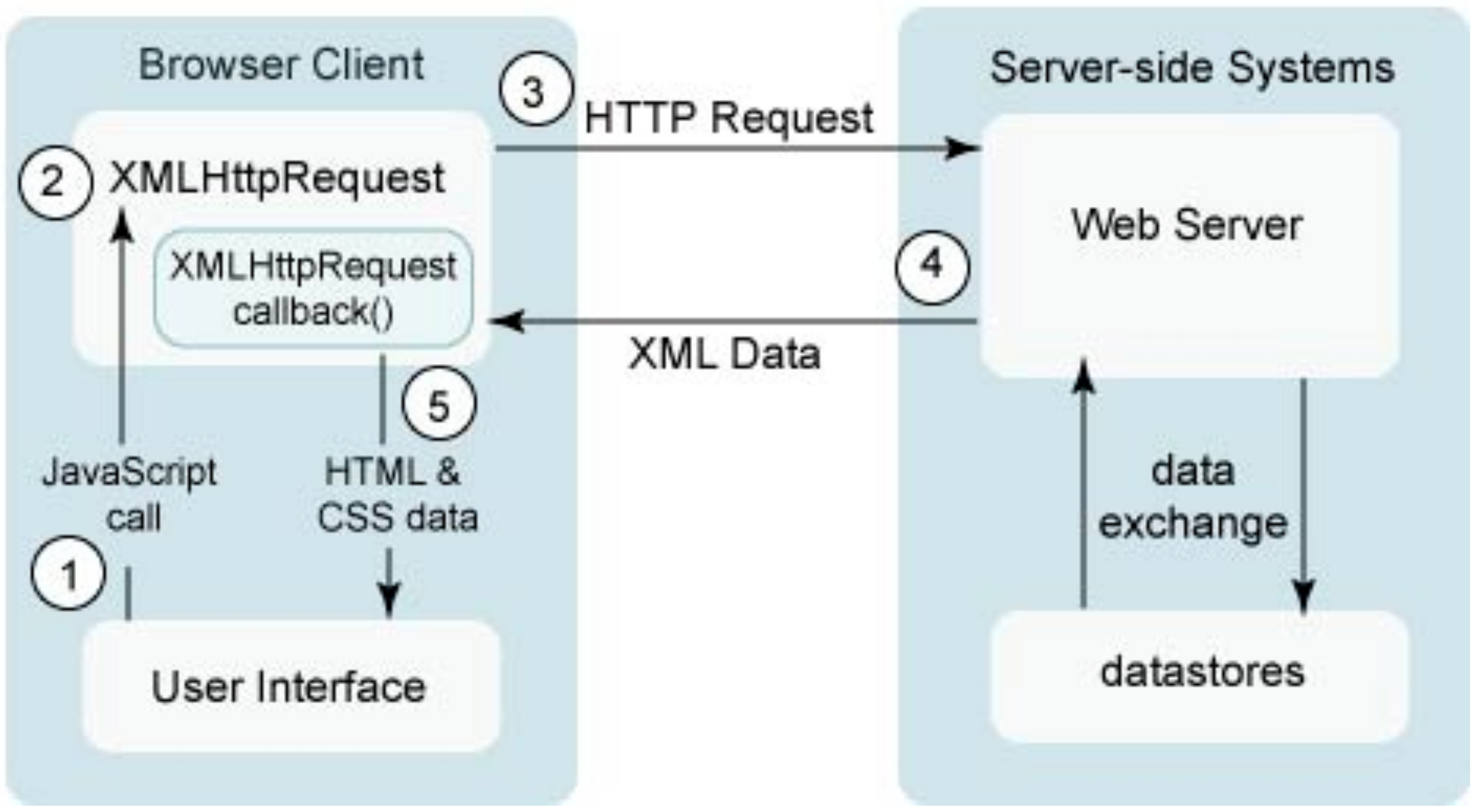
JSF and AJAX with Netbeans 5.5

Wanasanan Thongsongkrit

(NAS) :)

AJAX


Browser Client
Server-side Systems
3
HTTP Request
2
XMLHttpRequest
XMLHttpRequest callback()
Web Server
4
XML Data
5
JavaScript call
HTML & CSS data
1
data exchange
User Interface
datastores

AJAX's shortcoming

- Because AJAX is new, it has very inconsistent support among browsers.
- Also, to develop with AJAX, you need to have some knowledge of JavaScript, which is out of reach for many page authors.

