BIBSYS Ask – the architecture

Hans Olaf Gundersen, BIBSYS
BIBSYS Ask - the architecture

• Federated search client
• Built for the Internet
• Focus: easy access to the content
• Aspect oriented search functionality
BIBSYS Ask - the architecture

• Uses key technologies:
  – Domain model
  – RESTful web services
  – JSP (JAVA server pages)
  – Javascript and AJAX client side in users WEB-browser
Domain Model

- Identifies the relationships among all major entities within the system.
- Consists of objects with reasonable responsibilities - the domain objects.
- Must be able to complete all tasks for the applications using them.
- The naming of the objects and usage of them should correspond with your business to make it understandable and easy to use and avoid confusion.

RESTful web services
Also called a RESTful web API

• Simple web service implemented using HTTP and the principles of REST

• Collection of resources, with three defined aspects:
  – the base URI for the web service, such as http://example.com/resources/
  – the MIME type of the data supported by the web service. This is often JSON, XML or YAML but can be any other valid MIME type.
  – the set of operations supported by the web service using HTTP methods (e.g., POST, GET, PUT or DELETE).

For more information, see: http://en.wikipedia.org/wiki/Restful_web_service
//This is a super simple example of a JSON Service
//Defining an inner class to populate with data
class Car {
    private String color="None";
    private int numberOfWheels=4;
    private String brand="Unknown";
    public Car(String _color,int _numberOfWheels,String _brand) {
        color = _color; numberOfWheels = _numberOfWheels; brand = _brand;
    }
}

//Creating a list to put the "Car"-objects into
ArrayList carList = new ArrayList();
//Adding "Car"-objects
carList.add(new Car("Red",6,"Ford"); carList.add(new Car("Yellow",4,"Volvo");
carList.add(new Car("Blue",4,"Toyota");

//Using XStream to automatically serializing
XStream xstream = new XStream(new JsonHierarchicalStreamDriver());
String json = (xstream.toXML(carList));

//Setting proper content type and encoding before responding...
response.setContentType("application/json;charset=utf-8");
<%=json%>
"list": [
  {
    "color": "Red",
    "numberOfWheels": 6,
    "brand": "Ford",
    "outer-class": {}
  },
  {
    "color": "Yellow",
    "numberOfWheels": 4,
    "brand": "Volvo",
    "outer-class": {
      "@reference": "../../org.apache.jsp.json.example_005fservice_jsp$1$Car/outer-class"
    }
  },
  {
    "color": "Blue",
    "numberOfWheels": 4,
    "brand": "Toyota",
    "outer-class": {
      "@reference": "../../org.apache.jsp.json.example_005fservice_jsp$1$Car/outer-class"
    }
  }
]

<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  
  function init(_event) {
    var target = $('target');
    target.appendChild(new Element('li').update("Sending request..."));

    new Ajax.Updater(target, '/ask2/json/example_service.jsp', {
      evalJSON: true,
      method: 'get',
      insertion: function(_target, _response) {
        //Clearing status message...
        _target.update('');

        //Parsing the response
        var dataList = _response.evalJSON().list;
        for (var i=0, data; data = dataList[i]; i++) {
          var resultItem = new Element('li').update("A " + data.color + " " + data.brand + " with " + data.numberOfWheels + " wheels;");
          _target.appendChild(resultItem);
        }
    });
  }
</script>
</head>
</html>
Client code running in users web browser – using AJAX
A Red Ford with 6 wheels;
A Yellow Volvo with 4 wheels;
A Blue Toyota with 4 wheels;
A search for «java» in BIBSYS Ask

