WOWODC '012

MONTREAL JUNE 30, JULY 1ST AND 2ND 2012

Security and performance designs for client-server communications

Helmut Tschemernjak
HELIOS Software GmbH
www.helios.de

Scope of This Presentation

- How we did certain client-server implementations
  - Using WebObjects without an extra WebServer
  - Login authentication options
  - Setting native process security
  - Java WO to native server protocol designs
  - Streaming content to Web clients (downloads/uploads)
  - Server-based preview generation
  - XML communication between iOS App and WebObjects

The Solution Example

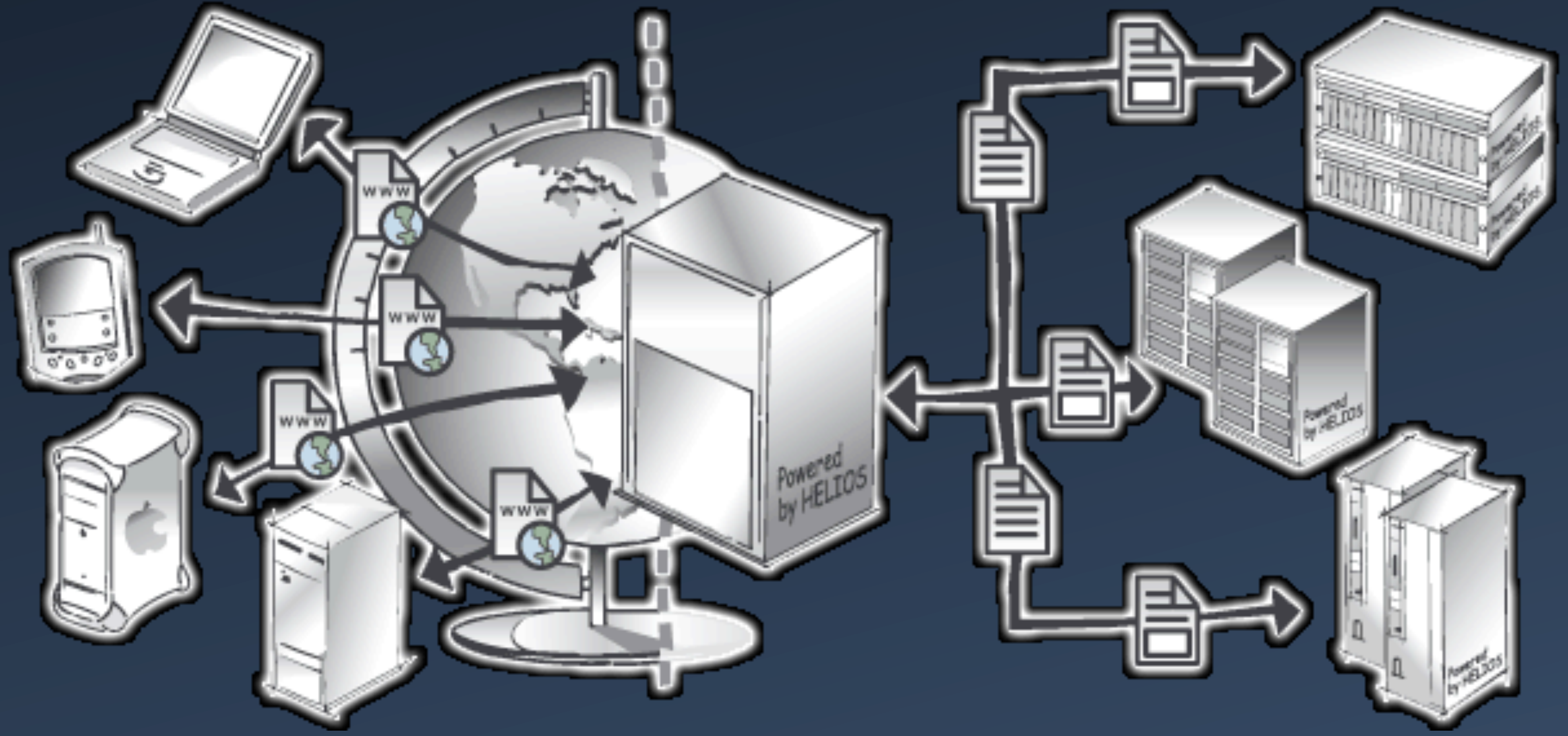
Web clients

Web server
(WebObjects based)

File server
(with production data)

WebShare
Highest-Performance Server for Real Time Remote File Access

HELIOS



File Server Role

- Hosts many TB of data
  - Data should not be available on the Web server (no NFS mounts)
- Image rendering must be done on the file server to transfer only low-res to Web clients
- Authentication needs to be done with the file server account
- File access should enforce the user's file permissions (ACLs, NTFS, UNIX, …)

Web Server (WebObjects based)

- We decided to deploy WebObjects only
  - No extra Web server needed
  - No dependency on Apache, ISS
  - No WebObjects adaptor needed
  - No dependency on OS Linux/UNIX/Windows 32 or 64-bit)
  - Easier installation