Learning AJAX

- Fast (easy) if you
  - are a JavaScript guru
  - have memorized the entire DOM API
  - own and study books on DHTML, JavaScript, CSS, AJAX and useful hacks for each technology
- Slow (hard) if you
  - come from a mostly static HTML/CSS background
  - are comfortable with traditional web application architectures built around an HTTP POST
  - primary use of JavaScript is cut-and-paste of cute animations and other cool in-page behaviors

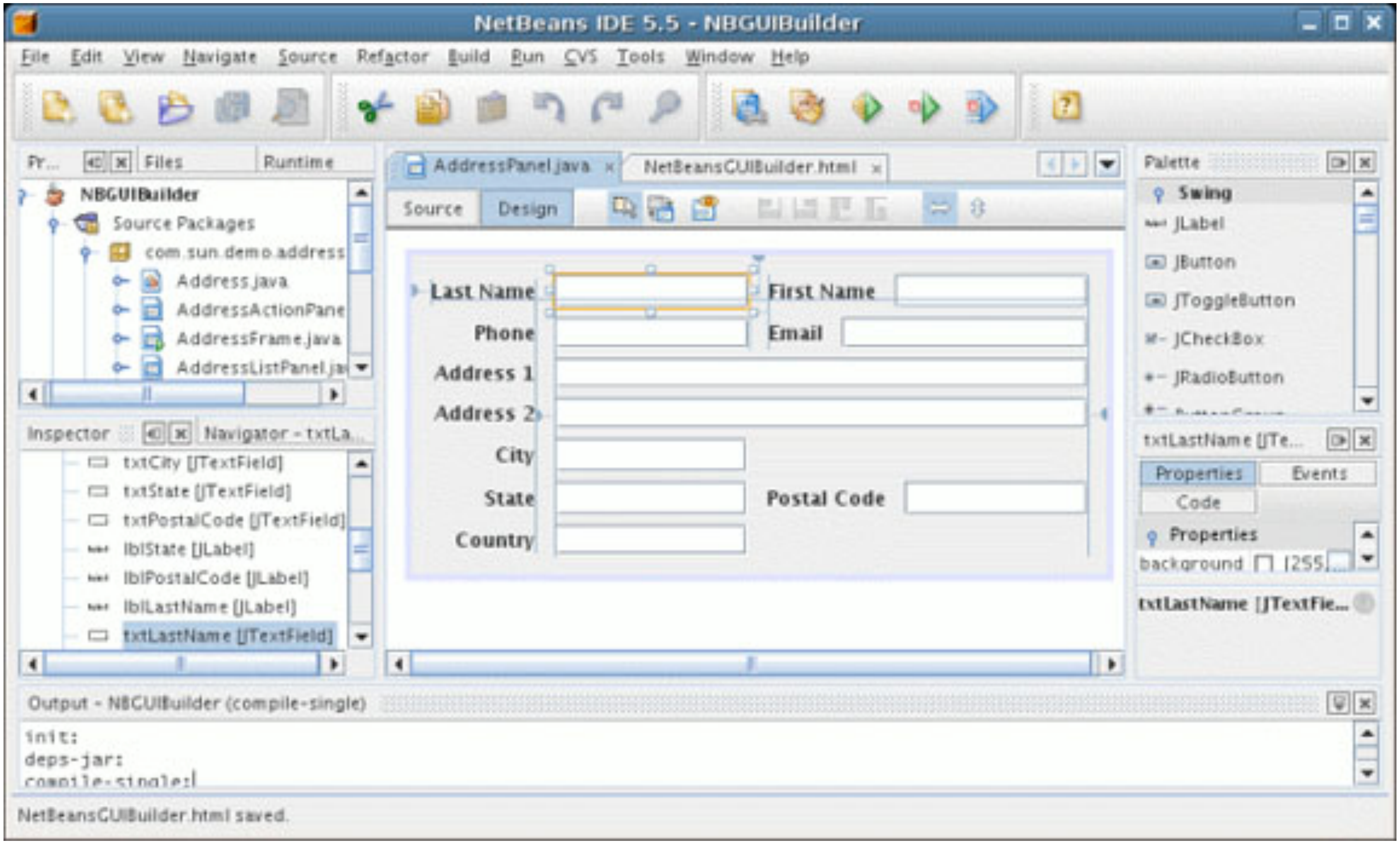
AJAX toolkits

- The complete list indicates some **160** toolkits exist
- Keith provided a pointer to a popularity survey of AJAX toolkits (as of September 23, 2006)
  - Prototype (48%)
  - Script.aculo.us (33%)
  - Dojo (19%)
  - DWR (12%)
  - Moo.fx (11%)
  - jQuery (7%)
  - Rico (5%)
  - Yahoo UI (5%)
  - Atlas (4%)
  - MochiKit (4%)
  - XAjAX (4%)
  - GWT (3%)

How to avoid learning javascript and all toolkits?

