

Using GT.M and EWD to deliver REST Services

Rob Tweed

M/Gateway Developments Ltd



What is REST?

- **RE**presentational **S**tate **T**ransfer
- Term invented by Roy Fielding
 - One of the authors of the HTTP protocol
- Simplification of web services
 - Reaction to the over-complexities that evolved in SOAP and WSDL

SOAP/WSDL versus REST

- SOAP/WSDL:
 - XML standards for Remote Procedure Call (RPC)
 - Specify a method and its inputs, and format of the output
- REST:
 - Simple requests for resources: URLs
 - Modifiers via name/value pairs

SOAP/WSDL versus REST

- SOAP/WSDL:
 - getUser()
 - Very formal structure to request/response
 - XML-based
- REST:
 - <http://example.com/getuser.php?id=123456>
 - No formal standards or pre-determined structures for request/response
 - XML, free text, anything!

REST

- Benefit:
 - Quick, simple, pragmatic way to expose resources and methods
- Downside:
 - Client cannot discover or build interface automatically
 - Needs to be told the required interface
- But...REST interfaces tend to be very simple

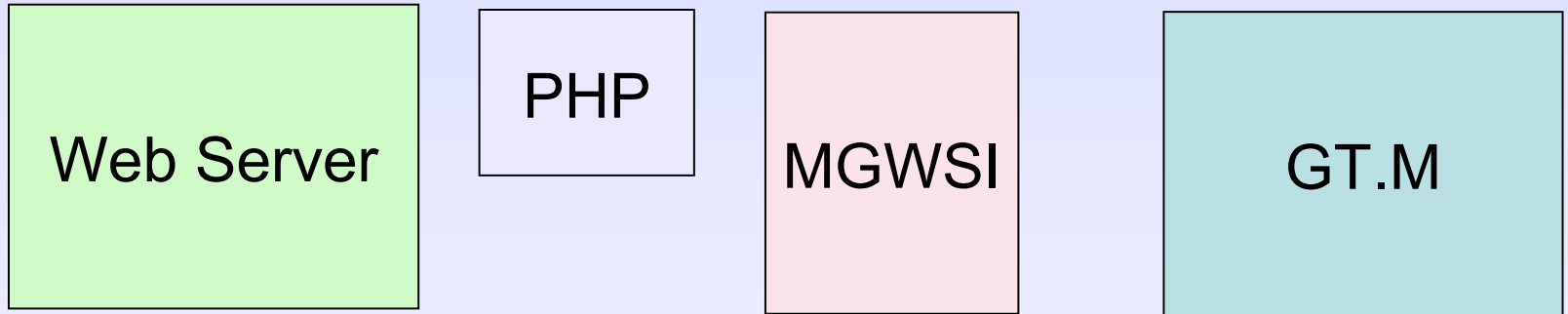
REST Interfaces

- Requests sent via HTTP or HTTPS
- Client can be:
 - A browser (useful for testing)
 - A program (normally)

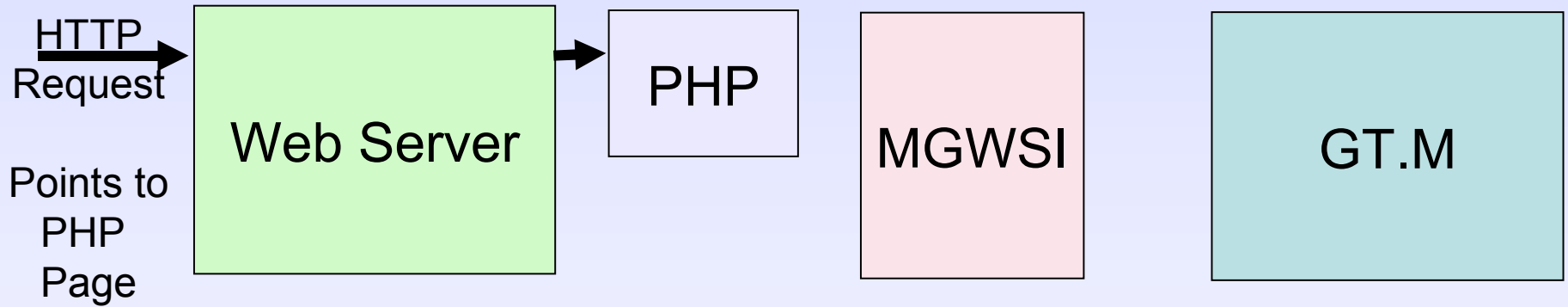
REST and GT.M

- REST has great potential for exposing the methods and data within a GT.M system in a completely open way
- MGWSI and EWD makes it quick, easy and highly scalable

