Nasdanika HTML

Fluent Java API for building Web UI

Overview

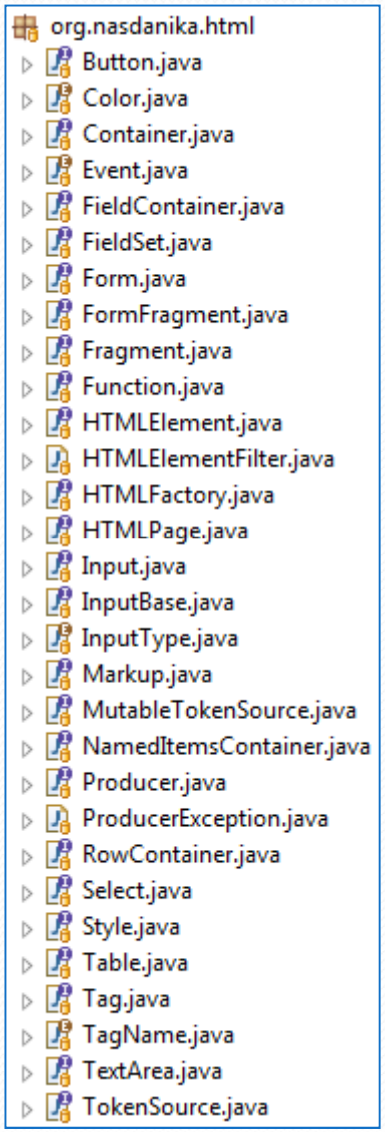
- Fluent Java API for building:
  - Low level – HTML elements
  - Mid level:
    - Bootstrap 4.x UI elements and 21 Bootswatch themes
    - Font Awesome 5.x icons
    - jsTree 3.3.7 nodes and context menus
    - KnockoutJS 3.4.x
  - High level – HTML Applications
    - Using abstractions of actions and property sources
    - From EMF models data and meta-data
- Dual delivery:
  - OSGi bundles – p2 repository
  - Jars - Maven repository (excluding EMF)

HTML

- API for building HTML elements
- Foundation for the other modules
- How to use:
  - Obtain HTMLFactory
  - Create HTML elements
  - Configure the elements:
    - Attributes
    - CSS Classes
    - Styles
- Output with `toString()` or `produce()`

```
HTMLFactory htmlFactory = HTMLFactory.INSTANCE;
Tag div = htmlFactory.div("Hello")
        .style().border("solid 1px")
        .on(Event.click, getClass().getResource("ClickHandler.js"));
System.out.println(div);
```

```
<div onclick="alert('Hi!')" style="border:solid 1px">Hello</div>
```