- Use components that encapsulate AJAX inside
- Benefits
  - Hide functionality behind simple building blocks
  - Page author do not have to write all java scripts themselves but let the component do the complicated work
  - Page authors have an easier time maintaining their pages
  - Reusable components
- Technology used: Java Server Faces (JSF)
  - author can just drag and drop the components onto a page using a tool such as Sun Java Studio Creator or the NetBeans IDE.

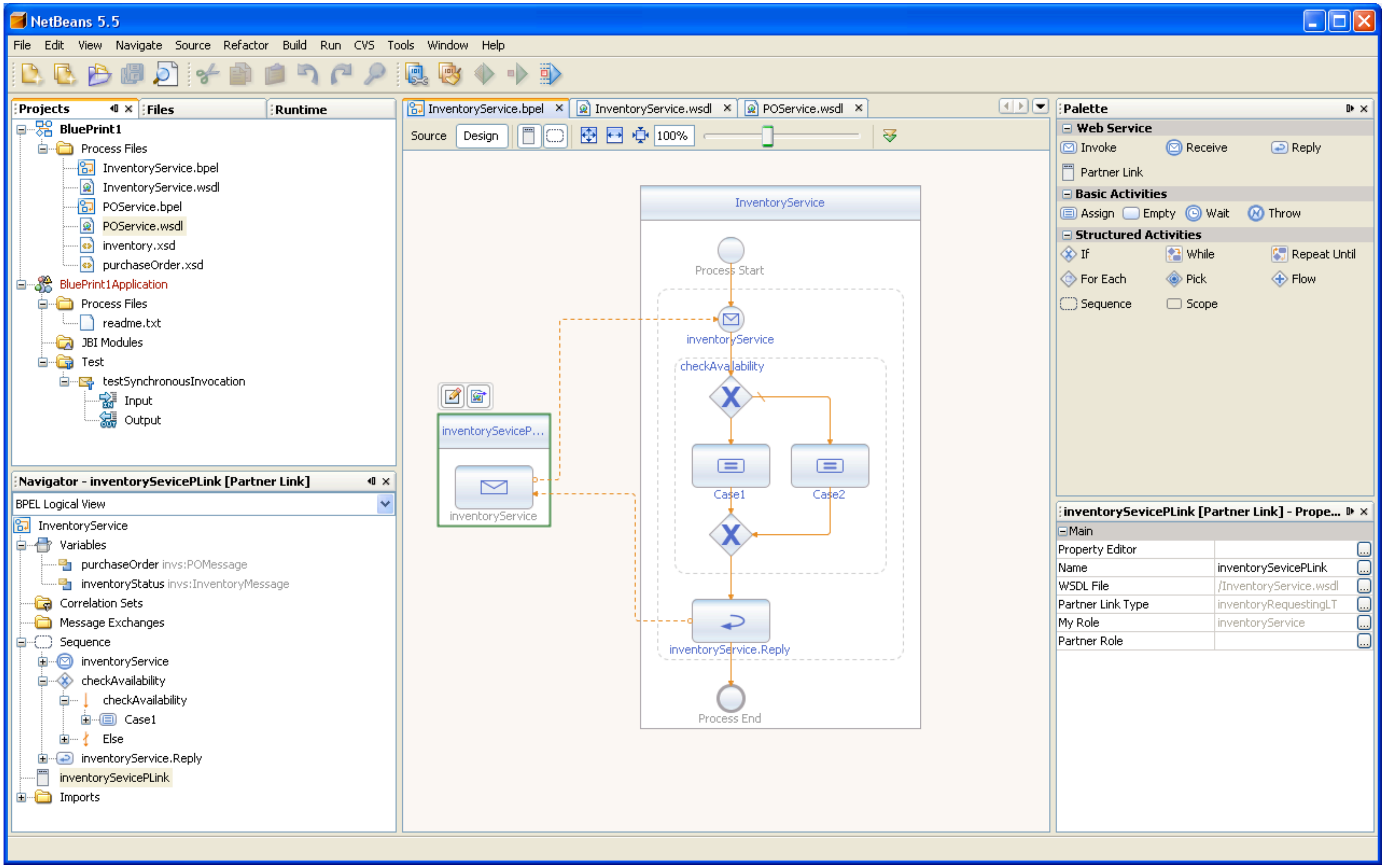
Create Great-Looking GUIs With NetBeans IDE 5.5



```
init:
deps-jar:
compile-single:
```

