



Java is a trademark of Sun Microsystems, Inc.

JavaOneSM

Ajaxifying Legacy Web Applications

Anas Mughal

Bluenog

<http://www.bluenog.com>

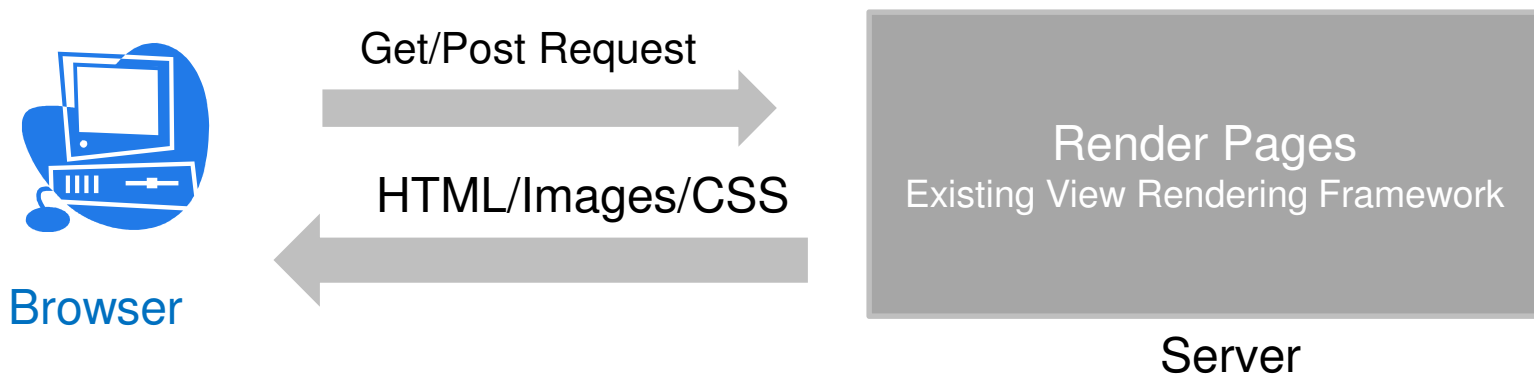


Topics

- > Overview: Legacy vs. RIA
- > Design Approach
- > Data Formats
- > Tools
- > Demo
- > Ajaxifying Portal
- > Portal Demo
- > QA

Web 1.0 Applications

- > Pages are rendered server-side
- > Client-side is stateless
- > Server-side Model-View-Controller (MVC)
- > Full-page refresh



Ajax Applications

- > State-full client
- > Server-side provides data services
- > Client-side MVC framework
- > No full-page refresh