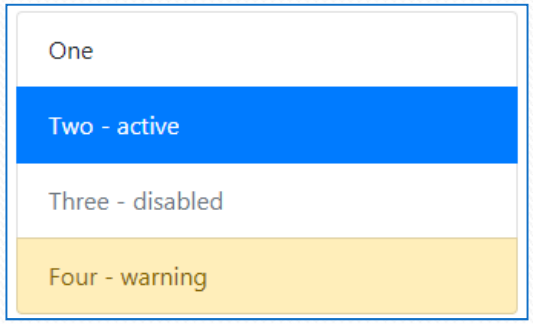


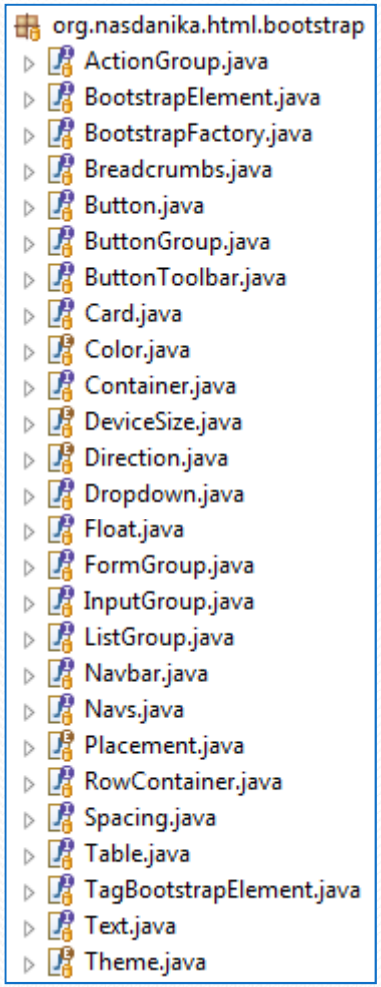
Bootstrap

- API for building Bootstrap 4 elements
- Built on HTML
- How to use:
  - Obtain BootstrapFactory
  - Create elements
  - Configure the elements
- Output with `toString()` or `produce()`
- Fallback to HTML API's when needed

```
ListGroup listGroup = BootstrapFactory.INSTANCE.listGroup(false);

listGroup.item(false, false, Color.DEFAULT, "One");
listGroup.item(true, false, Color.DEFAULT, "Two - active");
listGroup.item(false, true, Color.DEFAULT, "Three - disabled");
listGroup.item(false, false, Color.WARNING, "Four - warning");
```





Font Awesome

- API for building Font Awesome 5 icons
- Built on HTML
- How to use:
  - Obtain FontAwesomeFactory
  - Create icons
  - Configure the icons
- Output with `toString()` or `produce()`
- Fallback to HTML API's when needed

```
org.nasdanika.html.fontawesome
  FontAwesomeFactory.java
    FontAwesomeFactory
      INSTANCE
      cdn(P) <P extends HTMLPage> : P
      from(String, Style, T) <T extends HTMLElement<?>> : Icon<T>
      getHTMLFactory() : HTMLFactory
      icon(String, Style) : Icon<Tag>
      stack() : Stack
  Icon.java
    Icon<T extends HTMLElement<?>>
      Flip
      Rotate
      Size
      Stack
      Style
      fixedWidth() : Icon<T>
      flip(Flip) : Icon<T>
      li() : Icon<T>
      pullLeft() : Icon<T>
      pullRight() : Icon<T>
      rotate(Rotate) : Icon<T>
      size(Size) : Icon<T>
      spin() : Icon<T>
      toHTMLElement() : T
      ul() : Icon<T>
```

```
Icon<Tag> icon = FontAwesomeFactory.INSTANCE.icon("university", Style.SOLID)
        .size(Size.x5)
        .rotate(Rotate.R180);
```