NetBeansGUIBuilder.html saved.

NetBeans Enterprise Pack (Beta version)



Property Editor	
Name	inventorySevicePLink
WSDL File	/InventoryService.wsdl
Partner Link Type	inventoryRequestingLT
My Role	inventoryService
Partner Role	

jMaki Framework (plug-in)

- JavaScript Wrapper framework for the Java platform
- wraps popular AJAX frameworks into a JSP or JSF tag
- Provides a common programming model to developers
- Familiar to Java EE application developers
- Leverages the widgets from popular frameworks (See)

Dojo	Flickr	Google	Scriptaculus
Mochikit	Spry	Yahoo	DHTML

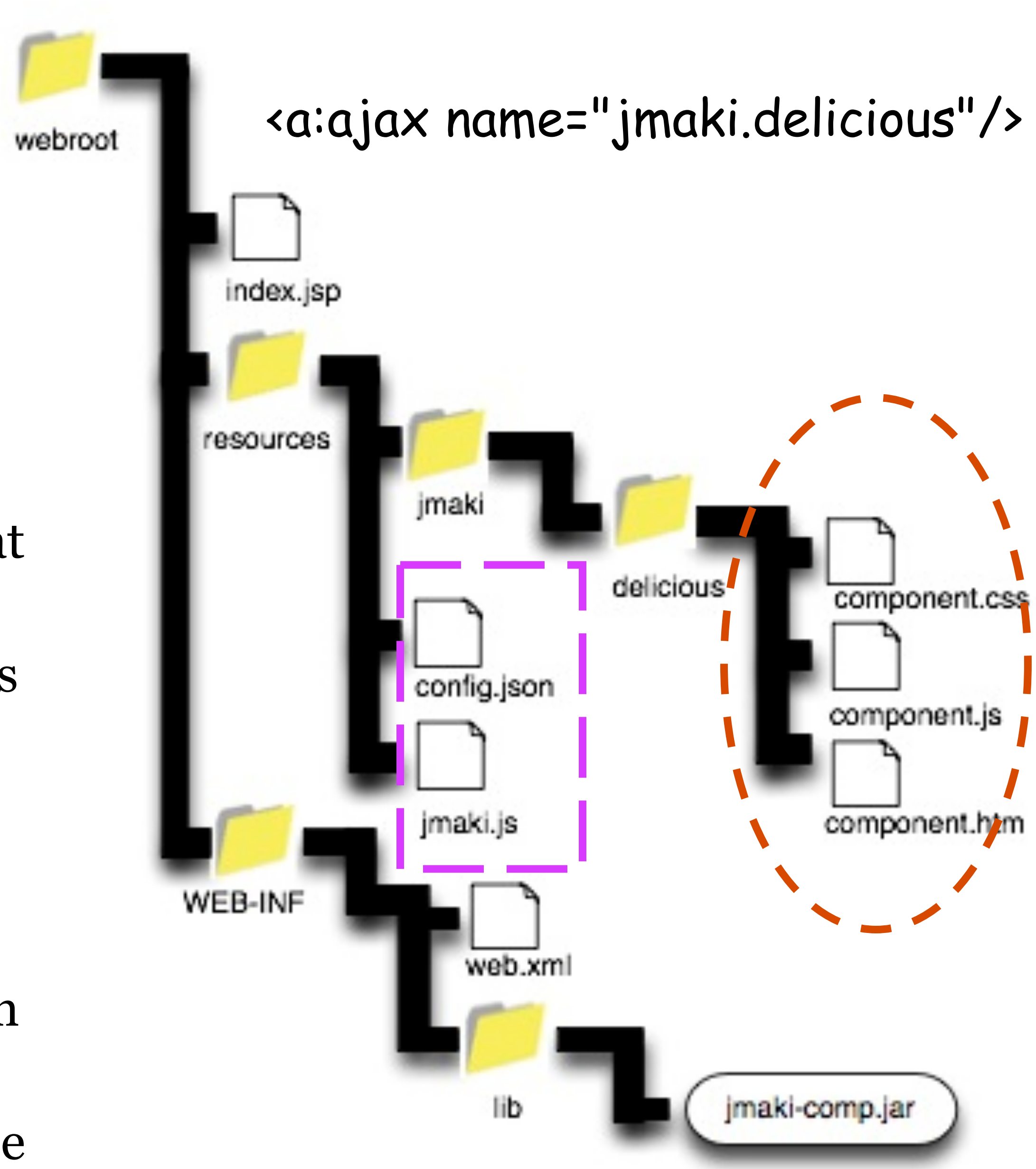
- What you need is: jMaki Plug-in

https://ajax.dev.java.net/files/documents/3115/41646/org-netbeans-modules-sun-jmaki.nbm

Basic jMaki Application Structure

jmaki.js ➪ the JavaScript bootstrapper and utilities that manages

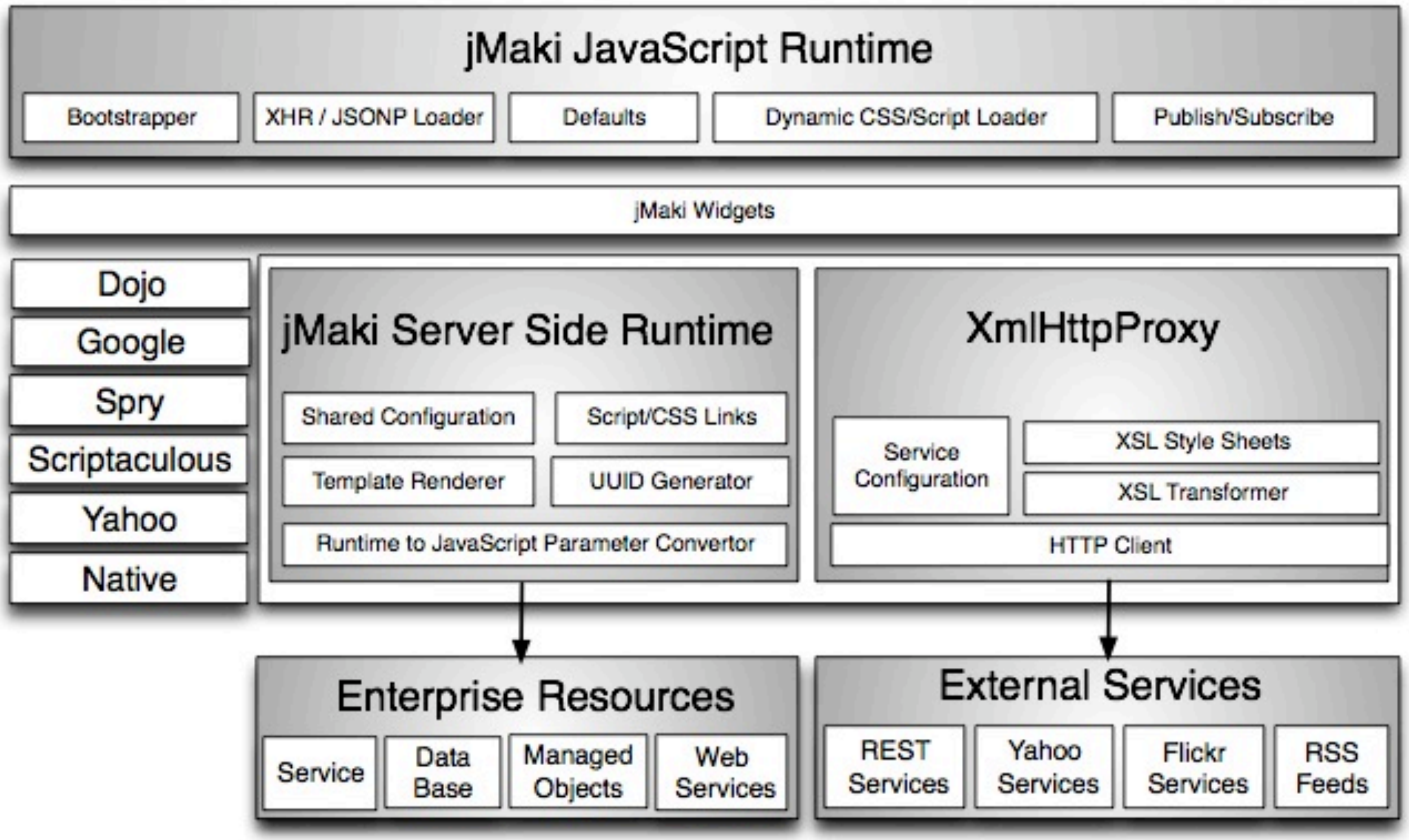
- the loading of jMaki widgets on the client,
- makes XMLHttpRequests,
- loads additional resources,
- provides inter-widget communication using publish and subscribe
- stores widget instances to be shared across an application.



