

Web Applications Overview

Daniel Silva

daniel.silva@platinumsolutions.com

Principal, Platinum Solutions

Presenter

- Currently a Principal at Platinum Solutions, a local consultancy firm
- Computer programmer since 1996 focused on distributed programming
- Worked in a variety of industries, such as tourism, food service and defense/intel
- Plays World of Warcraft

Web Application Definition

- A web application is a computer program that exposes its resources through unique Uniform Resource Identifiers (URIs) through the Hypertext Transfer Protocol (HTTP)

Thank you for coming
;)

Web Applications

- Parts of a web application
- Example web application
- Implications of a web application
- How web sites compare to web apps
- Clients for a web application
- Amazon S3 walkthrough

Web Application Definition

- A web application is a computer program that exposes its resources through unique Uniform Resource Identifiers (URIs) through the Hypertext Transfer Protocol (HTTP).
- Regardless of the technology being used, this is always the case.

HTTP

- Hypertext Transfer Protocol
- Communication protocol designed to work metaphorically the way physical letters and envelopes work within a particular network
- Supports any kind of data in its envelopes
- Envelopes can be empty

URI

- Uniform resource identifiers
- Specification for the *address* you put on an HTTP envelope before sending it
- Uniquely names a resource
- <http://www.cnn.com> is the URI for the top-level web page of the CNN web site.

Example: Greeter Application

Greeter Class

```
1 public class Greeter {
2     public Greeter() { }
3     public String getGreeting(String name) {
4         if (name == null || name.length() == 0) {
5             name = "world";
6         }
7         return "Hello, " + name;
8     }
9 }
```

Example: Greeter Application

Greeter Application (non-web)

```
1 public class GreeterApplication {
2     public static void main(String[] args) {
3         Greeter greeter = new Greeter();
4         if (args.length > 0) {
5             System.out.println(greeter.getGreeting(args[0]));
6         } else {
7             System.out.println(greeter.getGreeting(null));
8         }
9     }
10 }
```

Example: Greeter Application

Greeter Application (Web-ified)

```
1 import java.io.PrintWriter;
2 import java.io.IOException;
3 import javax.servlet.ServletException;
4 import javax.servlet.http.HttpServlet;
5 import javax.servlet.http.HttpServletRequest;
6 import javax.servlet.http.HttpServletResponse;
7 public class GreeterServlet extends HttpServlet {
```

Example: Greeter Application

Greeter Application (Web-ified)

```
8     public void doGet(HttpServletRequest req, HttpServletResponse res)
9         throws ServletException {
10        Greeter greeter = new Greeter();
11        String name = req.getParameter("name");
12        res.setContentType("text/xhtml");
```

Example: Greeter Application

Greeter Application (Web-ified)

```
13     try {
14         PrintWriter out = res.getWriter();
15         out.write("<!DOCTYPE html\n");
16         out.write("PUBLIC \"/-//W3C//DTD XHTML 1.0 Strict//EN\n");
17         out.write("\nhttp://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd\n");
18         out.write("<html\n");
19         out.write("<head\n");
20         out.write("    <title>Greeting Representation in XHTML</title\n");
21         out.write("</head\n");
22         out.write("<body\n");
23         out.write("    <p>" + greeter.getGreeting(name) + "</p\n");
24         out.write("</body\n");
25         out.write("</html\n");
32         out.flush();
33     } catch (IOException e) {
34         throw new ServletException(e);
35     }
36 }
37 }
```

Example: Greeter Application

Greeter Application (Web-ified)

```
1 <servlet>
2     <servlet-name>GreeterServlet</servlet-name>
3     <servlet-class>GreeterServlet</servlet-class>
4 </servlet>
5 <servlet-mapping>
6     <servlet-name>GreeterServlet</servlet-name>
7     <url-pattern>/greeting</url-pattern>
8 </servlet-mapping>
```

This is a J2EE-spec web.xml snippet... all J2EE web applications must have a web.xml configuration file that maps URIs to Servlets, among other configuration details.

- Requests to /greeting would be processed by the GreeterServlet
- <http://localhost/mywebapp/greeting?name=Daniel>

Key Parts in Example

- `getGreeting(String)` exposed through `/greeting` URI
- Available over HTTP through a J2EE container
- Anything less, and it's not a web application
- Anything more, it continues to be a web application

What is so special
about web applications?