WebObjects Direct Connect & HTTPS

```
public class Application extends WOApplication {

public static void main(String argv[]) {

    /* enable direct HTTP connections */
     if (System.getProperty("WODirectConnectEnabled") == null)
          System.setProperty("WODirectConnectEnabled", "true");

    /*
     * Contents/Resources needs the following files:
     * adaptorssl.key: the SSL key file generated via the java keytool:
     *     keytool -genkey -keystore serverkeys -keyalg rsa -alias qusay
     * adaptorsslpassphrase: A script/program which outputs the keystorepass
     *                       on stdout, e.g.:
     *                       #!/bin/sh
     *                       echo -n hellothere
     */
   if (System.getProperty("SSLPort") != null) {
      System.setProperty("WOAdditionalAdaptors", "({WOAdaptor=WOSSLAdaptor;})");
   }
   ...
```

WebObjects Direct Connect – Multiple Hosts

```
public static void main(String argv[]) {
    ...
    if (System.getProperty("WOHost") != null) {
        /* Build and set property string for WOAdditionalAdaptors property.
         * The first host will be served by the default WOAdaptor, If only
         * one hostname is defined WOAdditionalAdaptors will be set to "()"
         * representing an empty array unless SSLPort is set. If SSLPort is
         * set, a WOSSLAdaptor will be added for each defined hostname.
         */
        woHosts = System.getProperty("WOHost").split("\\s*,\\s*");
        /* sslActive and sslOnly flags are set in adaptorWithName method */
        boolean isSSL = (System.getProperty("SSLPort") != null);
        StringBuffer b = new StringBuffer("(");
        for (short i = 0; i < woHosts.length; i++) {
            if (i > 0) /* first defined host is served by default WOAdaptor */
                b.append("{WOAdaptor=WODefaultAdaptor;},");
            if (isSSL) /* add a SSL adaptor for each host */
                b.append("{WOAdaptor=WOSSLAdaptor;},");
        }
        /* overwrite WOAdditionalAdaptors property */
        System.setProperty("WOAdditionalAdaptors", b.append(")").toString());
    }
```

WebObjects Direct Connect – Multiple Adaptors

```
public WOAdaptor adaptorWithName(String name, NSDictionary anArgsDictionary) {
    if (adaptorSettings == null)
        adaptorSettings = new NSMutableDictionary(anArgsDictionary);

    int idx, port;
    String portPref;

    if (name.equals("WOSSLAdaptor") == false) { /* WODefaultAdaptor or WSNullAdaptor */
        portPref = System.getProperty("WOPort");
        /* return a WSNullAdaptor for any non SSL adaptor if WOPort is set to "0" */
        if ("0".equals(portPref)) {
            name = "WSNullAdaptor";
            sslOnly = true;
        }
        idx = httpAdaptorCount++;
    } else { /* WOSSLAdaptor */
        portPref = System.getProperty("SSLPort");
        sslActive = true;
        idx = sslAdaptorCount++;
    }

    try {
        port = Integer.parseInt(portPref);
    } catch (NumberFormatException e) {
        NSLog.debug.appendln("ERROR: Could not parse port configuration for WOAdaptor '" + name + "': " + e);
        return null;
    }

    /* set the adaptors host if any host is defined */
    if (woHosts != null) {
        NSLog.debug.appendln("adaptorWithName: " + name + " for host '" + woHosts[idx] + "'" + (port != 0 ? " on port " + port : ""));
        adaptorSettings.setObjectForKey(woHosts[idx], "WOHost");
    }

    adaptorSettings.setObjectForKey(new Integer(port), "WOPort");
    adaptorSettings.setObjectForKey(name, "WOAdaptor");
    return super.adaptorWithName(name, adaptorSettings);
}
```

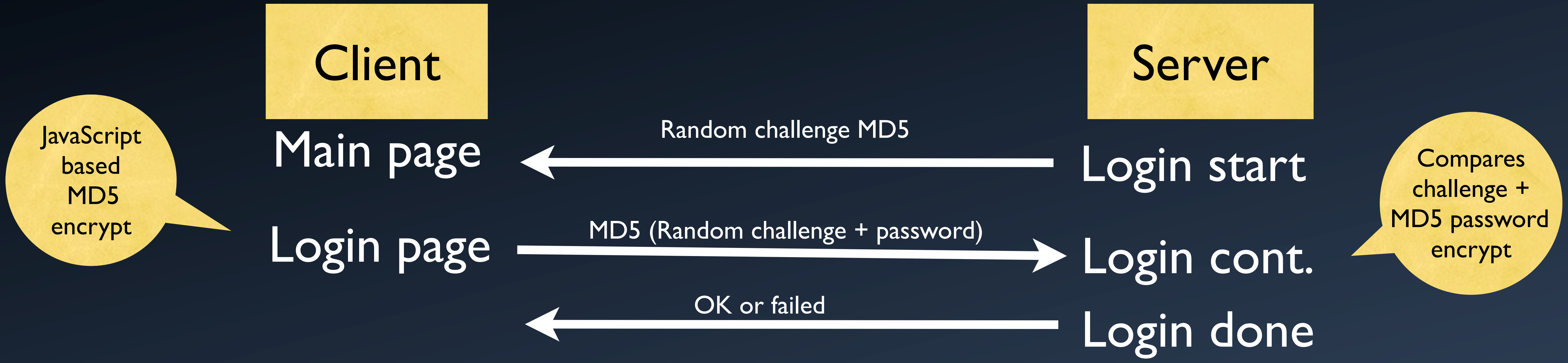
WebObjects Direct Connect – GZIP Content

```
public void appendToResponse(WOResponse aResponse, WOContext aContext) {
    super.appendToResponse(aResponse, aContext);

    aResponse.setHeader("Accept-Encoding, Accept-Language", "Vary");
    String encodings = aContext.request().headerForKey("Accept-Encoding");
    if (encodings == null || encodings.indexOf("gzip") == -1)
        return;

    try {
        byte [] content = aResponse.content().bytes(0, aResponse.content().length());
        ByteArrayOutputStream byteArrayOStream = new ByteArrayOutputStream(content.length / 3);
        GZIPOutputStream gzipOStream = new GZIPOutputStream(byteArrayOStream);
        gzipOStream.write(content, 0, content.length);
        gzipOStream.close();

        NSData contentGzipped = new NSData(byteArrayOStream.toByteArray());
        aResponse.setHeader("gzip", "Content-Encoding");
        aResponse.setHeader(String.valueOf(contentGzipped.length()), "Content-Length");
        aResponse.setContent(contentGzipped);
    } catch(IOException e) {
        D.LOG(D.CMD, "GZIP response failed: " + e);
    }
}
```

