AJAX and Persistence: Emerging Patterns and Pain Points

Craig Russell

Larry White

Smitha Kangath

Sun Microsystems, Inc.

TS-8614

Goal of This Talk

Explore challenges and opportunities integrating data persistence and AJAX applications

Agenda

AJAX Overview

Frameworks and Communication Patterns

Data Model and Caching

Client/Server Protocol Examples

Server Architectures

Conclusion

AJAX Overview

- Asynchronous JavaScript™ technology and XML
- Web Page == Application
- XMLHTTP Object
  - Request to host
  - Asynchronous reply calls event handler
  - User's JavaScript technology event handling code
    - Analyzes reply
    - Updates DOM, causing partial browser refresh
  - Request and reply content defined by JavaScript technology and server-side component(s)

AJAX Overview

- Benefits
  - More lively applications (no page refresh)
  - Richer presentation by combining multiple data sources
- Tradeoff
  - More complex programming
  - Possible performance issues

AJAX Example: maps.google.com

- Moveable static content
  - Tiled geographic information
    - Satellite images
    - Street maps
- Dynamic content
  - Persistent data looked up on the server (model)
  - Flags identifying points of interest (view)
    - Pizza joints
    - Tattoo parlors
    - Animal rescue centers
  - Detailed descriptions based on user actions (controller)

Key Takeaway—Is there Any Good Mexican Food Near Moscone?

Maps Print Email Link to this page

Results **1-10** of about **37,900** for **Mexican** near **San Francisco, CA**

Categories: Restaurants, Restaurant Mexican

A **Pancho Villa Taqueria**
3071 16th St, San Francisco, CA
(415) 864-8840

B **La Taqueria**
2889 Mission St, San Francisco, CA
(415) 285-7117

C **Papalote Mexican Grill**
3409 24th St, San Francisco, CA
(415) 970-8815

D **Colibri Mexican Bistro**
438 Geary St, San Francisco, CA
(415) 440-2737

E **Maya Restaurant**
303 2nd St, San Francisco, CA
(415) 543-2928

Agenda

AJAX Overview

Frameworks and Communication Patterns

Data Model and Caching

Client/Server Protocol Examples

Server Architectures

Conclusion

Frameworks and Communication Patterns

Scores of frameworks are available

- Client-side
  - Widget libraries
  - DOM manipulation tools
- Server-side
  - JavaServer™ Faces software ManagedBean
  - Servlets, JSP™ technology-based pages, CGI
- Integration with cool sites
  - maps.google.com
  - paypal.com

Frameworks and Communication Patterns

Look for Reusable Components

- No single framework will meet all your requirements
- Choose frameworks that encourage mix/match approach

Communication With Server

- HTTP Protocol Wraps Message
  - HTML
  - XML
  - REST (HTML or XML)
  - JavaScript technology
    - JSON
- RPC
- Web Service (SOAP/WSDL format)

Agenda

AJAX Overview

Frameworks and Communication Patterns

Data Model and Caching

Client/Server Protocol Examples

Server Architectures

Conclusion

Data Model and Caching

- General principles
  - Cached data minimizes server communication
  - DOM can contain arbitrary data
  - Format of data is application-specific

Data Model and Caching

- Cached data minimizes server communication
- Client-side caching solutions
  - Browser cache
  - Home-grown written in JavaScript technology
  - Other solutions

Caching

- Client-side caching solutions
  - Browser cache
    - Works fine except when it doesn't
    - Beware—browser-specific behavior and different user settings cause issues
  - HTTP Expiration Model—cache directives
    - HTML meta tags (this only works when using HTML)
      -
```
<head>
...
<meta http-equiv="Cache-Control"
  content="max-age=1800">   <!-- ½ hour -->
<meta http-equiv="Expires"
  content="Tues, 23 May 2006 1:00:00 GMT">
</head>
```

Caching

- Client-side caching solutions
  - HTTP Expiration Model—cache directives
    - HTTP Headers
      - **HTTP/1.1 200 OK**
        Cache-Control: Public, max-age=1800
        Expires: Tues, 23 May 2006 1:00:00 GMT
        Content-Type: text/html;charset=ILO-8859-1
        ...etc
    - Can be used with XML or other content

Caching

- Client-side caching solutions (browser cache)
  - HTTP Validation Model
    - A unique ticket used for each HTTP response
    - Browser sends ticket with subsequent requests
    - Server can send fresh data or send HTTP 304 (no changes)
  - Difficult to apply to dynamically generated content—servlets, etc.
    - Special coding required

Caching

- Client-side caching solutions
  - Home-grown written in JavaScript technology
    - Global variables—anything you want, as long as it's a JavaScript technology object or a JavaScript technology array