Distributed Computing

- Web applications are distributed computing (client/server) applications
- Functions can be broken up into smaller functions, then given their own server
- HTTP and URI specifications address communication and addressability issues

Scalability

- Multiple instances on multiple servers
- Load balancing appliances for making use of this architecture
- Load balancers can *also* be replicated
- It's how sites like amazon.com and ebay.com provide the services they do, at the level they do

Consumers

- Typical computer program is predictable in its execution; web applications are not
- Web applications are available to *any program* that has HTTP support
- Data served may be used in conjunction with data from other applications to create a new kind of service

Example: Google Reader

- Google Reader (<http://goggle.com/reader>) is an RSS aggregator (web) application
- RSS is is an XML format that describes publications like blog entries or podcasts
- Web applications that provide an RSS representation of their data can be plugged into Google Reader

Accessibility

- HTTP libraries in most computing languages for writing custom clients
- Instant clients in two very popular web browsers in the market (for (X)HTML data)
- Freedom and energy of the World Wide Web at your beck and call

Web Site vs. Web Application: What's the diff?

- There is none; a web site *is* a web application
- Static resources are still resources bound to URIs and accessed over HTTP
- Always involve some application listening on a port for HTTP requests
- The leap to processing user input is small

Human Web

- The web is full of URIs that can be accessed by ordinary people through a web browser
- Examples include web sites like CNN.com, Amazon.com, and Ebay.com
- Non-browser client apps exist that consume web apps, but are ultimately available through a browser

Human Programmable Web

- Beyond the Human Web; web applications meant solely for other applications to use
- GreeterServlet could return custom XML (instead of XHTML)
- Web apps could return *any* data structure
- Imagine a Java Byte Stream in an HTTP envelope; this is possible

Web Services

- Programmable web similar to Web Services but isn't the same
- “Web Services” is a very broad term; covers technologies not tied to the web
- The “Web Services Stack” is a set of specification for developing web applications; also covers non-web applications

Web Services Example

- SOAP is a communication protocol for exchanging XML (only) messages
- An envelope format, similar to HTTP
- Doesn't require HTTP or use of URIs for naming resources (only service endpoints)
- SOAP is always a valid "Web Service" regardless of use of HTTP/URIs
- SOAP popular for RPC-style architecture

SOAP was once an acronym for Simple Object Access Protocol and Service Oriented Architecture Protocol, but now it's just SOAP to avoid confusion

Programmable Web Example

- Data needs to be shared in a particular way
- Database view possible
- Web application that exposes data at particular URIs over HTTP solves the same problem, but with more benefit
- Elegant solutions exist for issues like Security, Data Integrity and Validation, etc.

Amazon S3 (Simple Storage Service)

- An online storage web application service offered by Amazon
- Offers virtually unlimited storage through a web application programming interface
- Uses the same data storage and infrastructure the massive online store uses

Amazon S3 (Simple Storage Service)

- Amazon provides a client application to interact with S3, uses HTTP
- Two kinds of objects: Bucket and Object
- Buckets are uniquely named collections of objects
- Objects are data structures, in any format, along with a name and some meta-data

Amazon S3 (Simple Storage Service)

- Buckets can be accessed at <http://s3.amazonaws.com/{bucket-name}>
- Objects can be accessed at [http://s3.amazonaws.com/{bucket-name}/
{object-name}](http://s3.amazonaws.com/{bucket-name}/{object-name})
- Object names can have the '/' character in its name to simulate hierarchy

Amazon S3 (Simple Storage Service)

- S3 is a working example of how powerful the programmable web can be
- Amazon is able to expose its data infrastructure for anybody to use
- Subscribes to simple HTTP for communication and URIs to name its resources

Web Apps Summary

- Use HTTP and URIs at the core
- Scalable, accessible and open
- Web sites are web apps, too
- Browsers drive the Human Web (i.e., Google Reader)
- Custom clients drive the Programmable Web (i.e., Amazon S3)

Web Applications Overview

Daniel Silva

daniel.silva@platinumsolutions.com

Principal, Platinum Solutions

Thank you for coming
=)