config.json ➪ configuration of 3rd party libraries used by jMaki

jMaki:

- made up of JavaScript Runtime, the Server Side Runtime, and the XmlHttpProxy.



JavaScript Runtime (jmaki.js)

- responsible for
  - bootstrapping all widgets and passing parameters provided by a server-side runtime.
  - makes sure that each widget instance gets the correct parameters that were passed from the server-side runtime.
  - uses default parameters (if not provided) that may then be customized for each widget.
  - provides convenient APIs for performing an XMLHttpRequest and loading resources based on JSON with Padding (JSONP).
  - provides APIs for a widget to load extra scripts, styles, and resources needed by a widget.
  - provides a **publish subscribe mechanism** for widget-based communication.
  - provides a common namespace to store and access widgets

The key point of the API is that you can program to one API and access widgets from any given toolkit.

Server-Side Runtime

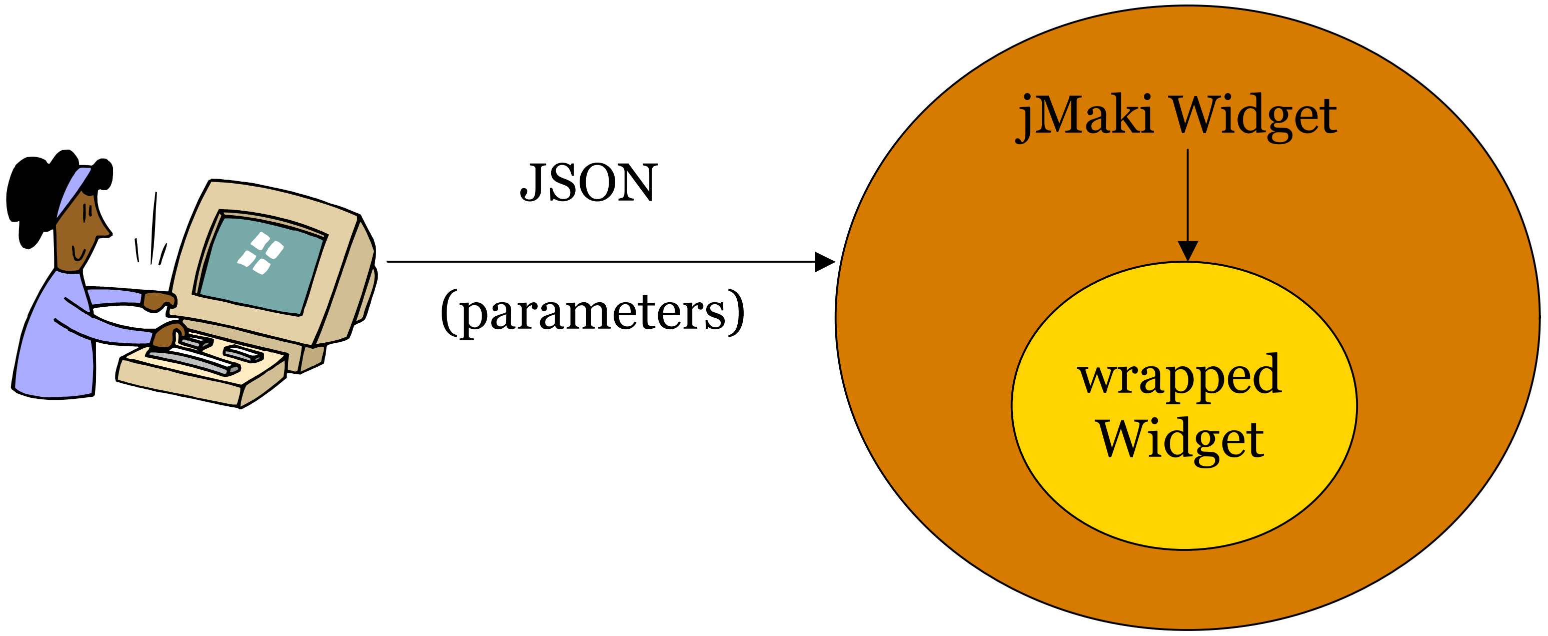
- responsible for
  - applying changes and rendering HTML templates.
  - renders all script and CSS references based on which type is centrally configured.
  - responsible for serializing parameters (specified as attributes in a JSP or JSF tag) that are passed to the JavaScript runtime.
  - capable of mapping widget values back into server-based model data, such as managed objects, web services, or databases.

