# Web Application Development

Produced by

David Drohan (ddrohan@wit.ie)

Department of Computing & Mathematics
Waterford Institute of Technology

http://www.wit.ie

Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

Introduction to

# WEB SERVICE APIs

ACCESSIBLE WEB APPLICATION COMPONENTS

# Agenda

Background – Web Basics

Overview of Web Service APIs (What/How & Why)

Working with Web Service APIs

Creating JavaScript-Based Web Service APIs

Deploying and testing Web Service APIs

# Agenda

Background – Web Basics

Overview of Web Service APIs (What/How & Why)

Working with Web Service APIs

Creating JavaScript-Based Web Service APIs

Deploying and testing Web Service APIs

# WEB BASICS

THE BASIC OPERATIONS OF THE WEB

# Web Basics: Simple Set of Operations, via the HTTP API

HTTP provides a simple set of operations. Amazingly, all Web exchanges are done using this simple HTTP API:

- GET = "give me some info" (Retrieve)
- POST = "here's some new info" (Create)
- PUT = "here's some update info" (Update)
- DELETE = "delete some info" (Delete)

The HTTP API is CRUD (Create, Retrieve/Read, Update, and Delete)

# Web Basics: Protocols

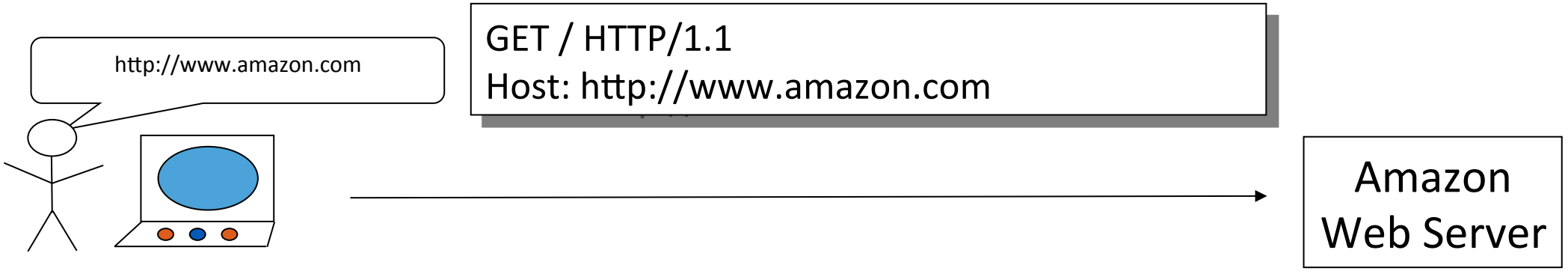
A protocol is a standardized way for a Client and the Web Server to interact.

- Examples of protocols : SOAP, HTTP using REST etc.

At the very least this protocol has to provide

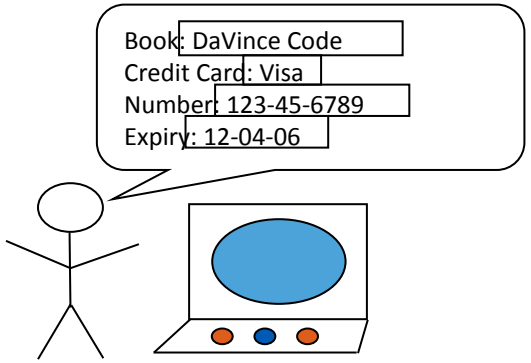
- a way for the client to request the provision of a service
- a way to represent the types of data sent and received

# Web Basics:  Retrieving Info using *HTTP GET*




- The user types in at his browser: http://www.amazon.com
- The browser software creates a **HTTP header** (no **Payload**)
    - The **HTTP header** identifies:
        - The desired action: GET ("get me a resource")
        - The target machine (www.amazon.com)

# Web Basics: Adding Info using *HTTP POST*



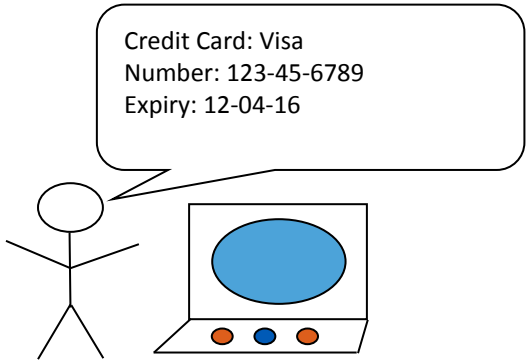

POST / HTTP/1.1
Host: http://www.amazon.com

Book: Da Vince Code
Credit Card: Visa
Number: 123-45-6789
Expiry: 12-04-06

Amazon Web Server

- The user fills in the Web page's form
- The browser software creates an HTTP header with a payload comprised of the form data
  - The **HTTP Header** identifies:
    - The desired action: POST ("here's some new info")
    - The target machine (amazon.com)
  - The **Payload** contains:
    - The data being POSTed (the form data)