<table>
<thead>
<tr>
<th>Title</th>
<th>JAVA developer’s journal</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISSN</td>
<td>1998-3696</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Title</th>
<th>Server-side GPS and assisted-GPS in Java</th>
</tr>
</thead>
<tbody>
<tr>
<td>Author</td>
<td>Neil Harper</td>
</tr>
<tr>
<td>Year</td>
<td>2010</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Title</th>
<th>Programming Finite Elements in Java</th>
</tr>
</thead>
<tbody>
<tr>
<td>Author</td>
<td>Gennady Nikishkov</td>
</tr>
<tr>
<td>Year</td>
<td>2010</td>
</tr>
<tr>
<td>ISBN</td>
<td>9781488829725</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Title</th>
<th>Operating System Concepts with Java</th>
</tr>
</thead>
<tbody>
<tr>
<td>Author</td>
<td>Abraham Silberschatz</td>
</tr>
<tr>
<td>Year</td>
<td>2010</td>
</tr>
<tr>
<td>ISBN</td>
<td>9780470390791</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Title</th>
<th>JavaServer faces 2.0, the complete reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Author</td>
<td>Ed Burns, Chris Schalk, Neil Griffin</td>
</tr>
<tr>
<td>Year</td>
<td>2010</td>
</tr>
</tbody>
</table>
Narrowing the search using the scan boxes
Search in local and remote databases on the net
A Search with firebug enabled
<table>
<thead>
<tr>
<th>Status</th>
<th>Domain</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>ab.bibsys.no</td>
<td>6.6KB</td>
</tr>
<tr>
<td>200</td>
<td>ab.bibsys.no</td>
<td>9KB</td>
</tr>
<tr>
<td>200</td>
<td>ab.bibsys.no</td>
<td>24.5KB</td>
</tr>
<tr>
<td>200</td>
<td>ab.bibsys.no</td>
<td>27.6KB</td>
</tr>
<tr>
<td>300</td>
<td>ab.bibsys.no</td>
<td>1.0KB</td>
</tr>
<tr>
<td>200</td>
<td>ab.bibsys.no</td>
<td>15.4KB</td>
</tr>
<tr>
<td>200</td>
<td>ab.bibsys.no</td>
<td>15KB</td>
</tr>
</tbody>
</table>

**Response Headers**

**Date:** Fri, 22 Apr 2016 13:08:47 GMT

**Server:** Apache/2.4.13 (Ubuntu)

**Content-Length:** 2898

**Connection:** Keep-Alive

**Content-Type:** application/json; charset=UTF-8

**Cache-Control:** max-age=43200

**Host:** ab.bibsys.no

**User-Agent:** Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/50.0.2661.75 Safari/537.36

**Accept-Language:** en,zh,zh-CN,zh-TW;q=0.5

**Accept:** application/json, text/plain, */*

**Accept-Encoding:** gzip, deflate

**Connection:** keep-alive

**2-Request-URI:** /xml/epub

**3-Request-Type:** 1.1

**Referrer:** https://ab.bibsys.no/xml/epub

**Code:** 200

**Status:** 200 OK

**Domain:** ab.bibsys.no

**Size:** 0.0 KB
### Response from server

```
<table>
<thead>
<tr>
<th>GET</th>
<th>status</th>
<th>url</th>
<th>response_time</th>
</tr>
</thead>
<tbody>
<tr>
<td>/api</td>
<td>200</td>
<td>/api/services/cqlserver契</td>
<td>24.518</td>
</tr>
<tr>
<td>/api</td>
<td>200</td>
<td>/api/services/cqlserver契</td>
<td>27.640</td>
</tr>
<tr>
<td>/api</td>
<td>200</td>
<td>/api/services/cqlserver契</td>
<td>120</td>
</tr>
<tr>
<td>/api</td>
<td>200</td>
<td>/api/services/cqlserver契</td>
<td>13.419</td>
</tr>
</tbody>
</table>
```

```javascript
Object { documents: more... }
[ Object { recordId: "s1234567", more... }, Object { recordId: "s1556567", more... }, Object { recordId: "s123567", more... }, 7more... ]
Object { recordId: "s123567", more... }
```

```
```

```json
{
  "id": "123456789",
  "title": "Java Programming",
  "author": "John Doe",
  "year": 2020,
  "publisher": "O'Reilly Media",
  "isbn": "978-1491901882",
  "language": "en",
  "type": "book",
  "description": "A comprehensive guide to learning Java programming.",
  "tags": ["Java", "Programming", "Computer Science"],
  "publicationDate": "2020-01-01",
  "availability": "available",
  "language": "en"
}
```
Autocomplete in action
Advantages

– Services are not tied to one application. Services are targeted for one goal and are reused.

– It is easy to add services

– Layers below the RESTful webservices are independent of the database server

– JSON (JavaScriptObjectNotation) streams are simple and Javascript understands JSON native.
Disadvantages

- Webbrowser might tell you «mixed content» when your application mix secure and unsecure connections.

- Some webbrowser have slow performance or/and bad implementation of javascript.
Questions?

We would love to tell you more about BIBSYS Ask at our stand no 7 at the Emtacl10 exhibition