Login Authentication Options

- Cleartext logins are bad
- HTTPS encrypts data, however:
  It is cleartext again within Web app
- JavaScript MD5 checksum is better
- RSA encrypted password
  to work against a password server

MD5 Example



Client

Server

JavaScript based MD5 encrypt

Main page

Random challenge MD5

Login start

Login page

MD5 (Random challenge + password)

Login cont.

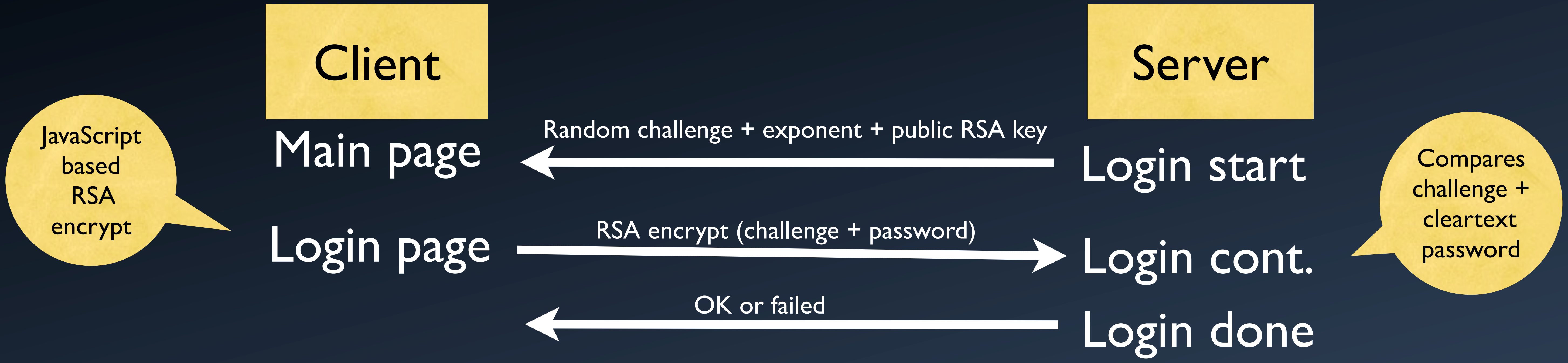
Compares challenge + MD5 password encrypt

OK or failed

Login done

- No need for cleartext passwords on the server
- Challenge avoids replaying login packets

RSA Example



Client

Server

JavaScript based RSA encrypt

Main page

Random challenge + exponent + public RSA key

Login start

Login page

RSA encrypt (challenge + password)

Login cont.

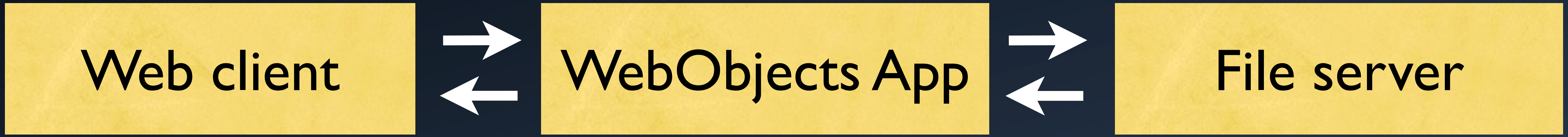
Compares challenge + cleartext password

OK or failed

Login done

- Server can decode cleartext password
  RSA request can also be forward to a password server
- Challenge avoids replaying login packets