XmlHttpProxy

- provides a generic JSON-based access to a widget range of XML-based services using an HTTP client.
  - services include RSS feeds, Yahoo services such as geocoding, Flickr image searches, and many more to come.
- allows widgets to access services in a uniform way by providing XSL-to-JSON transformations that can be easily customized.

How author configure widgets' parameters via jMaki?

- using JSON



Using Your Own Data With a jMaki Widget

- to add your own data to a widget (JSON format):
  - Using a static file
  - Using a JavaServer Faces managed bean
  - Using a JSP page or a servlet

Demo: Publish and Subscribe Mechanism with Yahoo Geocoder

```
<a:ajax name="yahoo.geocoder" service="/xhp"/>
```

widget

```
<script type="text/javascript">
  function geoCoderListener(coordinates) {
  var targetDiv = document.getElementById("geocoder001_message");
  var reponseText  = "";
  for (var i in coordinates) {
     reponseText += "Latitude=" + coordinates[i].latitude + " Longitude=" +
    coordinates[i].longitude + "<br>";
  }
  targetDiv.innerHTML = reponseText;
  }
  // subscribe to the topic '/yahoo/geocode' to which this widget publishes events
  jmaki.subscribe("/yahoo/geocoder", geoCoderListener);
 </script>
```

Subscribe

```
<div id="geocoder001_message"></div>
```

Display location

Geocoder's Component.html (hidden)

```
<div id="${uuid}">
 <form
   onsubmit="jmaki.attributes.get('${uuid}').getCoordinates();
   return false;">
   Location:  <input type="text" id="${uuid}_location">
   <input type="button" value="Get Coordinates"
   onclick="jmaki.attributes.get('${uuid}').getCoordinates();">
 </form>
</div>
```

Geocoder's Component.js (hidden)

```
if (typeof jmaki.GeoCoder == 'undefined'){

 jmaki.GeoCoder = function(_widget) {
   var topic = "/yahoo/geocoder";

   var uuid = _widget.uuid;
   var service = _widget.service;
   if (typeof widget.args != 'undefined' &&
      typeof widget.args.topic != 'undefined') {
          topic = widget.args.topic;
   }
```

$\longleftarrow$ *uses default parameters*

Geocoder's Component.js (hidden)

```
var location;
this.getCoordinates = function() {        ← Wrapped function
    location =
    encodeURIComponent(document.getElementById(uuid +
    "_location").value);

    var encodedLocation = encodeURIComponent("location=" +
    location);

    var url = service + "?key=yahoogeocoder&urlparams=" +
    encodedLocation;

    jmaki.doAjax({url: url, callback: function(req) { var _req=req;
    postProcess(_req);}});

  }
```

Componet.js (hidden)

```
function postProcess(req) {
      if (req.readyState == 4) {
         if (req.status == 200) {
            var response = eval("(" + req.responseText + ")");
            jmaki.publish(topic, response.coordinates);
         }
      }
   }
}
}

var geocoder = new jmaki.GeoCoder(widget);

// add to the instance map for later refernece
jmaki.attributes.put(widget.uuid, geocoder);
```

Publish response

References

- https://ajax.dev.java.net/
- https://ajax.dev.java.net/download.html
- http://javaserver.org/jmaki/
- http://www.netbeans.org/
- http://java.sun.com/javaee/javaserverfaces/ajax/tutorial.jsp
- http://www.javapassion.com/handsonlabs/ajaxjmakiintro/
- http://www.google.com/apis/maps/