Caching—Other Solutions

- dojo.storage—key/value pairs
  - Stored using various strategies—for now, Flash Storage

```
var results = dojo.storage.get(key);
  // save some value:
  try{
    dojo.storage.put(key, value,
                     saveHandler);
  }catch(exp){
    alert(exp);
  }
```

  - Values—String or “JSON-ified” JavaScript technology object

Caching—Other Solutions

- Modeling complex relationships is possible using XML or JavaScript technology but not as natural as with a fully object-oriented language like Java™ technology
- Java technology (at present) is not widely used on the client-side
- Possible Java technology-based solutions?
  - Applet or Java Web Start software client app used as client-side cache
    - Use with Java technology ↔ JSON transformation
    - Would the community accept/use it?

Data Model

- DOM can contain arbitrary data
  - Each use case in an app tends to require a different sub-set (“view”) of data
  - Difficult using key-value pairs with opaque values
  - Developing equivalent of alternate indices requires complex coding
- Format of data is application-specific
- JSON can make caching easier

General Caching Issues

- Staleness of data
  - Is the data static (unchanging)?
    - No need to refresh
      - e.g., world aerial map geo tiles
    - Beware—even seemingly static data can change
      - Remember the country “Yugoslavia”?

General Caching Issues

- Staleness of data
  - Dynamic data
    - How often does it change?
    - How important (from a business perspective) is it to never have stale data?
      - e.g., an on-line store may sell an “out-of-stock” item if it can be re-stocked in time—or if customers can be otherwise assuaged if problems occur

General Caching Issues

- Size of data
  - Even static data may be too large to reasonably cache on the client
    - e.g., all the geo aerial tiles in the world (at all zoom scales)
  - Server-side caching is an option
    - Performance trade-off

General Caching Issues

- Caching optimization techniques
  - Selective
    - Cache based on user actions—e.g., selections
  - Lazy-loading
    - Defer loading until data requested
  - Pre-fetching
    - Make inferences to pre-fetch required data before it is requested
      - Single user history—e.g., “you generally browse comedy movies”
      - Or user community history—e.g., “people who rented this movie tend to like Mexican food too”

General Caching Issues

- Caching optimization techniques
  - Techniques may be combined
  - Trade-off between
    - General-use/less-optimizable caching
    - Do-it-yourself customized caching

Agenda

AJAX Overview

Frameworks and Communication Patterns

Data Model and Caching

Client/Server Protocol Examples

Server Architectures

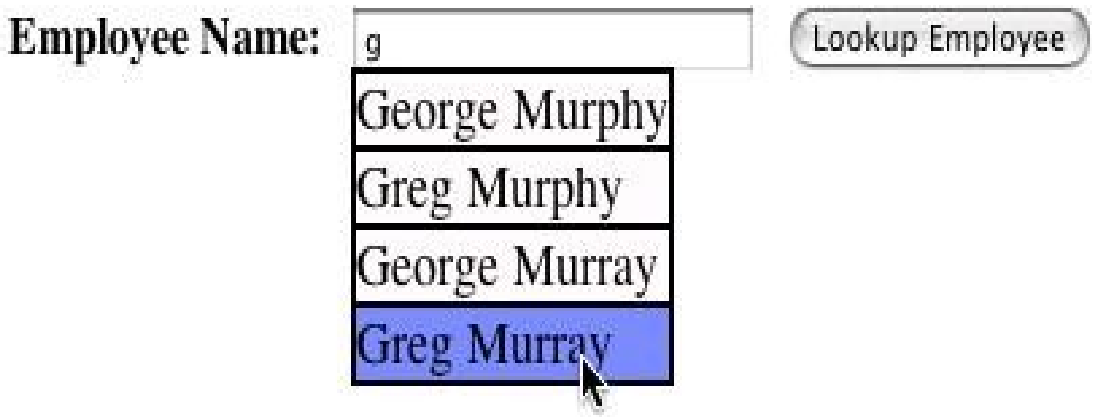
Conclusion

Client/Server Protocol Examples

- Different formats—XML, HTML, JSON, etc.
- Data Model can be independent of the format
- A helper class or a filter can format the data

Follow the MVC approach—don't mix formatting logic in your persistent entities

Example: Auto-Completion



Example: Auto-Completion

- One of the most popular patterns
- Basic idea: Combo box (text plus drop-down)
  - User types, script provides suggestions
  - Keystrokes may send async message to server
  - Message header has partial field content
  - Server decides what to send back (heuristics)

Watch out for too much server interaction

Wait while user types

Consider submitting many fields at once

Example: ValueList Handler

- Another popular pattern
- Basic idea: Scroll box
  - Output portion of query result
    - Fetch and return first “N” rows
  - Get more rows while user thinks