# Web Basics: Updating Info using *HTTP PUT*




PUT / HTTP/1.1
Host: http://www.amazon.com

Credit Card: Visa
Number: 123-45-6789
Expiry: 12-04-16

Amazon Web Server

- The user fills in the Web page's form new expiry date
- The browser software creates an HTTP header with a payload comprised of the form data
  - The **HTTP Header** identifies:
    - The desired action: PUT("here's some update info")
    - The target machine (amazon.com)
  - The **Payload** contains:
    - The data being PUTed (the form data, with the new expiry date)

# Web Basics:  Deleting Info using ***HTTP DELETE***


- The (logged in) user selects to delete his/her card
- The browser software creates a **HTTP header** (no **Payload**)
    - The HTTP header identifies:
        - The desired action: DELETE (”delete my resource")
        - The target machine (www.amazon.com)
        - The specific resource (usually an  ‘id’ parameter)

Introduction to

# WEB SERVICE APIs

ACCESSIBLE WEB APPLICATION COMPONENTS

# What are Web Service APIs? (Short Answer)

A Web Service API is a resource that

- Is accessed via HTTP (or HTTPS)
- Returns XML (generaly) : SOAP (formally) / REST (informally)

A simple definition:

"a Web Service API is an application component accessible over open protocols".

# What are Web Service APIs? (Long Answer W3C)

A Web Service API is a software system

- identified by a URI
- whose public interfaces and bindings are defined and described using XML.
- Its definition can be discovered by other software systems.
  - These systems may then interact with the API in a manner prescribed by its definition, using XML based messages conveyed by Internet protocols.

- *Definition from http://www.w3.org/TR/wsa-reqs/#id2604831*

# What are Web Service APIs ?

A way of allowing applications to interact over the web. When using these APIs you will not be creating actual end-user apps but rather 'instruments' for others to use in their apps.

Examples of Web Service APIs in action:

- *Twitter* offers tweeting as a web service api such that other developers can include tweeting into their apps
- *Amazon* offers a lot of functionality as web service apis e.g. S3 (Simple Storage Service)

# Web Apps vs. Web Services

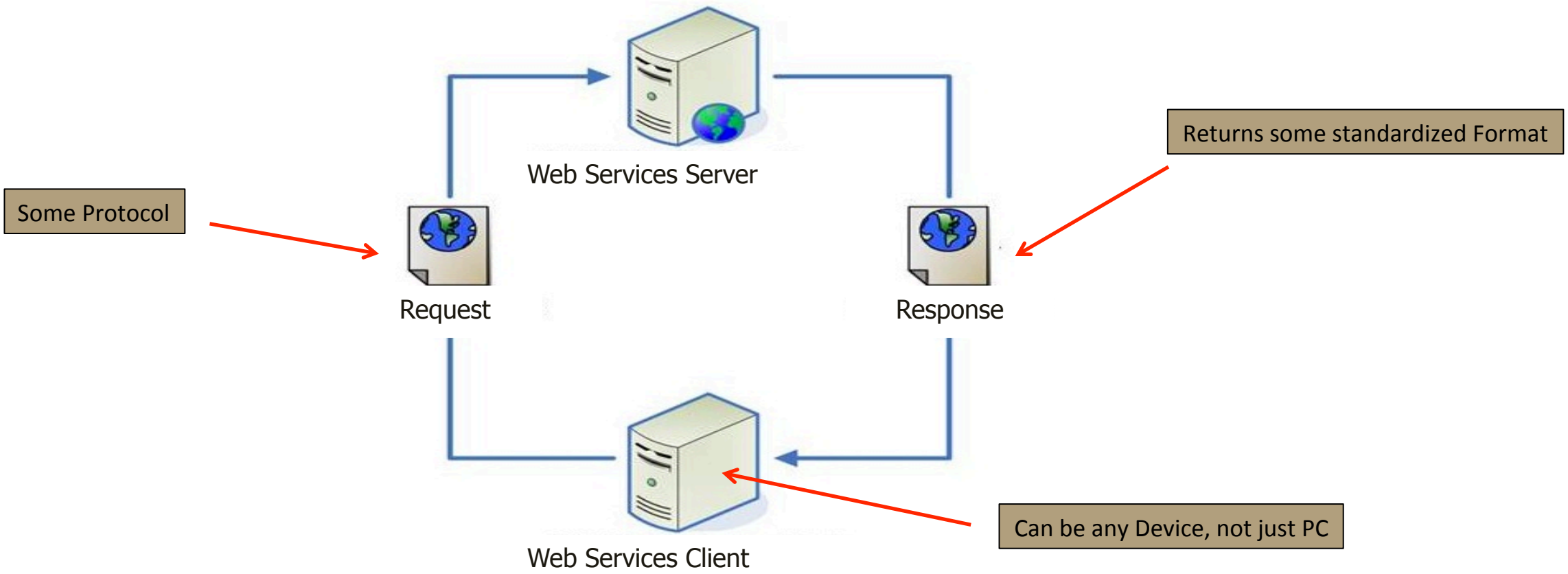
## Web Apps

- Return HTML
- Take GET or POST data as input
- Result intended for a human (via a browser)
- Informal (at best) description of data that resource accepts and result that resource returns