jsTree

- API for building jsTree nodes and context menus
- Built on HTML
- How to use:
  - Obtain JsTreeFactory
  - Create nodes and menus
  - Configure the nodes and menus
- Output to JSON

```
JsTreeFactory jsTreeFactory = JsTreeFactory.INSTANCE;
JsTreeNode rootNode = jsTreeFactory.jsTreeNode();
rootNode.icon("far fa-user");
rootNode.text("User");
rootNode.id(htmlFactory.nextId());
rootNode.hasChildren();
JSONArray jsTreeRootNodes = new JSONArray();
jsTreeRootNodes.put(rootNode.toJSON());
```

```
org.nasdanika.html.jstree
  JsTreeContextMenuItem.java
    JsTreeContextMenuItem
      action(Object) : JsTreeContextMenuItem
      addSubMenuItem(String, JsTreeContextMenuItem) : JsTreeContextMenuItem
      createSubMenuItem(String) : JsTreeContextMenuItem
      disabled() : JsTreeContextMenuItem
      disabled(boolean) : JsTreeContextMenuItem
      icon(Object) : JsTreeContextMenuItem
      label(Object) : JsTreeContextMenuItem
      separatorAfter() : JsTreeContextMenuItem
      separatorAfter(boolean) : JsTreeContextMenuItem
      separatorBefore() : JsTreeContextMenuItem
      separatorBefore(boolean) : JsTreeContextMenuItem
      shortcut(Object) : JsTreeContextMenuItem
      shortcutLabel(Object) : JsTreeContextMenuItem
      subMenu(Object) : JsTreeContextMenuItem
      title(Object) : JsTreeContextMenuItem
      toJSON() : JSONObject
  JsTreeFactory.java
    JsTreeFactory
      INSTANCE
      bind(HTMLElement<?>, Object) : Tag
      bind(String, Object) : Tag
      buildAjaxJsTree(String, String) : String
      buildJsTree(Iterable<JsTreeNode>) : JSONObject
      buildJsTree(JsTreeNode...) : JSONObject
      cdn(P) <P extends HTMLPage> : P
      getHTMLFactory() : HTMLFactory
      jsTreeContextMenuItem() : JsTreeContextMenuItem
      jsTreeNode() : JsTreeNode
  JsTreeNode.java
    JsTreeNode
      Collector<R>
      accept(Collector<R>) <R> : R
      anchorAttribute(String, Object) : JsTreeNode
      children() : List<JsTreeNode>
      createChild() : JsTreeNode
      disabled() : JsTreeNode
      disabled(boolean) : JsTreeNode
      getData() : Object
      getData(String) : Object
      getId() : Object
      hasChildren() : JsTreeNode
      icon(String) : JsTreeNode
      id(Object) : JsTreeNode
      listItemAttribute(String, Object) : JsTreeNode
      opened() : JsTreeNode
      opened(boolean) : JsTreeNode
      selected() : JsTreeNode
      selected(boolean) : JsTreeNode
      setData(Object) : JsTreeNode
      setData(String, Object) : JsTreeNode
      text(Object) : JsTreeNode
      toJSON() : JSONObject
      toJSON(Predicate<JsTreeNode>) : JSONObject
```

KnockoutJS

- API for building KnockoutJS bindings
- Built on HTML

```
KnockoutBindingsSource.java
  KnockoutBindingsSource
    Binding
    getAllBindings() : Collection<Binding>
KnockoutControlFlow.java
  KnockoutControlFlow<T>
    bind(String, Object, Object) : T
    component(Object) : T
    foreach(Object) : T
    foreach(Object, Object) : T
    generateObservables(String...) : String
    if_(Object) : T
    if_(Object, Object) : T
    ifnot(Object) : T
    ifnot(Object, Object) : T
    with(Object) : T
KnockoutFactory.java
  KnockoutFactory
    INSTANCE
    cdn(P) <P extends HTMLPage> : P
    from(H) <H extends HTMLElement<?>> : Knockout<H>
    getHTMLFactory() : HTMLFactory
    virtualElement(Object...) : KnockoutVirtualElement
KnockoutFilter.java
KnockoutVirtualElement.java
  KnockoutVirtualElement
    getContent() : List<Object>
    isEmpty() : boolean
```

