Architecture

Introduction

- Applications developed using Vaadin include a web application servlet based part, user interface components, themes that dictate the look & feel and a data model that enables the binding of user interface components with data sources.

The `com.vaadin.Application` Class

- When developing a Vaadin based web application we define a class that extends `com.vaadin.Application`.
- For each session the server maintains an object instantiated from that class we define.
- That object is responsible for creating the user interface components and for receiving their events.

The User Interface Components

- The user interface components are created and laid out by the application.
- Each server side component has a client side Java Script based component with whom the user interacts.
- The server side component communicates with its client side component.

User Interface Events

- When the user interacts with the user interface events are created. They are first processed on the client side using JavaScript, and then they are passed over all the way through the HTTP server to the terminal adapter.
- The terminal adapter is responsible for communicating this data to the server side component. We can handle those events on the server side.

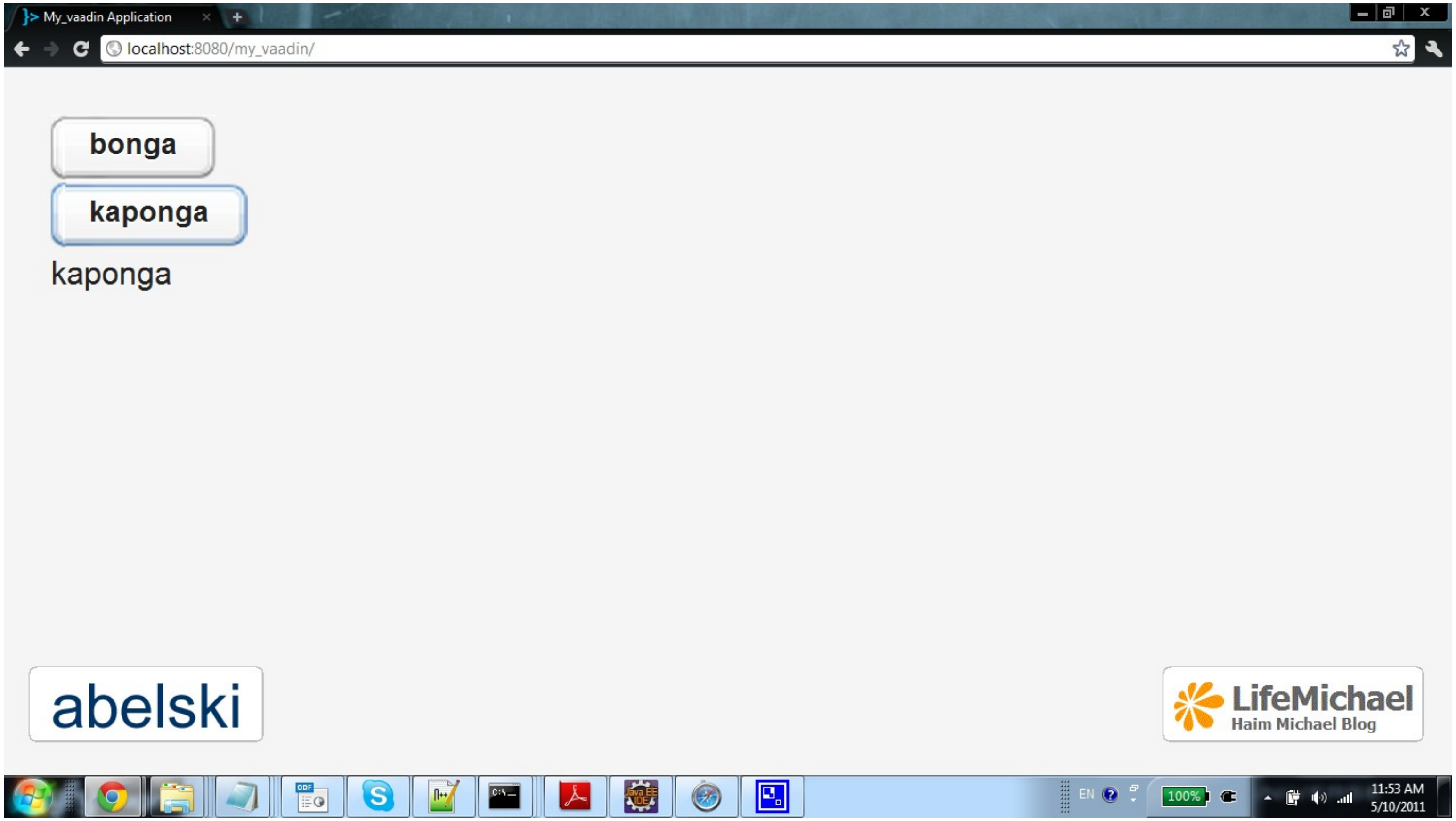
User Interface Events

```
package com.example.my_vaadin;

import com.vaadin.Application;
import com.vaadin.ui.*;
import com.vaadin.ui.Button.ClickEvent;
import com.vaadin.ui.Button.ClickListener;

public class My_vaadinApplication extends Application
{
    @Override
    public void init()
    {
        Window mainWindow = new Window("My_vaadin Application");
        final Label label = new Label("Hello Vaadin user");
        Button btBonga = new Button("bonga");
        Button btKaponga = new Button("kaponga");
        mainWindow.addComponent(btBonga);
        mainWindow.addComponent(btKaponga);
        mainWindow.addComponent(label);
```