File Server Access

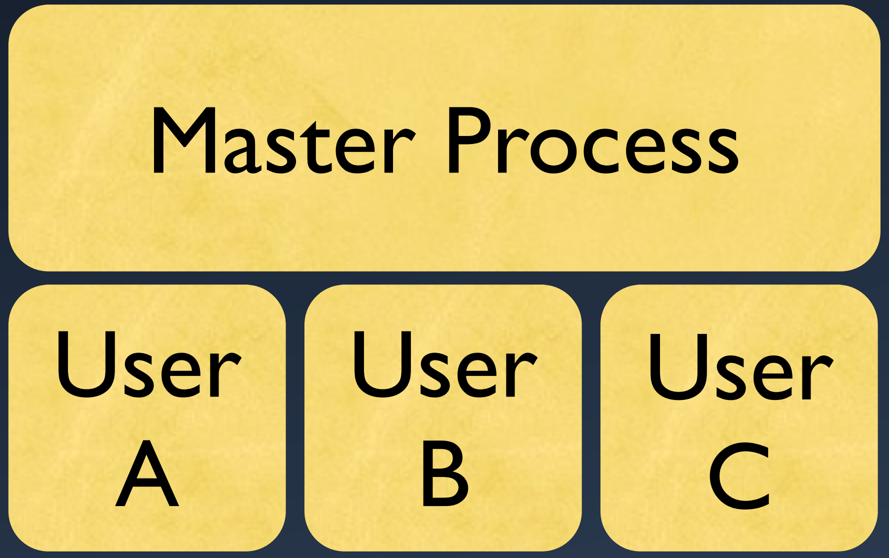


Web client ⇄ WebObjects App ⇄ File server

- File server hosts documents, images, videos, etc.
- Local users work with AFP/SMB directly on server volumes
- File system security can be be enforced
- Separate process per Web user allows asynchronous processing and protects other users in case of errors

File Server Process Design

- Master process accepts incoming connections
- Start process per user
  - Use fork on UNIX
  - Use fork+execv on Mac OS X in case you need use Cocoa/Carbon APIs
  - Use CreateProcessW on Windows with a username/password use CreateProcessAsUserW



Setting Process Security Context

- UNIX
  - After fork use setuid, setgid, setgroups
- Windows
  - CreateProcessAsUserW is one option
  - Check MSDN userToken related manuals to switch IDs:
    OpenThreadToken, SetThreadToken, GetTokenInformation, ImpersonateLoggedOnUser, RevertToSelf

Summary:
Authentication & Process Security

- Benefits from proper process setup
  - Integrates well into the OS
  - Quota (disk & other resources) works
  - File system access permissions works
  - Process security/isolation works
  - Auditing and tracing works
  - Automatically scaling – every user has its own process

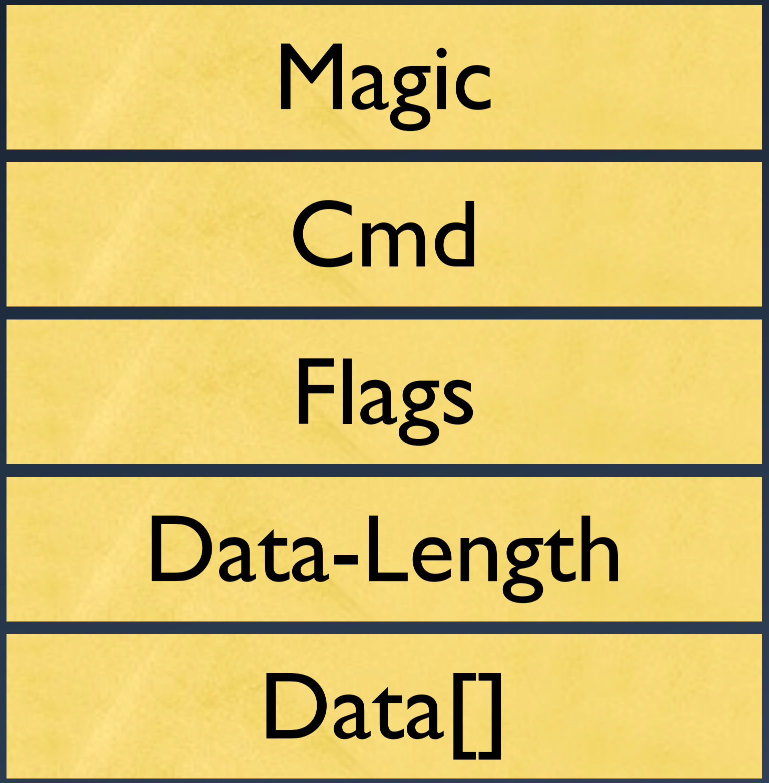
    It is clear that multiple threads can asynchronously do IO, however once the process dies it is over for all users.

Client-Server Protocol Design

- We have over 25 years of experience in client-server protocols
  - Apple Filing Protocol – AFP Server (since 1989)
  - MS-DOS network redirector client (in 1991)
  - Server Message Block – SMB/CIFS Server (since 1994)
  - WebShare three-tier solution (since 2002)
  - Java based Web server (experimental only)
  - Remote tasks automation (uses a HELIOS RPC system)

Client-Server Protocol Design II

- A simple protocol header
  Can be used in every Request & Response
  Read header including length first, then read data content