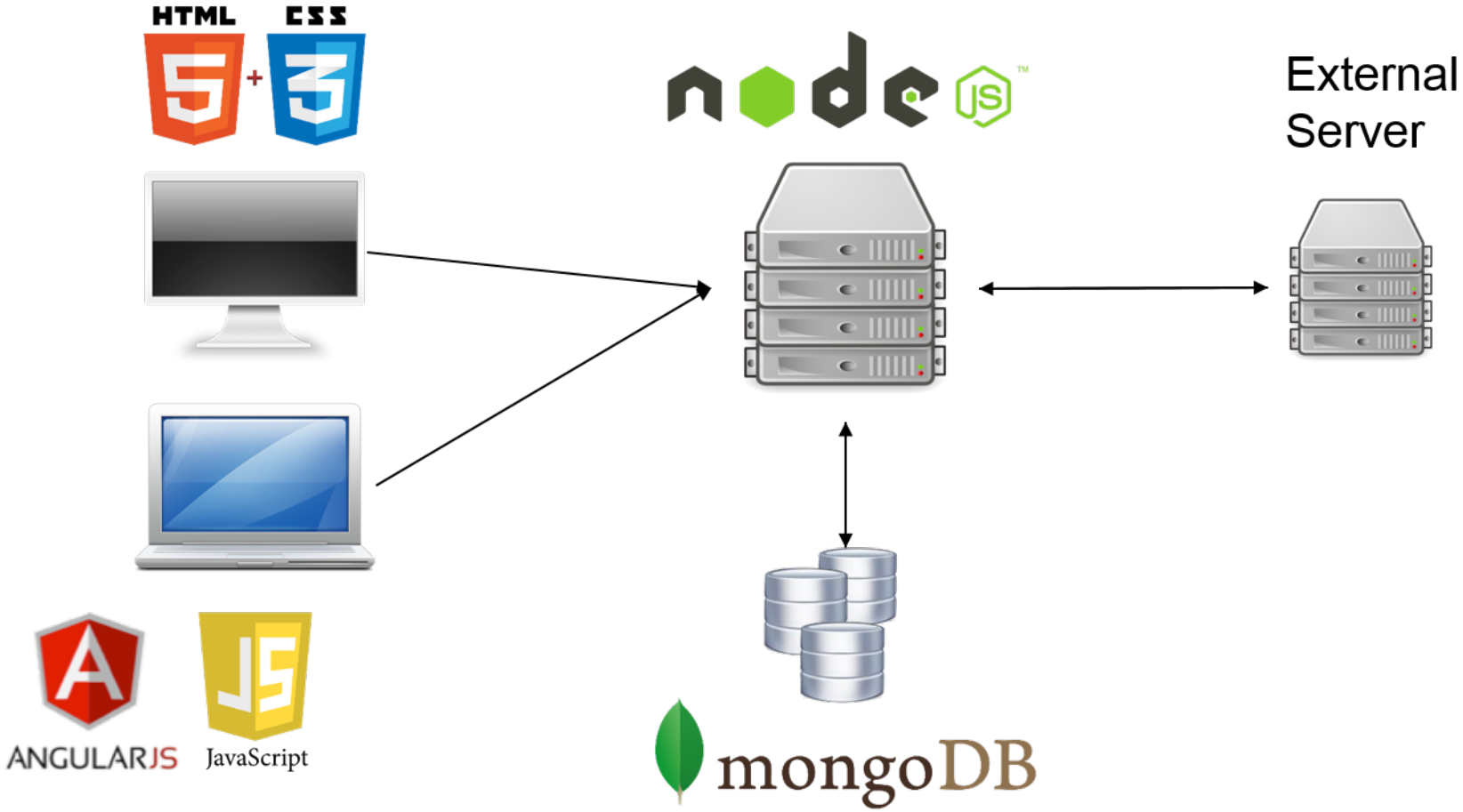
## Web Services

- Return XML
- Result intended for a program
- Formal definition of data that resource accepts and result that resource returns

# How do Web Services work ?



# How our Web Services work…



# Why Web Services ?

The client can use whatever programming language he/she pleases as long as the resulting code respects the protocol.