Watch out for too much server interaction

Wait until user gets near the end of the list

Fetch several rows at once

Java Pet Store Demo

XML vs. JSON

- JSON is a subset of the JavaScript language
- With JSON, the server returns text which is converted to a JavaScript language object (or array of objects)
  - No need to parse the XML returned by the server
  - eval() returns the corresponding JavaScript language object[s] from the String representation
  - What the server sends back can modify globals

XML Format

```
<items>
    <item>
        <id>10934</id>
        <name>Red Lobster</name>
        <imgURL>images/redlobster.gif</imgURL>
    </item>
    <item>
        <id>62903</id>
        <name>Lichen</name>
        <imgURL>images/lichen.gif</imgURL>
</items>
```

Server Side Query

```
// Java technology Persistence Application Programming Interface
List<Item> getItems(String query, int first, int max) {
  Query q = em.createQuery(
    "SELECT NEW Item(i.id, i.name, i.imgURL)
      FROM PItem AS i" + query);
  q.setFirstResult(first).setMaxResults(max);
  return (List<Item>)q.getResultList();
}
// Java technology Data Objects
List<Item> getItems(String query, int first, int max) {
  Query q = pm.newQuery(
    "SELECT INTO Item(id, name, imgURL)
      FROM PItem " + query);
  q.setRange(first, first + max);
  return (List<Item>)q.execute();
}
```

Server Formats Data as XML

```
StringBuffer sb = new StringBuffer("<items>\n");
//call the facade that accesses persistent entities
List items = facade.getItems(query, start, number);
Iterator<Item> it = items.iterator();
while (it.hasNext()) {
  Item c = it.next();
  sb.append("<item>\n");
  sb.append("<id>" + c.getCategoryID() + "</id>\n");
  sb.append("<name>" + c.getName() + "</name>\n");
  sb.append("<imgURL>" +c.getImageURL() +
  "</imgURL>\n");
  sb.append("</item>\n"); }
sb.append("</items>\n");
...
return sb.toString();
```

Client Interprets Data as XML

```
function parseCategories(asyncReq) {
  var items = asyncReq.responseXML.getElementsByTagName
    ("items")[0];
  for (i = 0; i<items.childNodes.length; ++i) {
    var xitem = items.getElementsByTagName
      ("item")[i];
    var item = new Item();
    item.id = xcategory.getElementsByTagName("id")[0].
      firstChild.nodeValue;
    item.name =
xcategory.getElementsByTagName("name")[0].
      firstChild.nodeValue;
    item.desc =
xcategory.getElementsByTagName("imgURL")[0].
      firstChild.nodeValue;
    appendItem(item);
  }
}
```

JSON Format

```
[
  {
    “id”:”10934”,
    “name”:”Red Lobster”,
    “imgURL”:”images/redlobster.gif”
  },
  {
    “id”:”62903”,
    “name”:”Lichen”,
    “imgURL”:”images/lichen.gif”
  }
]
```

Server Formats Data as JSON

```
StringBuffer sb = new StringBuffer("[\n");
//call the facade that accesses persistent entities
List items = facade.getItems(query, start, number);
Iterator<Item> it = items.iterator();
while (it.hasNext()) {
   Item c = it.next();
   sb.append("{\"id\":\"" + c.getItemID()+"\",");
   sb.append("\"name\":\"" + c.getName() + "\",");
   sb.append("\"imgURL\":\"" +
   c.getImgURL() + "\"");
   sb.append("]}");
   if (it.hasNext()) {
   sb.append(",\n");}
}//end while loop
sb.append("\n]"); return sb.toString();
```

Client Interprets Data as JSON

```
function parseItems(asyncReq) {
  var items = eval(asyncReq.responseText);
  for (i = 0; i<items.length; ++i) {
    appendItem(items[i]);
  }
}
```