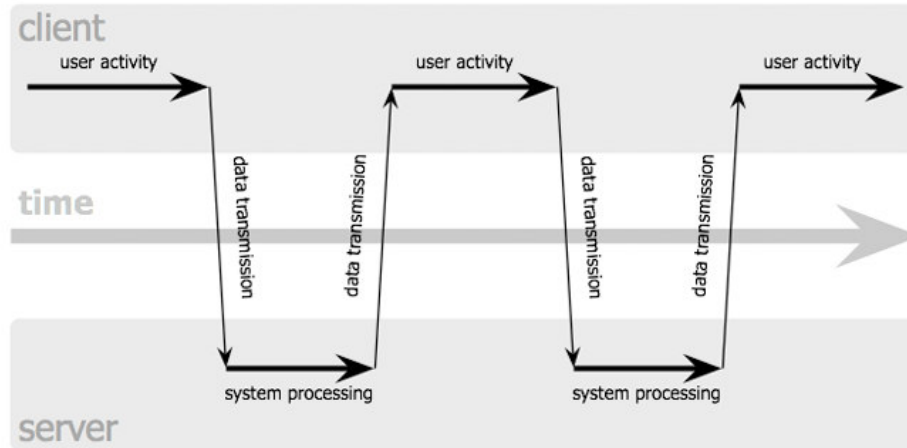
Browser



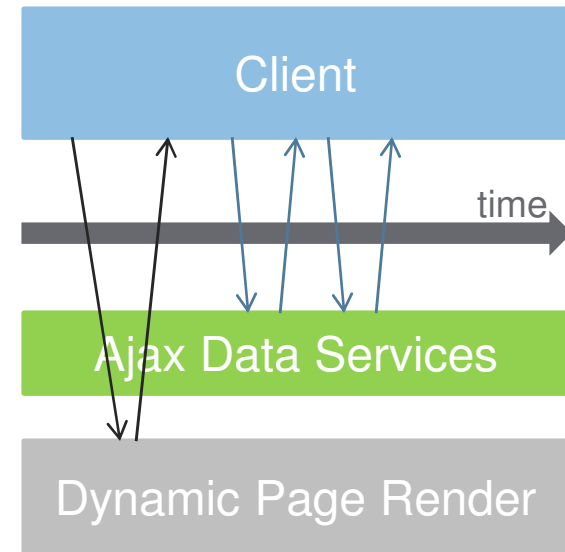
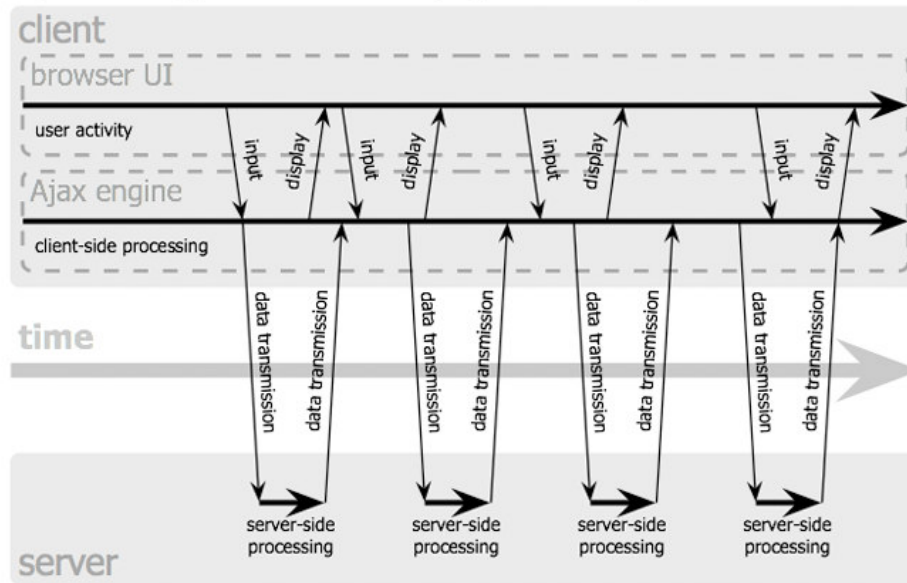
Ajax Data Services

Server

classic web application model (synchronous)



Ajax web application model (asynchronous)