GT.M/REST Architecture

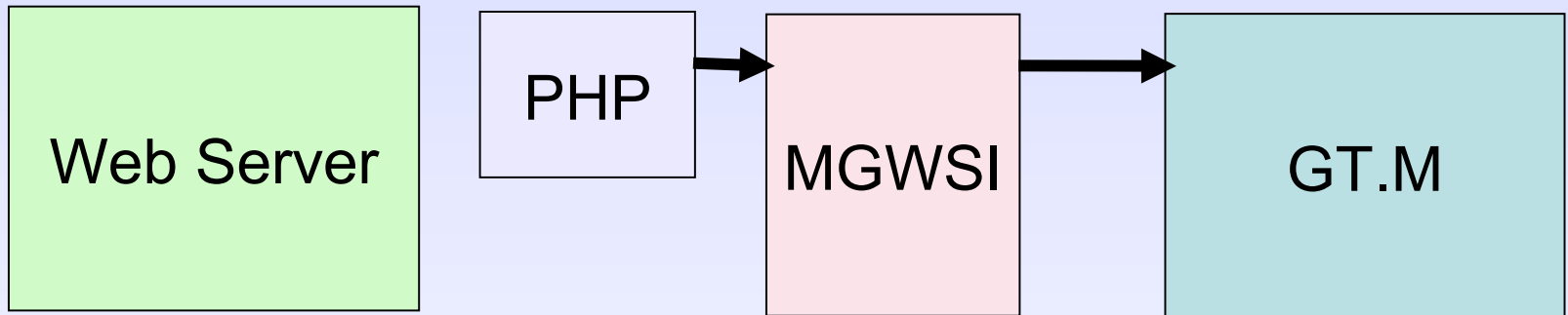


GT.M/REST Architecture

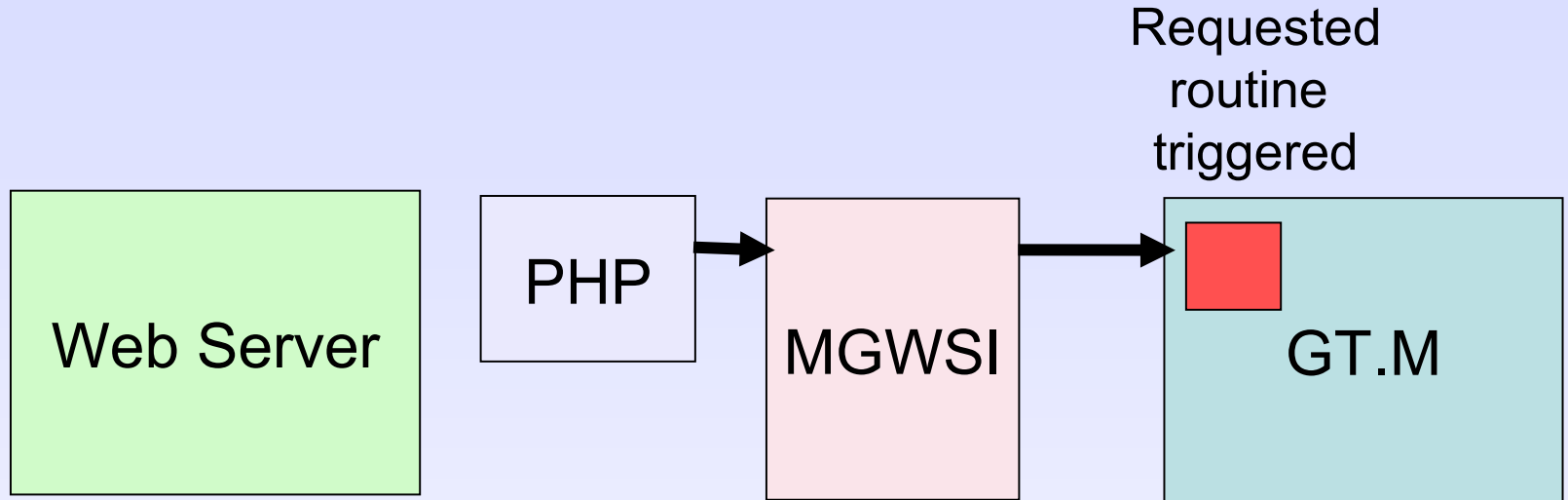


GT.M/REST Architecture

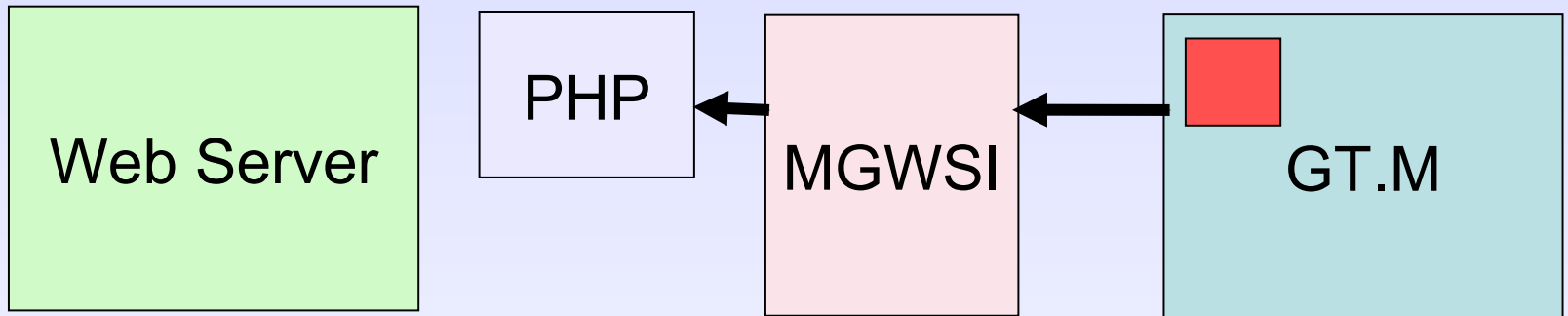
