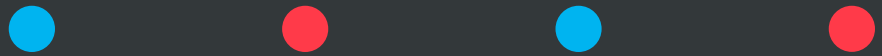
QUICK RECAP

A reminder of what we have done during Lecture 04



vaadin}>

LAST TIME...

- Layouts
  - How do they differ?
- Themes
  - How to use a theme?
  - How to create a theme?
- Navigation
  - What are views?
  - How to navigate between them?

EXTENDING THE VAADIN FRAMEWORK: SERVER-SIDE COMPONENTS

Development of Modern Web Applications (with Vaadin)

Lecture 05

vaadin

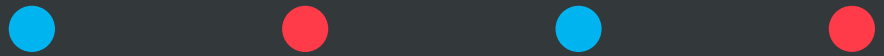
OVERVIEW

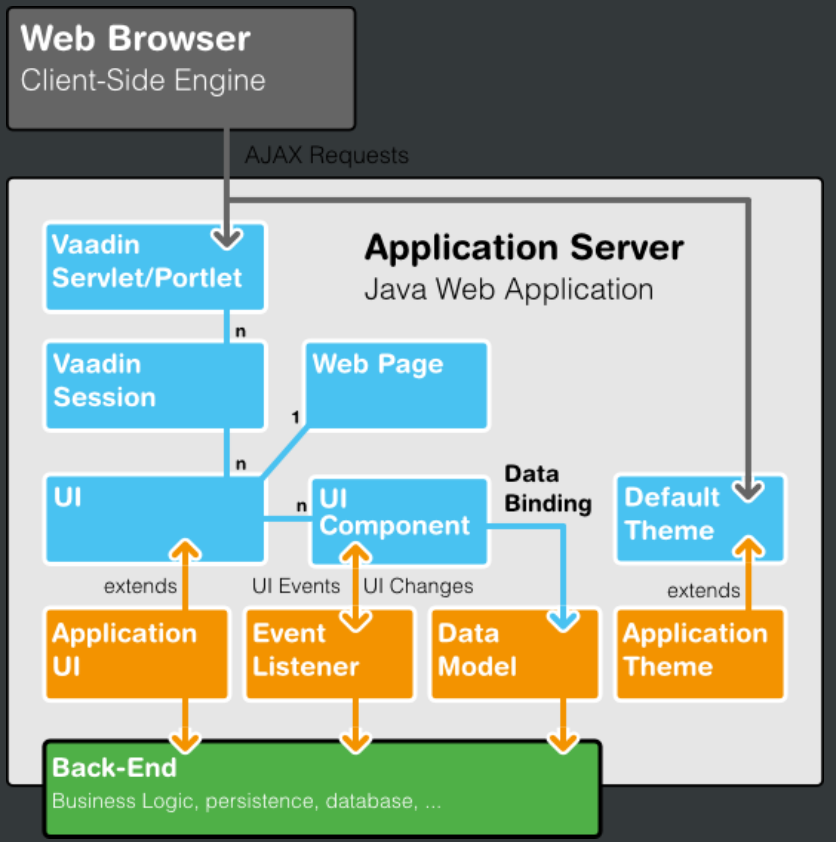
- Server-side architecture
- Resources
- Errors
- Extending components

- Shoutbox app continues

SERVER-SIDE ARCHITECTURE

Behind the scenes





OVERVIEW

Follows the three-layer approach –
this diagram show only the application layer
(i.e. the server side of a Vaadin app).

COM.VAADIN.UI.UI

Theory

A viewport
to an
application
running
in a web page

Practice

An HTML fragment
in which
an application runs
in a web page

COM.VAADIN.UI.UI

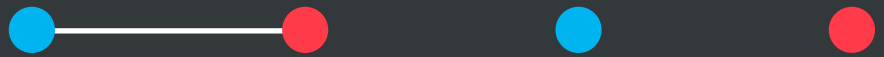
- Application must have at least one
  - In most cases, exactly one
- Always belongs to a page
  - `UI.getCurrent().getPage();`
- A page can have many UIs
  - In typical cases, exactly one
    - More, if you deploy to a portal
      - Different applications running on the same web page
- Connected to user session of an application