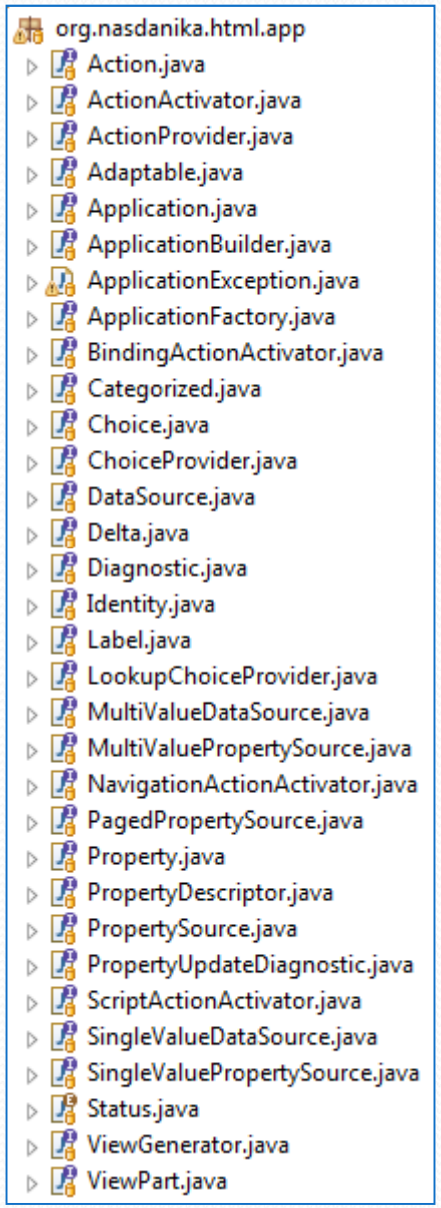
```
Knockout<H extends HTMLElement<?>>
  attr(Object) : Knockout<H>
  checked(Object) : Knockout<H>
  checked(Object, Object) : Knockout<H>
  click(Object) : Knockout<H>
  css(Object) : Knockout<H>
  disable(Object) : Knockout<H>
  disable(Object, Object) : Knockout<H>
  enable(Object) : Knockout<H>
  enable(Object, Object) : Knockout<H>
  event(Object) : Knockout<H>
  hasFocus(Object) : Knockout<H>
  hasFocus(Object, Object) : Knockout<H>
  html(Object) : Knockout<H>
  html(Object, Object) : Knockout<H>
  options(Object) : Knockout<H>
  options(Object, Object) : Knockout<H>
  selectedOptions(Object) : Knockout<H>
  selectedOptions(Object, Object) : Knockout<H>
  style(Object) : Knockout<H>
  submit(Object) : Knockout<H>
  template(Object) : Knockout<H>
  text(Object) : Knockout<H>
  text(Object, Object) : Knockout<H>
  textInput(Object) : Knockout<H>
  textInput(Object, Object) : Knockout<H>
  toHTMLElement() : H
  uniqueName(Object) : Knockout<H>
  value(Object) : Knockout<H>
  value(Object, Object) : Knockout<H>
  visible(Object) : Knockout<H>
  visible(Object, Object) : Knockout<H>
```

Application - philosophy

- Abstractions to thinks of user-system interaction as:
  - System invokes a user by passing them a callback (user) interface with actions to activate
  - Actions form a vocabulary of system-user interactions
  - The framework takes care of generating an HTML UI from actions and property sources
- Subject - Verb – Object => User - Action - Property source:
  - "Customer views account details":
    - Customer - user
    - Views account details – view action for "account" property source
- If it can be articulated, it can be automated

Application – key abstractions

- Label – something with text and icon
- Action – a label which can be activated by a user
- Data sources and properties – low-level data access
- Property sources and property descriptors – higher-level data access abstractions with UI attributes and actions
- Application – header, navigation bar, navigation panel, content panel, footer
- Application Builder – builds application
- Action Application Builder – builds application from an action tree
- ViewPart – contributor to UI construction
- ViewGenerator – provides common generation methods and access to factories



EMF

- EMF adapters to the application abstractions
- Use default implementations or customize
- Register with a resource set:

```
ComposedAdapterFactory composedAdapterFactory = new ComposedAdapterFactory();

composedAdapterFactory.registerAdapterFactory(
        new SupplierAdapterFactory<ApplicationFactory>(
                ApplicationFactory.class,
                this.getClass().getClassLoader(),
                BootstrapContainerApplicationFactory::new));

composedAdapterFactory.registerAdapterFactory(
        new FunctionAdapterFactory<ApplicationBuilder, EObject>(
                ApplicationBuilder.class,
                this.getClass().getClassLoader(),
                ViewActionApplicationBuilder::new));

composedAdapterFactory.registerAdapterFactory(
        new FunctionAdapterFactory<ViewAction, EObject>(
                ViewAction.class,
                this.getClass().getClassLoader(),
                EObjectViewAction::new));

resourceSet.getAdapterFactories().add(composedAdapterFactory);
```

- Adapt EObject to generate HTML UI:

```
Application application = EObjectAdaptable.adaptTo(eObj, ApplicationFactory.class).createApplication();
ApplicationBuilder applicationBuilder = EObjectAdaptable.adaptTo(eObj, ApplicationBuilder.class);
applicationBuilder.build(application);
```

- Can be used to generate static content and in dynamic Web applications