Magic

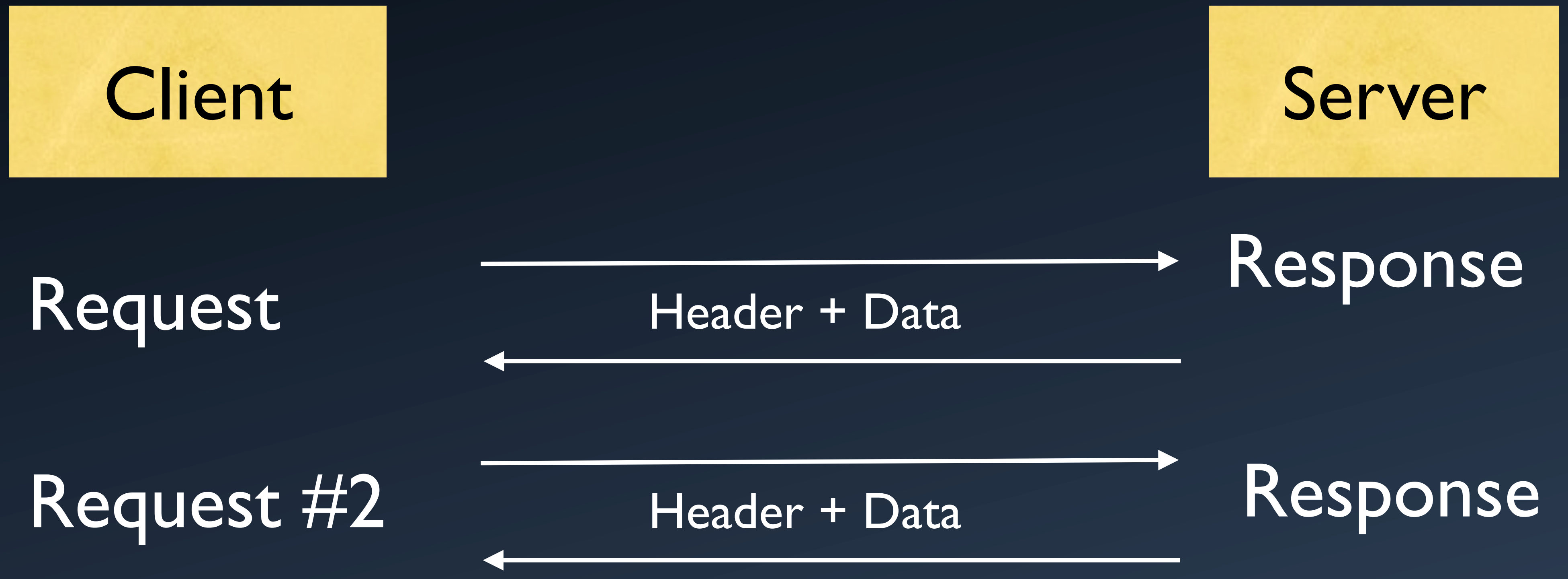
Cmd

Flags

Data-Length

Data[]

Sample Protocol Design cont.



Client

Server

Request

Header + Data

Response

Request #2

Header + Data

Response

- Looks easy
  - What to do with long delays in responses?
  - What to do with very large response streams?

Any Ideas?

- What to do with long delays in responses?
- What to do with very large response streams?

  For example: an image or file download/upload

Sample Protocol Design cont.

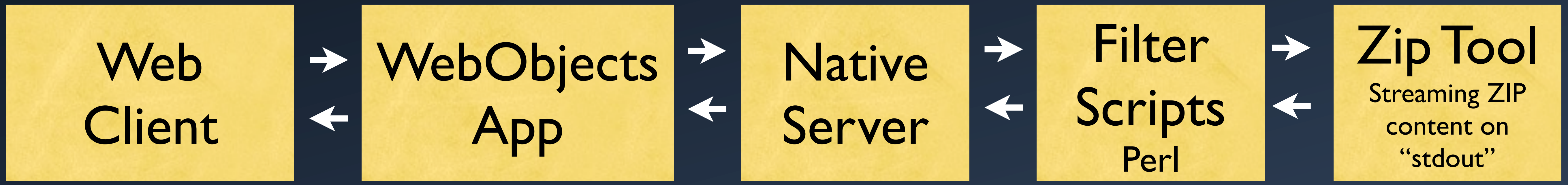
- Simply return a TCP port number in the response packet
- Connect with a separate TCP socket
- Pickup/send data until EOF
- Benefits are:
  - Asynchronous receives/sends
  - Works perfectly with TCP (streaming, delayed acks, etc.)
  - Main command requests can continue while large data is in transit

Sample Protocol Design cont.

```
    /*
     * Make sure your average requests fits into the socket buffer
     * this greatly improves streaming performance
     * check if SNDBUF/RCVBUF settings, if it is already large enough no need to change it
     */

setsockopt(s, SOL_SOCKET, SO_RCVBUF, (char *)&tcpRcvWinSize, sizeof(tcpRcvWinSize))
setsockopt(s, SOL_SOCKET, SO_SNDBUF, (char *)&tcpSendWinSize, sizeof(tcpSendWinSize))

    /*
     * Keep alive will remove dead connections more quickly
     * No delay is important if requests where you write the entire data in one go without a need that the
     * tcp kernel waits to collect more data before sending
     * REUSEADDR ensures that a restart of your server process can listen on the same port again
     */

on = 1;
setsockopt(s, SOL_SOCKET, SO_KEEPALIVE, (char *)&on, sizeof(on))
setsockopt(s, IPPROTO_TCP, TCP_NODELAY, (char *)&on, sizeof(on))
setsockopt(listenfd, SOL_SOCKET, SO_REUSEADDR, (char *) &on, sizeof(on)
```

Sample Protocol Design cont.