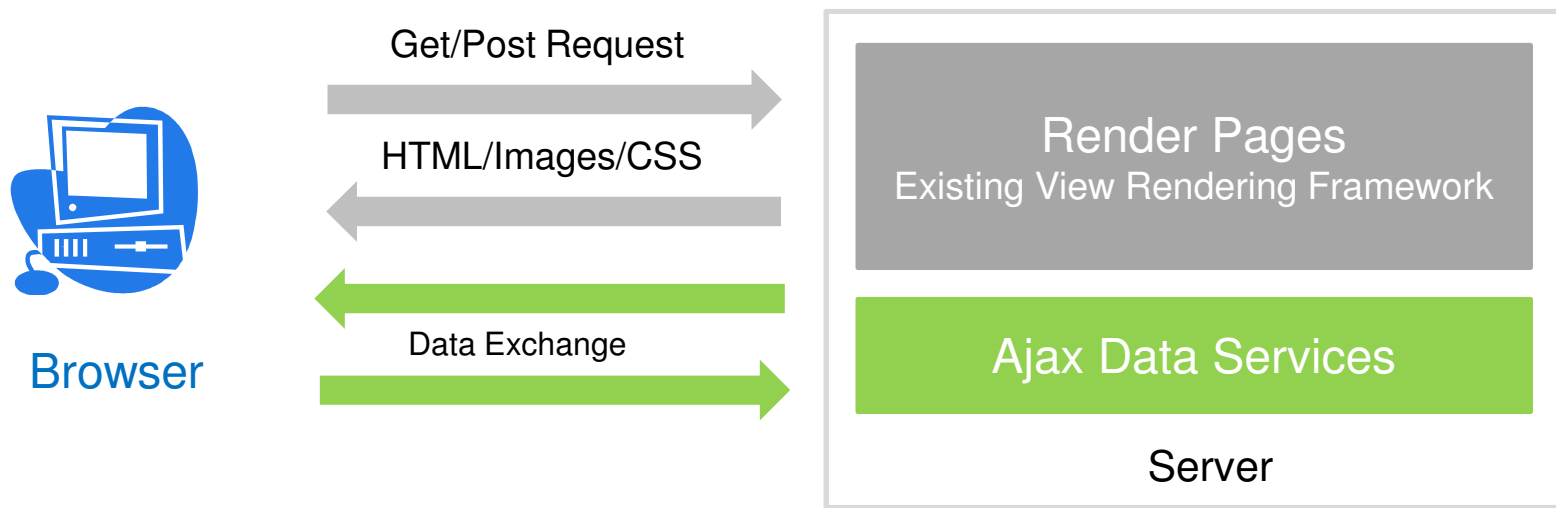


Suggested Model

Suggested Approach

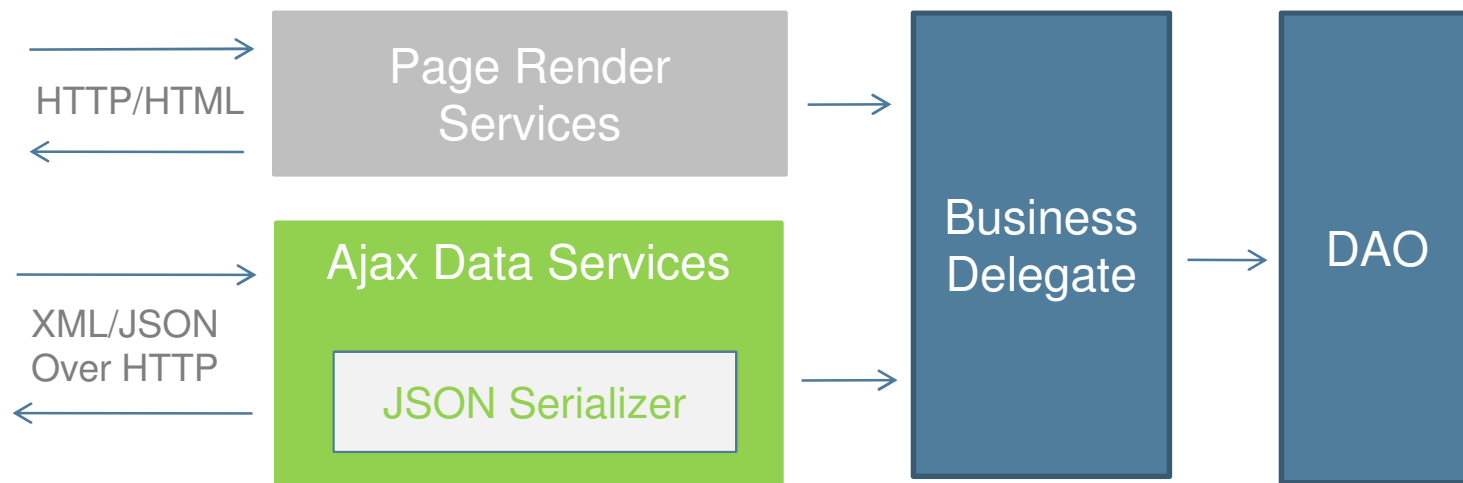
(for Ajaxifying legacy Applications)

- > Pages rendered server-side
- > Keep Server-side MVC
- > Add Ajax Data Services on server-side
- > Add Ajax **controls/widgets** on client-side



Design Approach

- > Incorporate Ajax data services layer
- > Integrate Ajax layer with business delegates
- > Incorporate Ajax widgets in View Templates



Ajaxifying Steps

- > Clean HTML -- HTML Validator
- > Clean CSS
- > Select Data Transport Format
- > Select Ajax Framework
- > Build Ajax Handlers
- > Incorporate Ajax Widgets

Data Transport Formats

- > JSON (JavaScript Object Notation)
 - Native JavaScript structure
 - Lightweight / compact
 - Retains type information

```
{"names": ["John", "Kevin"], "count": 3 , "price": 4.75}
```