COM.VAADIN.SERVER. VAADINSESSION

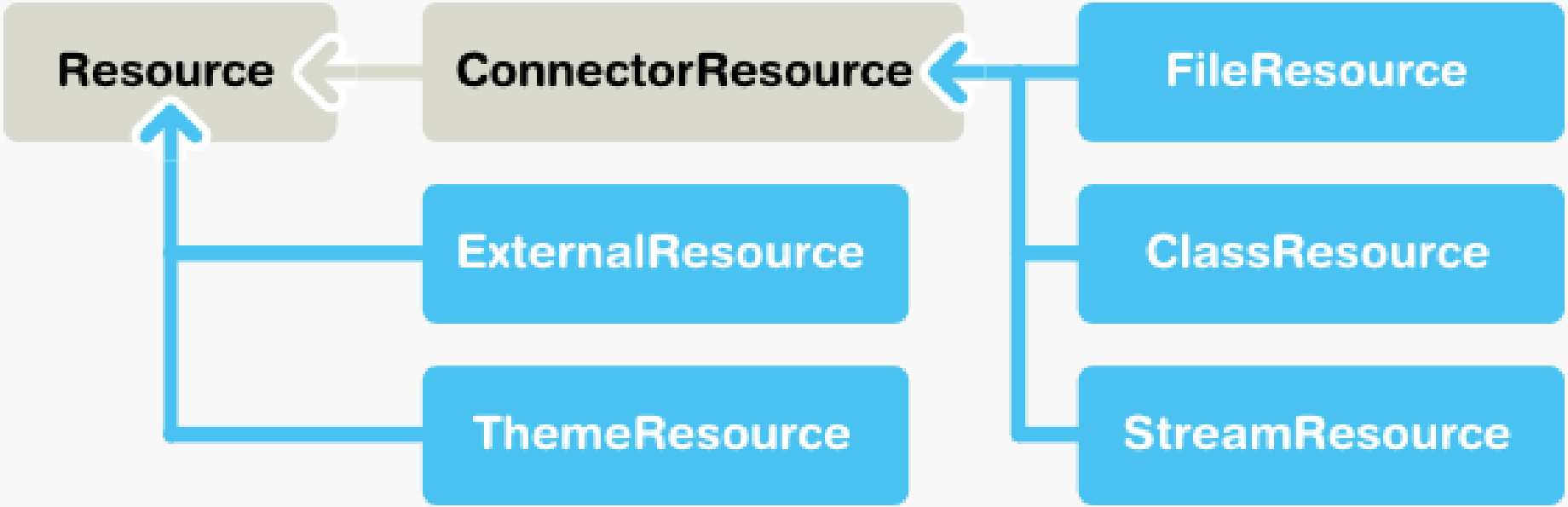
- Contains one or more currently open UIs
  - `VaadinSession.getCurrent().getUIs();`
- Finite validity
  - Starts when the user first opens the app
  - Ends on
    - Server-decided timeout
    - Explicit closing of the application
- Application storage
  - `getAttribute(Class<T>) → T;`
  - `getAttribute(String) → Object;`
  - `setAttribute(Class<T>, T);`
  - `setAttribute(String, Object);`

RESOURCES

Referencing external content



OVERVIEW


Resource
ConnectorResource
FileResource
ExternalResource
ClassResource
ThemeResource
StreamResource

COM.VAADIN.SERVER.
FILERESOURCE

- Files stored in the filesystem **of the server**
  - Cannot be referred by URI
  - Needs to be accessed by the servlet
- Defined with `java.io.File`
  - `new FileResource(File);`
  - Relative path, server-dependent
  - Absolute path
    - `VaadinService.getCurrent().`
      `getBaseDirectory().getAbsolutePath();`

COM.VAADIN.SERVER. CLASSRESOURCE

- Accessible with Java Class Loader
- Base directory `{appDir}/WEB-INF/classes`
  - `new ClassResource(String);`
  - `new ClassResource(Class<?>, String);`

COM.VAADIN.SERVER.
THEMERESOURCE

- Resource from a theme
  - Base directory `{appDir}/VAADIN/themes/{theme}`
- `new ThemeResource(String);`

COM.VAADIN.SERVER.
STREAMRESOURCE

- Anything
  - Readable with `InputStream`
- Dynamic content
  - Generated as a result of user actions
  - Requires file name
    - For storing on the client side

```
new StreamResource(StreamSource, String);
public interface StreamResource.StreamSource {
   public InputStream getStream();
}
```

COM.VAADIN.SERVER.EXTERNALRESOURCE

- Well, external
  - Meaning: not necessarily at the application server
- Must be available through a URL
- Optional content type
  - Browser auto-detection if not provided
  - MIME type
    - Multipurpose Internet Mail Extensions

```
new ExternalResource(String);
new ExternalResource(String, String);
new ExternalResource(URL);
new ExternalResource(URL, String);
```