- Example for streaming download content
  - Similar setup for upload, image previews, etc.



Web Client

WebObjects App

Native Server

Filter Scripts

Perl

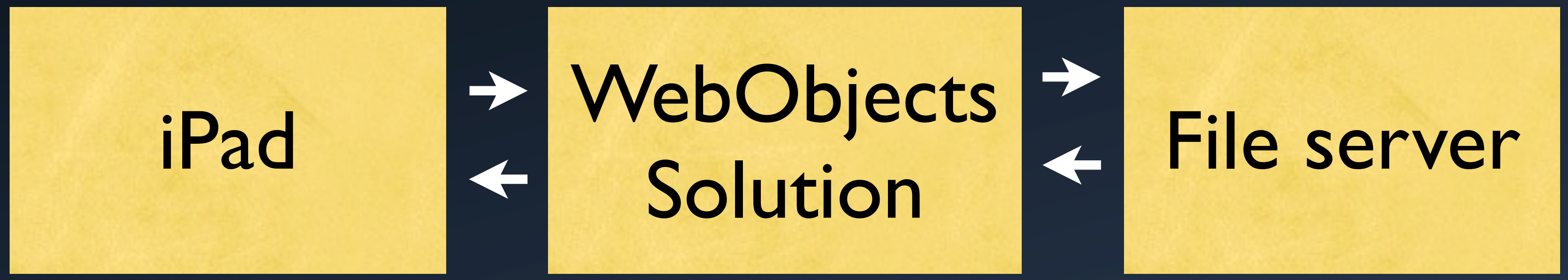
Zip Tool

Streaming ZIP content on "stdout"

Protocol Design – Summary

- A good protocol design makes your solution:
  - Scalable
  - Robust
  - Secure
  - Extensible

XML Communication between iOS Client App and WebObjects



- A sample XML content response for a remote file status
- XML gets basically generated with dynamic data by WO similar to generated Web content
- Code is only partial extraction to get an idea how it works

XML File: XWSStatMapping.xml

XWSStat

XWSExtendedDirectoryEntry

```
  <model>
 <entity name="de.helios.webshare.xml.XWSStat" xmlTag="stat">
     <property name="sharepoint" xmlTag="sharepoint" attribute="YES"/>
     <property name="items" xmlTag="file"/>
 </entity>
 <entity name="de.helios.webshare.xml.XWSExtendedDirectoryEntry" xmlTag="file">
     <property name="path" xmlTag="id" attribute="YES"/>
     <property name="entry.fsize" xmlTag="size" attribute="YES"/>
     <property name="entry.dsize" xmlTag="data-size" attribute="YES"/>
     <property name="entry.fileType" xmlTag="type" attribute="YES"/>
     <property name="entry.fileCreator" xmlTag="creator" attribute="YES"/>
     <property name="entry.labelID" xmlTag="label" attribute="YES"/>
     <property name="entry.modTime" xmlTag="modified" attribute="YES"/>
     <property name="entry.creationTime" xmlTag="created" attribute="YES"/>
     <property name="entry.openMode" xmlTag="mode" attribute="YES"/>
     <property name="entry.fileID" xmlTag="file-id" attribute="YES"/>
     <property name="entry.fmode" xmlTag="file-mode" attribute="YES"/>
     <property name="entry.fowner" xmlTag="file-owner" attribute="YES"/>
     <property name="entry.fgroup" xmlTag="file-group" attribute="YES"/>
     <property name="entry.fcomment" xmlTag="comment" attribute="NO"/>
 </entity>
</model>
```

XWSStat

```
    public class XWSStat {

    public String sharepoint;
    public NSArray items;

    public XWSStat(String aSharepointName, Vector someEntries) {
        sharepoint = aSharepointName;
        items = new NSArray(someEntries, new NSRange(0, someEntries.size()), true);
    }

    public String getSharepoint() {
        return sharepoint;
    }
}
```

XWSExtendedDirectoryEntry

```
    public class XWSExtendedDirectoryEntry {

    public iWSDirectoryEntry entry;
    public String path;

    public XWSExtendedDirectoryEntry(iWSDirectoryEntry anEntry, String aPath) {
        entry = anEntry;
        entry.fileCreator = WSUtils.stringToHex(entry.fileCreator);
        entry.fileType = WSUtils.stringToHex(entry.fileType);
        path = aPath;
    }
}
```

iWSDirectoryEntry

```
   public class iWSDirectoryEntry {
       ...
   public String fname;
   public String fileType;
   public String ficonID;
   public long fsize;
   public long modTime;
   public long creationTime;
   public long dsize;
   public Date mtime;
   public String mtimeStr;
   public String mtimeShortStr;
   public int   openMode;
   public String fmode;
   public String fowner;
   public String fcomment;
       ...
}
```