55

- > XML

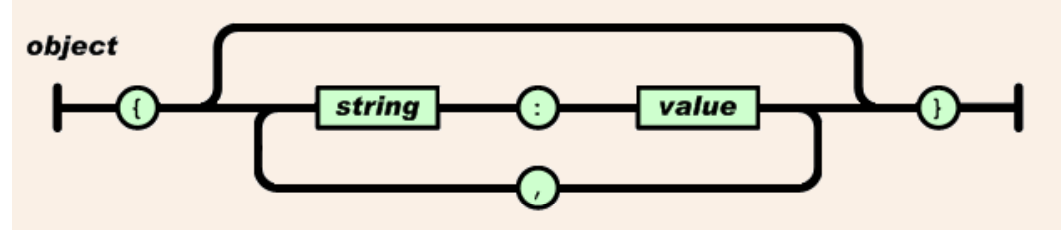
- Parsing is CPU intensive
- DOM may require large memory
- Does not retain type information
- Suited for data transformation needs

```
<xmldata>  
  <names>  
    <name>John</name>  
    <name>Kevin</name>  
  </names>  
  <count>3</count>  
  <price>4.75</price>  
</xmldata>
```

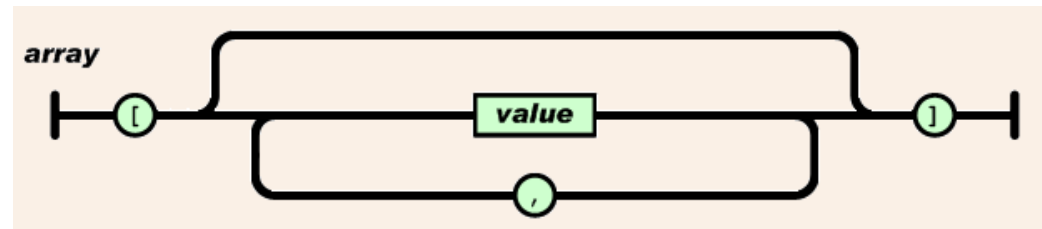
104

JSON Representation

- > A collection of name/value pairs.



- > An ordered list of values.



JSON Sample

```
{
  "claims" : [
    {
      "gender" : "M",
      "claimNumber" : "60801021250",
      "providerName" : "BEST PRACTICES INC",
      "charges" : 380,
      "firstName" : "fypfqcv",
      "lastName" : "ijlp"
    },
    {
      "gender" : "F",
      "claimNumber" : "60801054339",
      "providerName" : "BEST PRACTICES INC",
      "charges" : 453,
      "firstName" : "xxjmvdfg",
      "lastName" : "tiitupai"
    }
  ]
}
```

JSON Support on the Server-Side

- > Spring MVC Framework
 - Spring-json View (<http://spring-json.sourceforge.net>)
 - Supports: SOJO and JSON-lib
- > Struts Framework
 - Struts JSON Plugin (<http://tinyurl.com/b87ndu>)
 - Serializes entire action class variables
 - Provides incoming request interceptor
- > Build your own
 - Convert Java structures using a JSON library
 - Set content-type to “application/json”

Spring-JSON View

- > Include spring-json Library
- > Declare JSON View Bean

```
<bean id="jsonView" class="org.springframework.w.servlet.view.json.JsonView">
    <property name="jsonWriter"><ref bean="sojoJsonWriter"/></property>
    ...
</bean>
```

- > Assign JSON View in Controller:

```
public ModelAndView
handleRequest(HttpServletRequest, HttpServletResponse) {
    Map model = new HashMap();
    model.put("claims", claimDelegate.findClaims());
    return new ModelAndView("jsonView", model);
}
```

From Legacy to Ajax (Spring JSON View)

```
ResultsLegacyController.java
/**
 * Claims Results Legacy Controller
 */
public class ResultsLegacyController implements Controller {

    /** Logger */
    protected final Log logger = LoggerFactory.getLog(getClass());

    /**
     * Handle Request
     * @param request
     * @param response
     */
    public ModelAndView handleRequest(HttpServletRequest request,
        HttpServletResponse response) throws ServletException {

        List list = claimDelegate.findClaims();

        Map model = new HashMap();
        model.put("claims", list);

        return new ModelAndView("results legacy", model);
    }
}

ClaimsListController.java
/**
 * Claims Ajax controller
 */
public class ClaimsListController implements Controller {

    /** Logger */
    protected final Log logger = LoggerFactory.getLog(getClass());

    /**
     * Handle Request
     * @param request
     * @param response
     */
    public ModelAndView handleRequest(HttpServletRequest request,
        HttpServletResponse response) throws ServletException {

        List list = claimDelegate.findClaims();

        Map model = new HashMap();
        model.put("claims", list);

        return new ModelAndView("jsonView", model);
    }
}
```

Struts JSON Plugin

```
<struts>
  <package name="default" namespace="/"
    extends="struts-default, json-default" >

    <action name="getJSONResponse"
      class="com.demo.JsonDataService"
      method="getJSONData">

      <result type="json" />

    </action>

  </package>
</struts>
```