REFERENCING RESOURCES

com.vaadin.ui.Embedded

- Embeds the resource
  - Part of the page
  - Loaded with the page

com.vaadin.ui.Link

- Links to the resource
  - Makes new request

Component resources

- Icon is a resource
  - There may be other resources, depending on a component

ERRORS

Handling and reacting

OVERVIEW

- Every component can display errors
  - Error indicator displayed
  - Customisable error message
- Fields can be validated
  - Error when value is invalid

```
setComponentError(ErrorMessage);
getComponentError() → ErrorMessage;
```

COM.VAADIN.SERVER.ERRORMESSAGE

- Properties
  - HTML-formatted message
  - Error level
    - Five defined

COM.VAADIN.SERVER.ERRORMESSAGE

com.vaadin.terminal.UserError

- Controlled error
- Guidance to the user

com.vaadin.terminal.CompositeErrorMessage

- Combines many error messages

com.vaadin.terminal.SystemError

- Runtime exception in the system
- Not planned

Other error messages

- Part of validation
- Component-specific

COM.VAADIN.SERVER.
ERRORHANDLER

- Component-specific
  - Displays tooltip with message
  - Turns on error indicator
- UI-specific
- Application-specific
  - Logs to the console
- Server-specific
  - Not part of Vaadin

COM.VAADIN.SERVER. SYSTEMMESSAGESPROVIDER

Overview

- Indication of a severe invalid state
  - Requires restart of the app
- Message properties
  - Caption (error title)
  - Message (brief description)
  - URL (to redirect to)
  - Notification flag

```
VaadinService.
getCurrent().
setSystemMessagesProvider(...);
```

System messages

- sessionExpired
  - No activity for some time
    - Timeout configurable per app
- communicationErrorURL
  - Problem with connection between client and server
- authenticationError
  - 401 response code
- internalError
  - Serious problem with the framework or a component
- outOfSync
  - Client and server data are not synchronised properly
- cookiesDisabled
  - Cookies are disabled

EXTENDING COMPONENTS

Ways of adding new functionality

EXTENDING EXISTING COMPONENTS

- Keeps the old functionality
  - Unless overridden
- Type-safe
  - Can be used as old component
- Reduces work
  - Most of it is already done
  - Just add new stuff
- Suitable for a single component
  - Extending layouts or panels makes little sense

COM.VAADIN.UI. CUSTOMCOMPONENT

- Base class for composite components
  - Reusable groups of components
- Hides the underlying layout
  - `protected void setCompositionRoot(...);`
    - Takes any component as a parameter
      - Often a layout or other component container
    - Can be called only once
    - Must be called before the first use
      - Typically in the constructor
  - Size of layout is NOT size of the component
    - Unless you set layout's size to full
- No events by default

COM.VAADIN.UI.CUSTOMFIELD

- Base class for composite fields
  - Or fields of an unsupported type
  - `public Class<? extends TYPE> getType();`
    - Well, what the name says
- Hides the underlying layout
  - `protected Component initContent();`
    - Sets everything up
    - Called automatically
    - Usually returns a layout
- Maintains value and fires events

COM.VAADIN.UI.
CUSTOMLAYOUT

- Slightly confusing
- **Not** a base class for custom layouts
  - A layout that reads a template from XHTML
    - This was on the previous lecture ☺

BUNCH OF ABSTRACT CLASSES

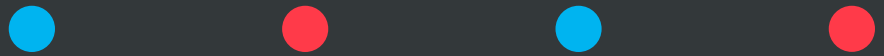
- `AbstractComponent`
- `AbstractField`
- `AbstractSelect`
- `AbstractTextField`
- `AbstractComponentContainer`
- `AbstractContainer`
  - …and more

- Use as an entry point for subclassing
  - Reuse what has already been done
  - Implement the missing details

DEMO!

Shoutbox step 8
http://github.com/vaadin-miki/shoutbox

end branch: `step-08`

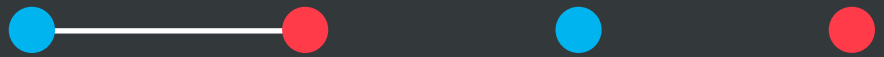


THE PLAN

- Simplify
  - Listen to container events in the layout itself
    - `Container.Viewer`
  - Make the view a subclass of that

SUMMARY

What did we do today



LESSONS OF TODAY (HOPEFULLY)

- The details of Vaadin
  - How to handle errors?
  - What are resources?
  - What is UI?
- Extending server-side components
  - What should be extended and when?

COMING UP NEXT

- Extending Vaadin, part 2
- Best practices / Mobile First

THE END

SUGGESTIONS?
QUESTIONS?

miki@vaadin.com
t: @mikiolsz

vaadin}>