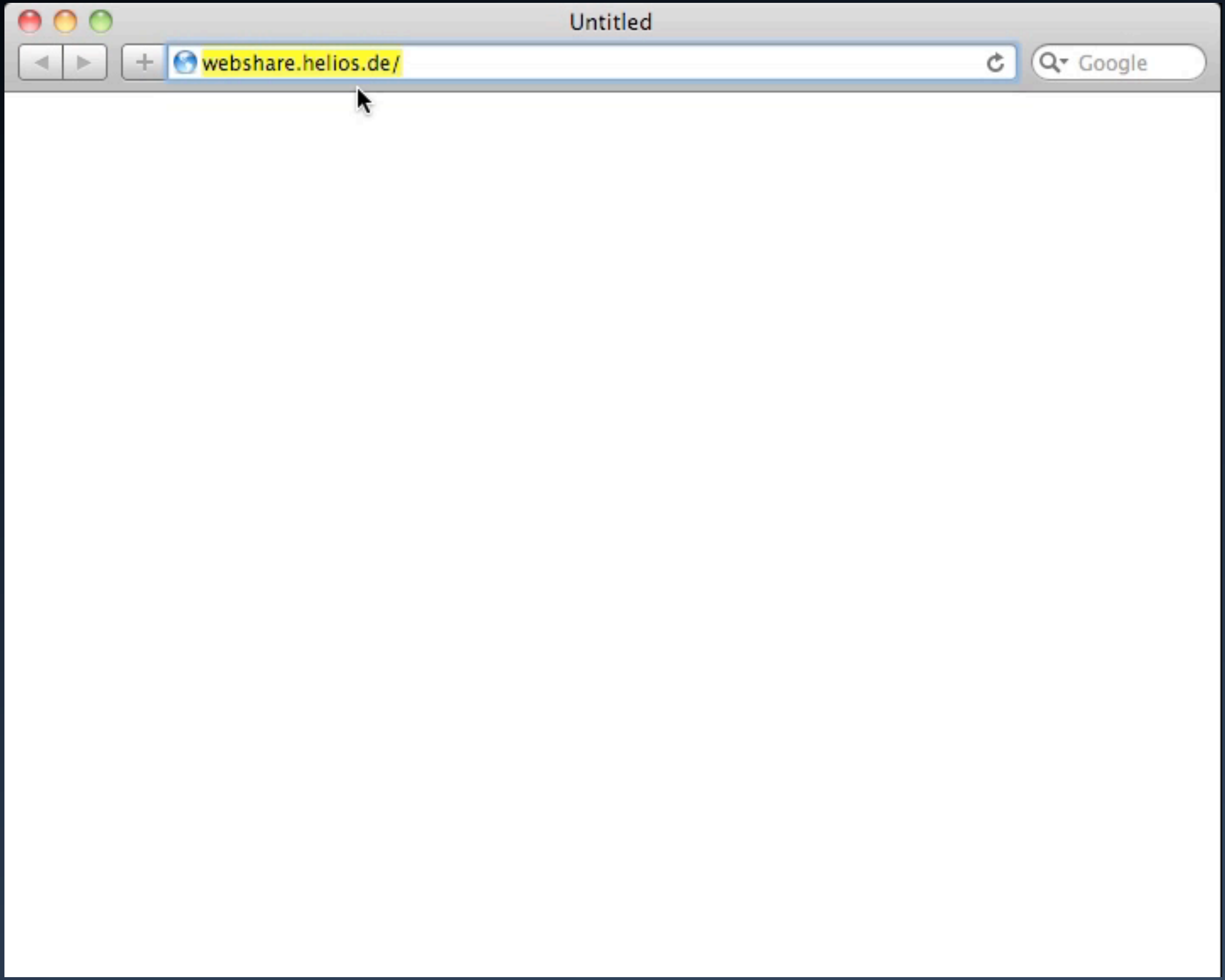
Direct Action Response

```java
registerRequestHandler(new WSManagerRequestHandler(), WSManagerRequestHandler.REQUEST_HANDLER_KEY); /* request handler setup in Application.java */
private static final WOXMLCoder StatListingCoder       = getXMLCoder("XWSStatMapping.xml");

public class WSDownloadManagerDirectAction extends com.webobjects.appserver.WODirectAction implements ParameterNames, XMLQualifiedNames {

    public WOActionResults statAction() {
        ...
        /* generate the XML response if any entries have been added to the Vector */
        if (entries != null && !entries.isEmpty()) {
            return getXMLResponseForStringAndStatus(StatListingCoder.encodeRootObjectForKey(new XWSStat(sharepoint, entries), E_STAT), WOMessage.HTTP_STATUS_OK);
        }
    }

    static WOResponse getXMLResponseForStringAndStatus(String someContent, int aStatus) {
        WOResponse res = new WOResponse();
        res.setContent(someContent);
        try {
            NSData data = res.content();
            res = WSUtils.generateResponseForInputStream(data.stream(), data.length(), DEFAULT_RESPONSE_TYPE, ZIP_RESPONSE);
            res.setStatus(aStatus);
        } catch (IOException ex) {
            D.LOG(D.CMD, "WebShareDownloadManagerDirectAction: Error while generating XML-Response : " + ex);
            res.setContent("<Exception><![CDATA[" + ex + "]]></Exception>");
            res.setStatus(WOMessage.HTTP_STATUS_INTERNAL_ERROR);
        }
        res.setDefaultEncoding("UTF8");
        res.setContentEncoding("UTF8");
        res.setHeader("text/xml; charset=UTF-8;", "Content-Type");
        return res;
    }
}
```