<http://localhost:9090/struts2jsonSampleApp/>

Other Frameworks

- > Restlet

<http://www.restlet.org/>

- > DWR

<http://directwebremoting.org/dwr/index.html>

- > Jersey – JSR 311

<https://jersey.dev.java.net/>

Jersey Framework

JAX-RS (JSR 311)

```
import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;

@Path("/helloworld")
public class HelloWorldResource {

    @GET
    @Produces("text/plain")
    public String getHelloMessage() {
        return "Hello World";
    }
}
```

<http://localhost:9090/JerseyAjaxMApp/rest/claims>

Tools

- > Server-side Serializers: (<http://json.org>)
 - SOJO, JSON-LIB
- > Formatter: jsonformatter.curiousconcept.com
- > Visualizer: chris.photobooks.com/json/default.htm
- > Validator: www.jsonlint.com , www.jslint.com
- > Editor: jsoneditor.net -- AIR
- > Editor, Minifier, Tree View: jsoneditor.appspot.com
- > Yahoo UI Compressor: minifier, obfuscator, gzip
- > Javascript Editor: SPKet Eclipse Plugin
- > Debugger: Firebug Eclipse Plugin

JS Loader

- > Consistent loading
- > Easy patching of bugs
- > Avoid duplicate loading of libraries

```
<script src="http://myserver/jsloader/jsloader.js"></script>

<script>
    JSLoader.load("extjs", "ext", "2.2.1-g");
    JSLoader.load("myLibs", "validation", "1.0");
</script>
```