Agenda

AJAX Overview

Frameworks and Communication Patterns

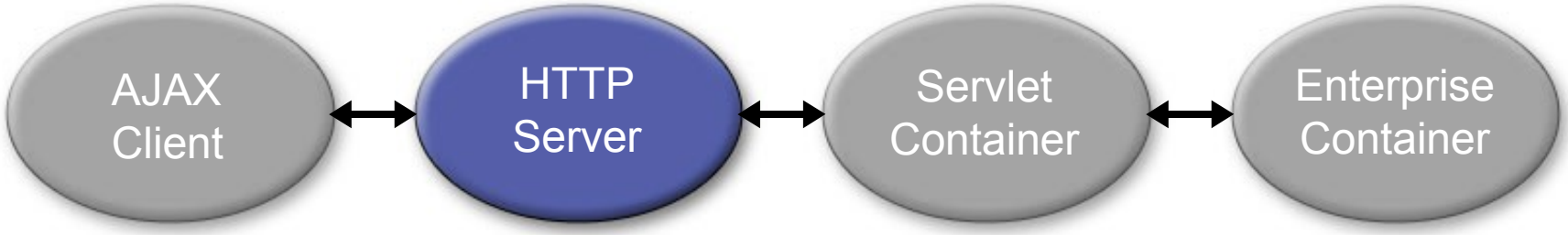
Data Model and Caching

Client/Server Protocol Examples

Server Architectures

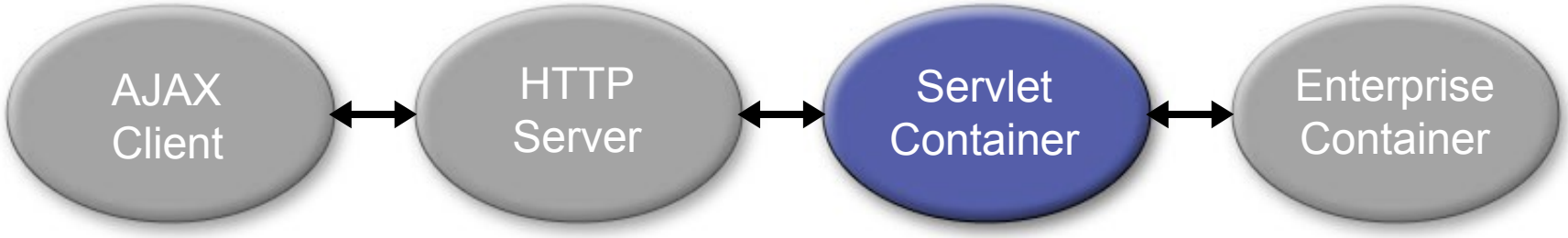
Conclusion

Server Architectures



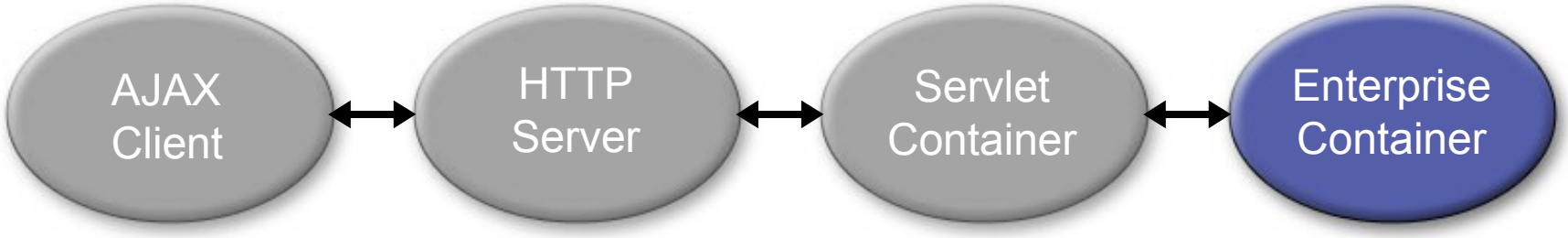
- HTTP server without Java technology (e.g., LAMP)
  - Static content
  - Dynamic content
    - CGI scripts
    - SQL access
    - PHP, Perl, Python, etc.

Server Architectures



- Servlet container (e.g., Tomcat)
  - Dynamic content
    - Servlets
    - SQL access via JDBC™ software
    - Java Data Objects (JDO), Hibernate, TopLink, Entity JavaBeans™ architecture
    - JavaServer Faces technology

Server Architectures



- Enterprise container
  - Dynamic content
    - Session JavaBeans architecture
    - JDO, Hibernate, TopLink, Entity JavaBeans architecture
    - Distributed transactions
    - WSDL/SOAP messages

Active Record vs. Data Mapper

Two of Many Patterns for Persistence

- Active record
  - Mixes domain object with persistence
  - e.g., Ruby on Rails
- Data mapper
  - Separates domain object from persistence
  - Mapping to datastore is separated
  - e.g., JDO, Entity Beans, Hibernate, TopLink

Agenda

AJAX Overview
Frameworks and Communication Patterns
Client/Server Protocol Examples
Data Model and Caching
Server Architectures
Conclusion

Conclusion

- AJAX enables more lively web applications
- Mix and match standard components
- Minimize server interactions
  - Use caching on client and server
  - Use predictive fetching
  - Aggregate server requests

Q&A

https://glassfish.dev.java.net/

http://java.sun.com/blueprints/ajax.html

http://developers.sun.com/ajax

AJAX and Persistence: Emerging Patterns and Pain Points

Craig Russell

Larry White

Smitha Kangath

Sun Microsystems, Inc.

TS-8614