User Interface Events

```
        btKaponga.addListener(new ClickListener()
        {
            private static final long serialVersionUID = 1L;
            public void buttonClick(ClickEvent event)
            {
                label.setValue("kaponga");
            }
        });
        btBonga.addListener(new ClickListener()
        {
            private static final long serialVersionUID = 1L;
            public void buttonClick(ClickEvent event)
            {
                label.setValue("bonga");
            }
        });
        setMainWindow(mainWindow);
    }
}
```

User Interface Events



kaponga

Client Side Engine

- The client side engine is responsible for rendering the user interface. It uses the Google Web Toolkit (GWT) framework.
- The client side engine communicates with the server side Terminal Adapter and informs it about user interactions.
- The client side engine uses the UIDL language when informing the Terminal Adapter about user events. UIDL stands for User Interface Definition Language. It is a JSON based language. The communication is mad using asynchronous HTTP or HTTPS requests.

Terminal Adapter

- The controls reside on the server side. They render themselves back to the web browser using the terminal adapter.
- The terminal adapter is an additional abstraction layer responsible for rendering the user interface controls back to the web browser.

Terminal Adapter

- The client side engine communicates with the terminal adapter using advance asynchronous requests. The terminal adapter passes over these requests to the relevant controls.

Terminal Adapter

- We can adjust the terminal adapter to work with any technology we want. We are not limited for using the Google Web Toolkit. It will be transparent for our application code.

Themes

- While the user interface logic is handled by the code we write in Java its presentation is handled by the themes via CSS.
- Vaadin provides default themes we can use. We can develop new ones.
- The themes can include HTML templates that define a custom layout.
- The themes can include images in use by the CSS and/or the HTML templates.

UIDL

- The terminal adapter renders the user interface back to the web browser using UIDL.
- UIDL stands for User Interface Definition Language. It is a JSON based language.

www.uidl.net

Data Models

- The Vaadin framework provides us with a data model we can use when interfacing the data the user interface components display.
- The user interface components can use that interface for updating the application data in a direct way. We can bind a control with a specific separated data source, such as a specific database table.

AJAX

- Ajax stands for Asynchronous JavaScript and XML. It is a technique for writing JavaScript code that interacts with the server side and updates the user interface in an asynchronous way.

Google Web Toolkit

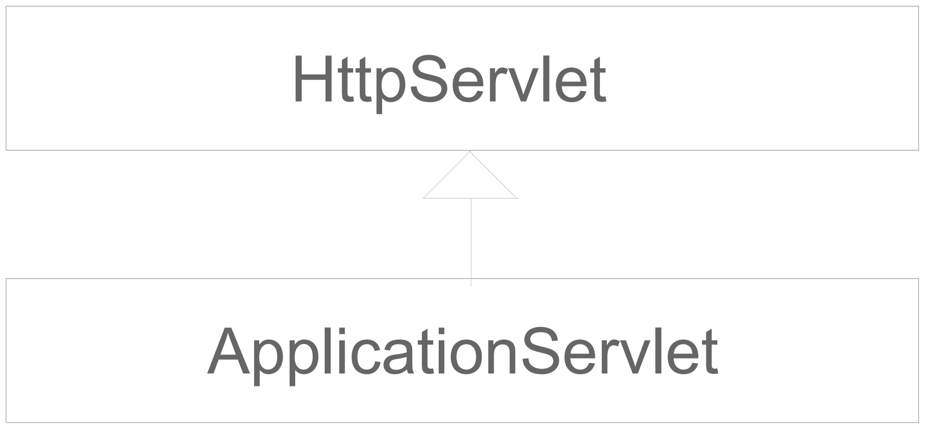
- The Google Web Toolkit (GWT) is a software development kit for developing rich internet applications without having to use JavaScript or other web browser technologies.

JSON

- JSON stands for JavaScript Objects Notation. It is a light weight data interchange format we can easily generate and parse.
- Parsing JSON messages is much faster comparing with XML ones.

Java Servlets

- The Vaadin framework is implemented on top of the Java Servlets API.
- When the web container receives the first request for the URL address the application is registered with, it instantiates the `ApplicationServlet` class.



Java Servlets

- When the first HTTP request for our application URL arrives the `ApplicationServlet` class is instantiated.
- Using the `HttpSession` interface each session is associated with an `Application` instance.
- During the application lifetime each user action is related to the proper `Application` instance.