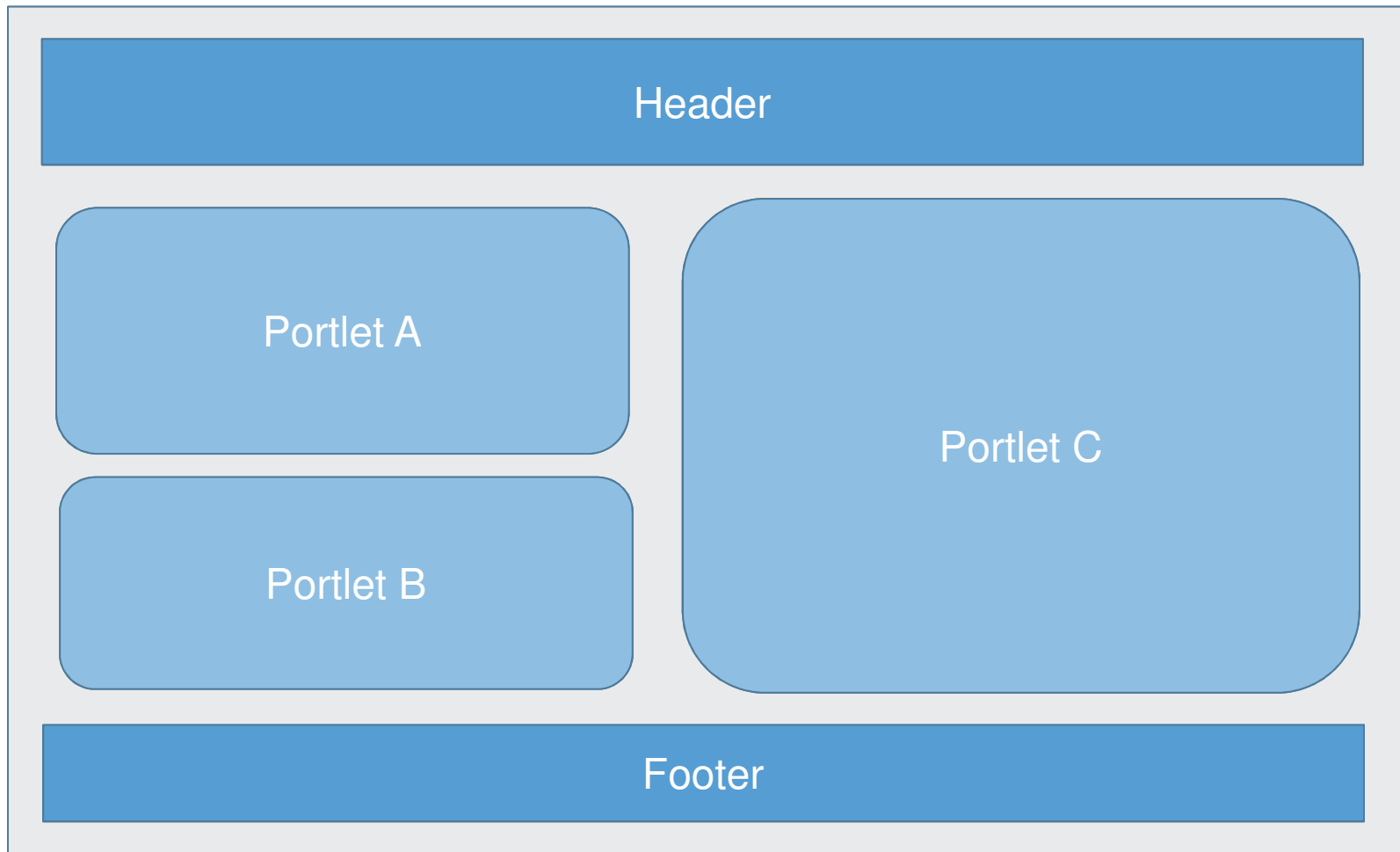
<http://www.jsloader.com>

Demo & Code Walk-thru

- > Display Results using Ajax Grid
(Data is embedded in HTML)
- > Grid Fetching Data Async
(Results not embedded in HTML)
- > Ajax Form and Fetching Grid Data Async
(Async Results)
- > Grid Supporting Pagination and Sorting
(Subset of data is fetched based on pagination and sorting)

<http://localhost:9090/ajax-demo1/>

Portals



Ajaxifying Portals

- > Layouts
 - Render using panels and layout managers
- > Portlets
 - Securing Ajax Communication
 - JSLoader
 - Inter-portlet Communication
 - Portlet View Manager

Securing Portlet Ajax Communication

> JSR 168 vs. JSR 286

- **Resource Serving**

Enables portlets to serve resources within the portlet context.

<http://developers.sun.com/portalserver/reference/techart/jsr168/>

Portlet Resource URL

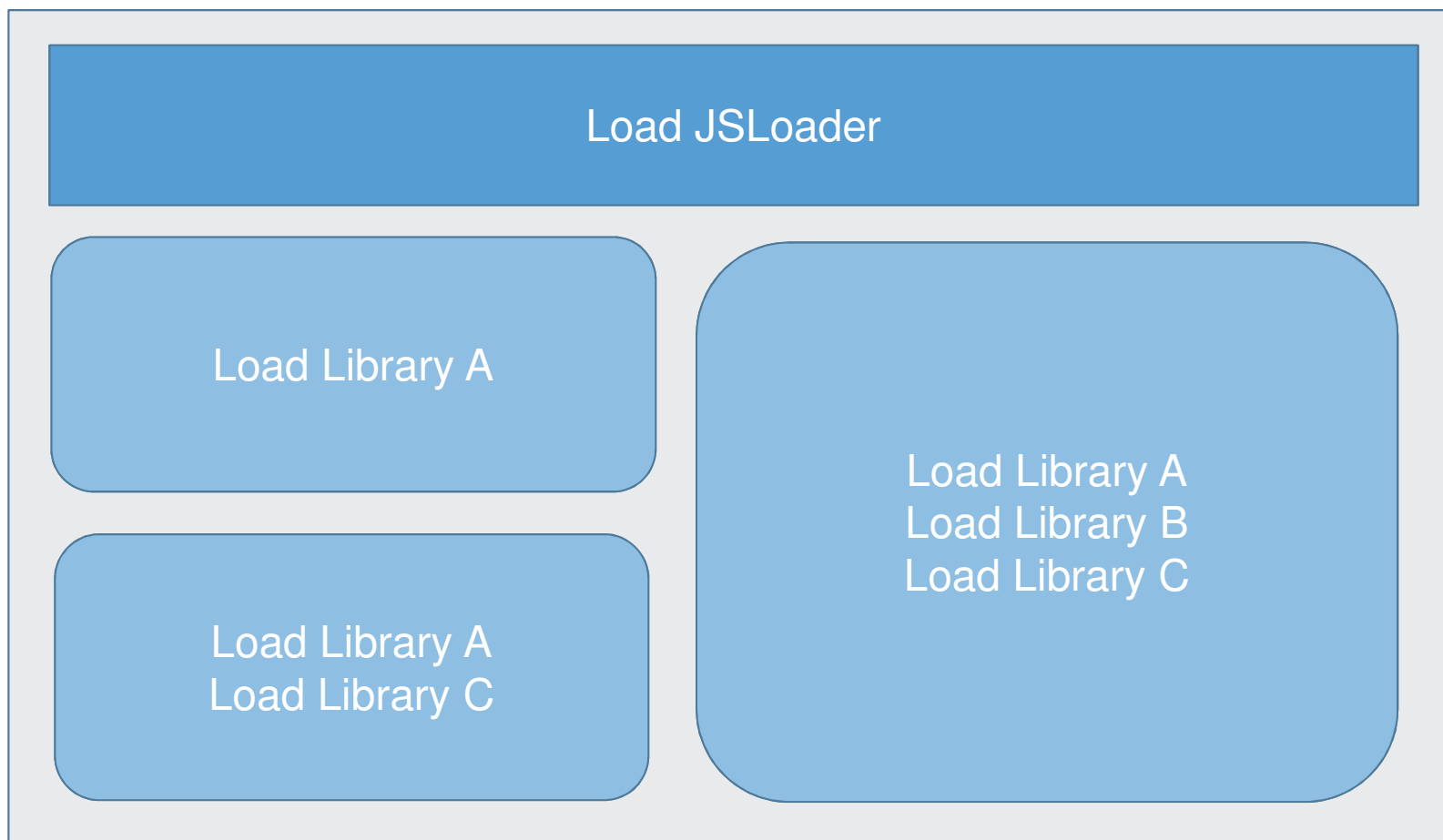
> JSR 286 Portal

```
<portlet:resourceURL var="resourceURL">  
  
</portlet:renderURL>
```