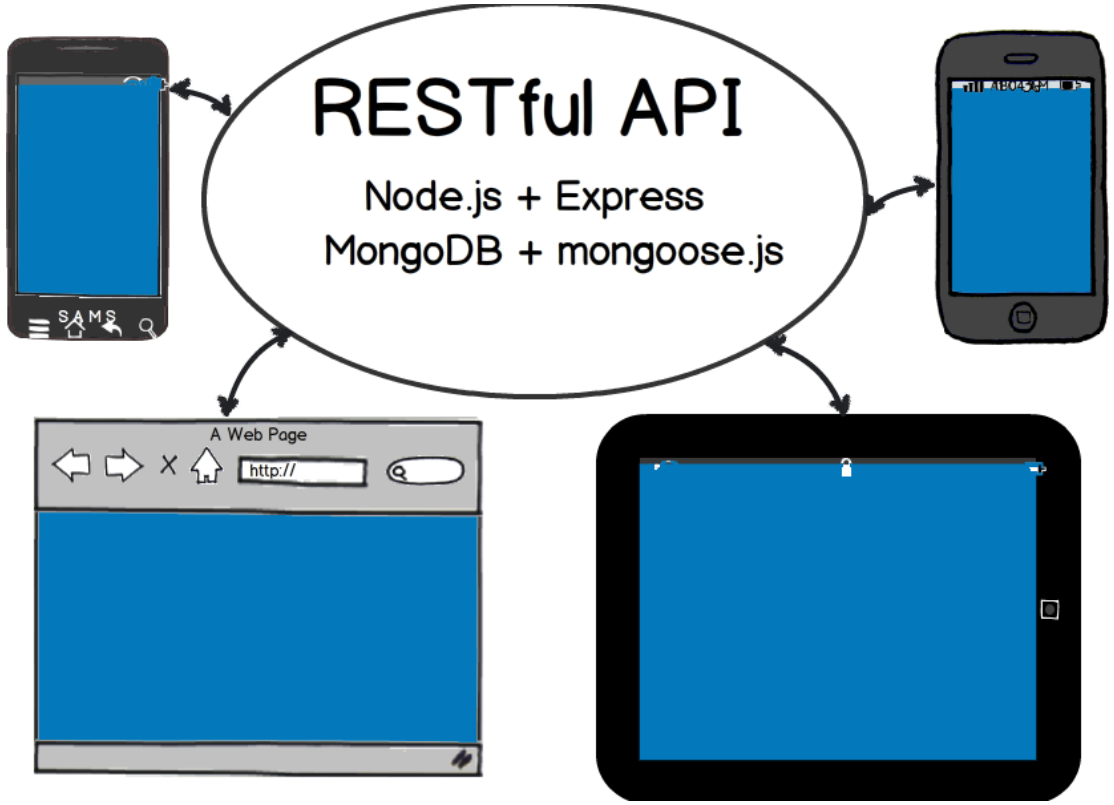
- Consequence: It is easy to interface applications using web services since little or no code adaptation is necessary

Use everything out there that's free to construct your own personalized mash-up application.

- Consequence: No need to reinvent the wheel.

An architecture composed of a collection of web services that communicate with each other is called a Service Oriented Architecture (SOA)

# REST




THE ARCHITECTURAL STYLE OF THE WEB

# What is REST?

"*REST* " was coined by Roy Fielding
in his Ph.D. dissertation [1] to describe a *design pattern* for implementing networked systems.

[1] http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm

# Why is it called "**Re**presentational **S**tate **T**ransfer"?


- The Client references a Web resource using a URL.
- A **representation** of the resource is returned (in this case as an HTML document).
- The representation (*e.g.,* Boeing747.html) places the client in a new **state**.
- When the client selects a hyperlink in Boeing747.html, it accesses another resource.
- The new representation places the client application into yet another state.
- Thus, the client application **transfers** state with each resource representation.

# REST Characteristics

REST is not a standard (unlike SOAP)

- You will not see the W3C putting out a REST specification.
- You will not see IBM or Microsoft or Sun selling a REST developer's toolkit.

REST is just a **design pattern**

- You can't bottle up a pattern.
- You can only understand it and design your Web services to it.

REST does prescribe the **use** of standards:

- HTTP
- URL
- XML/HTML/GIF/JPEG/*etc.* **(Resource Representations)**
- text/xml, text/html, image/gif, image/jpeg, *etc.* (Resource Types, MIME Types)

# REST Principles

Everything is a resource

Every resource is identified by a unique identifier

Use simple and uniform interfaces

Communication is done by representation

Be Stateless

*We'll look at these, and more, in the next section.*

# Questions?