      - For each option
    - .v-checkbox
      - In addition, for check boxes
    - .v-radiobutton
      - In addition, for radio buttons
    - .v-disabled
      - In addition, for disabled items

com.vaadin.ui.TwinColSelect

- Multiple selection based on two columns
  - *Options* and *selections*
- Some customisation
  - Captions for each column
  - Number of rows visible
- CSS rules
  - .v-select-twincol
    - .v-select-twincol-options-caption
    - .v-select-twincol-selections-caption
    - .v-select-twincol-options
    - .v-select-twincol-buttons
      - .v-button
        - .v-button-wrap
          - .v-button-caption
      - .v-select-twincol-deco
    - .v-select-twincol-selections

Demo!

- Different data selecting components
- Live coding
  - Please forgive errors
  - Do not snore

Other components

Few words about

Data presentation components

com.vaadin.ui.Table

- Tabular representation
- Lots of options
  - Headers, footers
  - Multiselect or no selection at all
  - Hidden columns
  - Moving columns
  - Sorting
  - In-place editing
  - Cell generators
  - Drag and drop
- Lots of code
- Subclassed by **`TreeTable`**

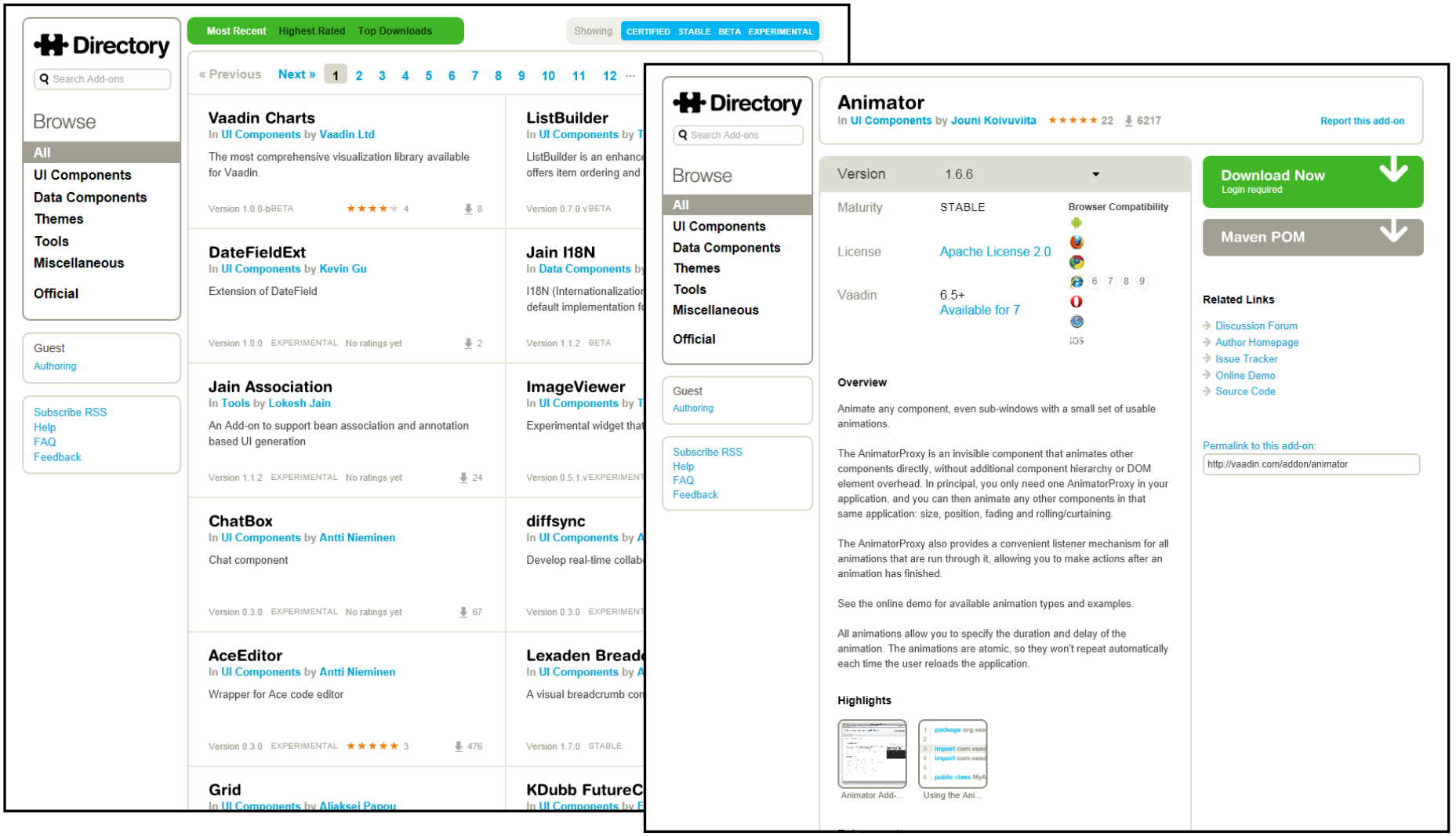
com.vaadin.ui.Tree

- Tree representation
  - Hierarchy
- Options
  - Multiselect
  - Drag and drop
  - Node collapsing
    - With custom icons

More core components

- **com.vaadin.ui.Upload**
  - For handling uploads from the browser
  - Sends the upload to a stream
- **com.vaadin.ui.ProgressIndicator**
  - Displays progress as a value from 0.0 to 1.0
  - Works on client side by polling the server
- **com.vaadin.ui.Slider**
  - For selecting numerical value within a range
  - Can be horizontal or vertical

http://www.vaadin.com/directory



Summary

What did we do today?

Lessons of today (hopefully)

- Component hierarchy
  - What are the basic interfaces?
  - What are the differences?
- Common properties
  - What are they?
  - Where are they defined?
- Components
  - How are they grouped?
  - What events do they broadcast and when?

Coming up next

- Basic principles of UI/UX design
  - Rolf Smeds, Developer and UX Designer
- Layouts, themes, styles and navigation
- Assignments

The end

Suggestions? Questions?

miki@vaadin.com
t: @mikiolsz