> Bluenog Portal / Jetspeed2 Portal

```
<portlet:renderURL var="resourceURL">  
  
  <portlet:param name="org.apache.jetspeed.portlet.resource.url" value="v" />  
  
</portlet:renderURL>
```


JSLoader



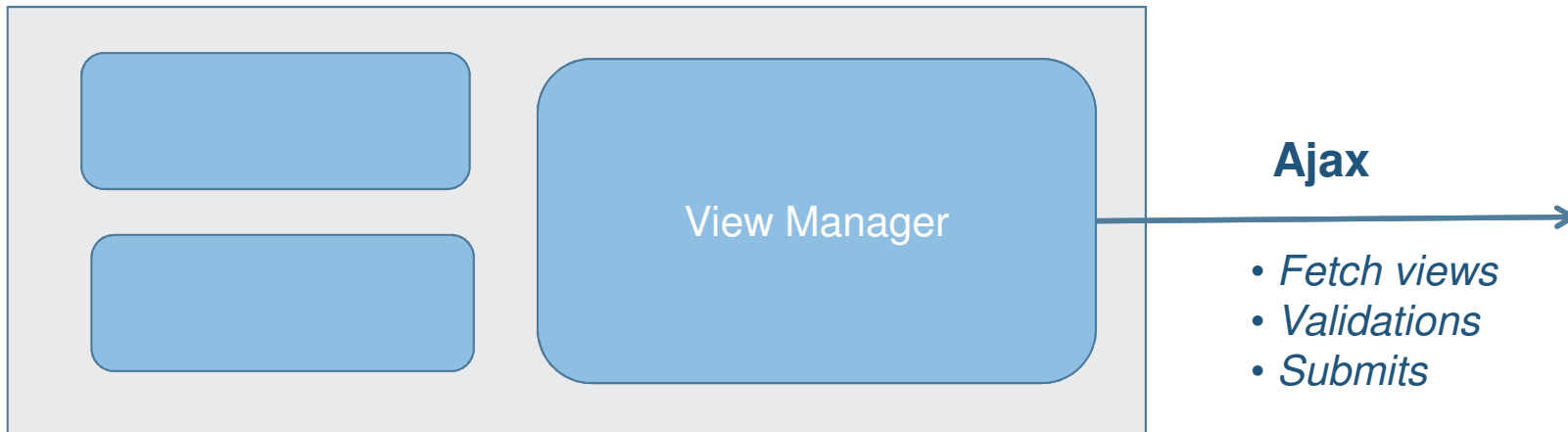
Inter-portlet Communication

- > Javascript framework eventing system
 - Most frameworks support “Publish-Subscribe”