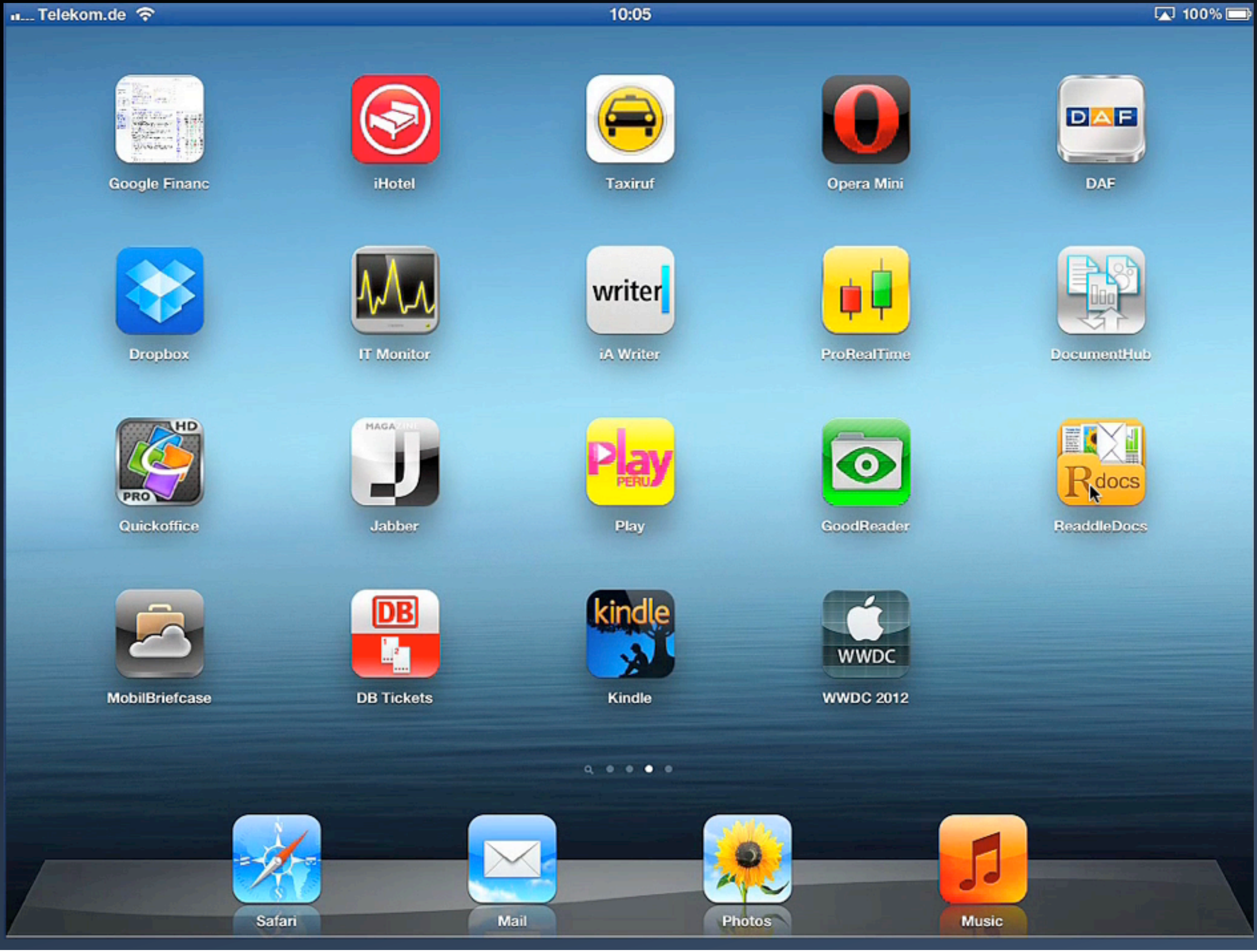
XML Communication – Summary

- WebObjects system to generate XML content
- We use XML protocols to communicate with:
  iPad Document Hub – accessing/syncing documents from iOS
  WebShare Manager – a remote desktop project syncing solution
- XML based commands, we have implemented:
  Login, EnumShares, EnumDirectory, FileStat, SpotlightSearch, Download, Upload, FileComments, ColorLabel, GetIcon, …

WebShare Video Tour



iPad Document Hub Video Tour



Montag, 2. Juli 2012

HELIOS Solutions for Developers

- Server Solution Suite includes:
  - AFP/SMB/Web file servers, imaging tools & PDF workflow
    Image/PDF conversion with ICC Color Management
  - Tool Server for remote automation of jobs
- iPad Document Hub
  - Shared source of complete iOS App for customers
    Allows developing your own apps utilizing HELIOS server services

WebObjects Wishes

- Turn WebObjects source code into Darwin
  This would allow us to maintain it
- Maintenance for WebObjects – fix problems,
  e.g. the 2 GB Upload stream limit: Apple bug report ID 10765546

WOWODC ••012

MONTREAL JUNE 30, JULY 1ST AND 2ND 2012

Q&A