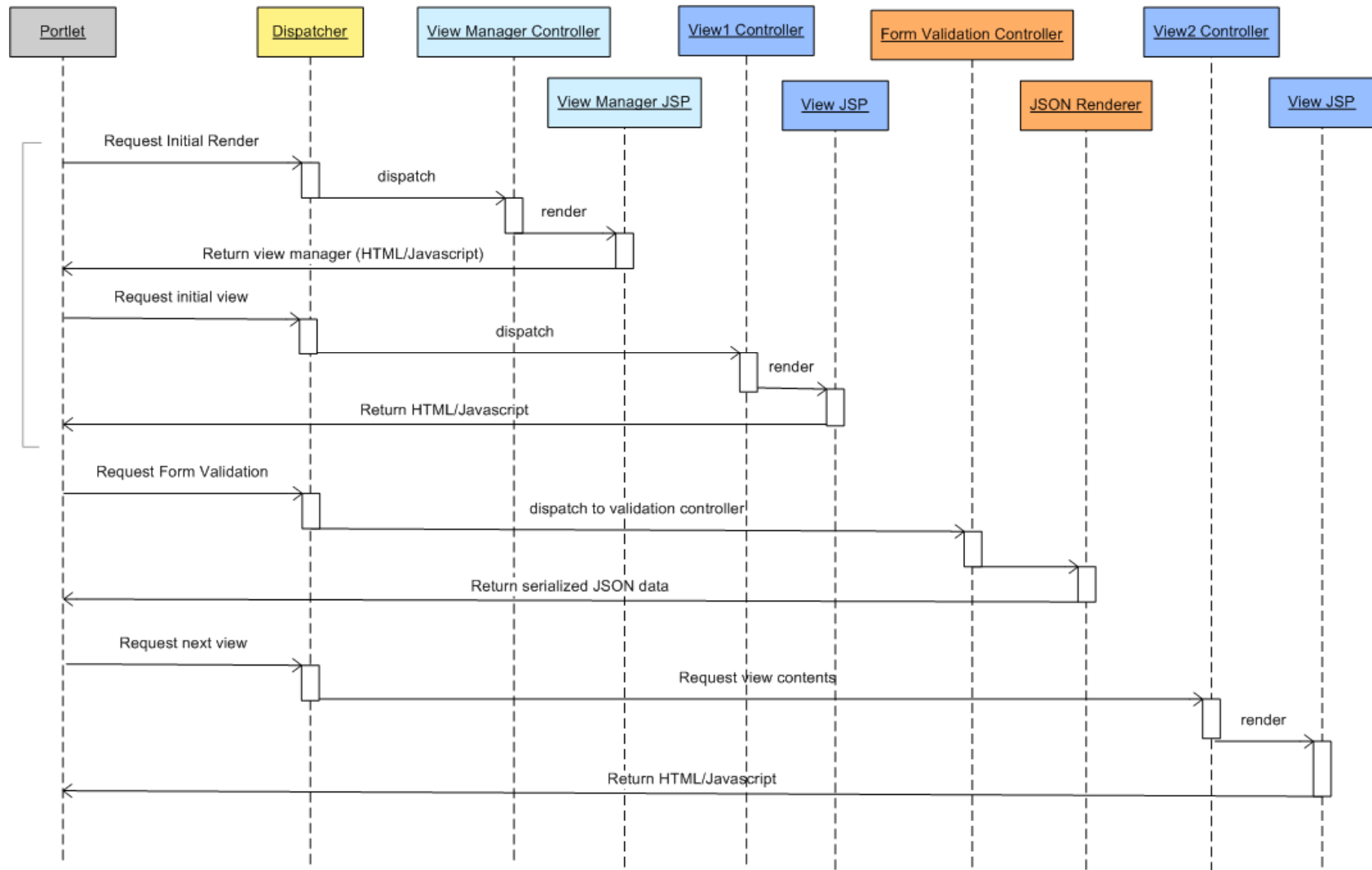
- > Tibco PageBus
 - “Publish-Subscribe” with hierarchies

Portlet View Manager

- > Persist resourceURL
- > Load and manage views
- > Submit/Validate using Ajax
- > History support



View Manager Sequence Diagram



Ajaxifying Portlets Demo

<http://localhost:8480/bluenog>

Conclusion

- > Clean HTML, CSS
- > Tools: JSON/JavaScript
- > JSLoader
- > Build Ajax Request Handlers
- > Add Ajax Widgets to View Templates



JavaOneSM

Thank You

Anas Mughal
anas.mughal@bluenog.com



Demo Source Code available at:
<http://info.bluenog.com/ajax>

Check for updated code and slides at:
<http://blog.anas-mughal.com>