Calls out to GT.M
for data:
RPC call



GT.M/REST Architecture

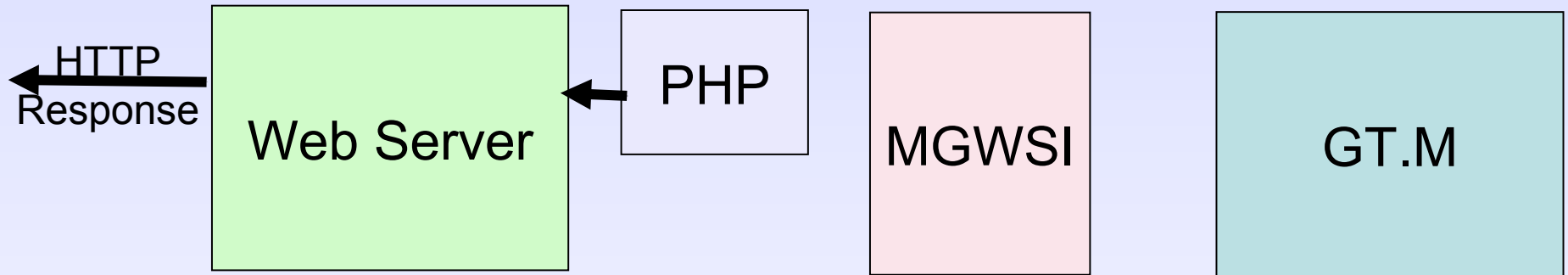


GT.M/REST Architecture



GT.M routine returns
the data to PHP page
as array (by reference)

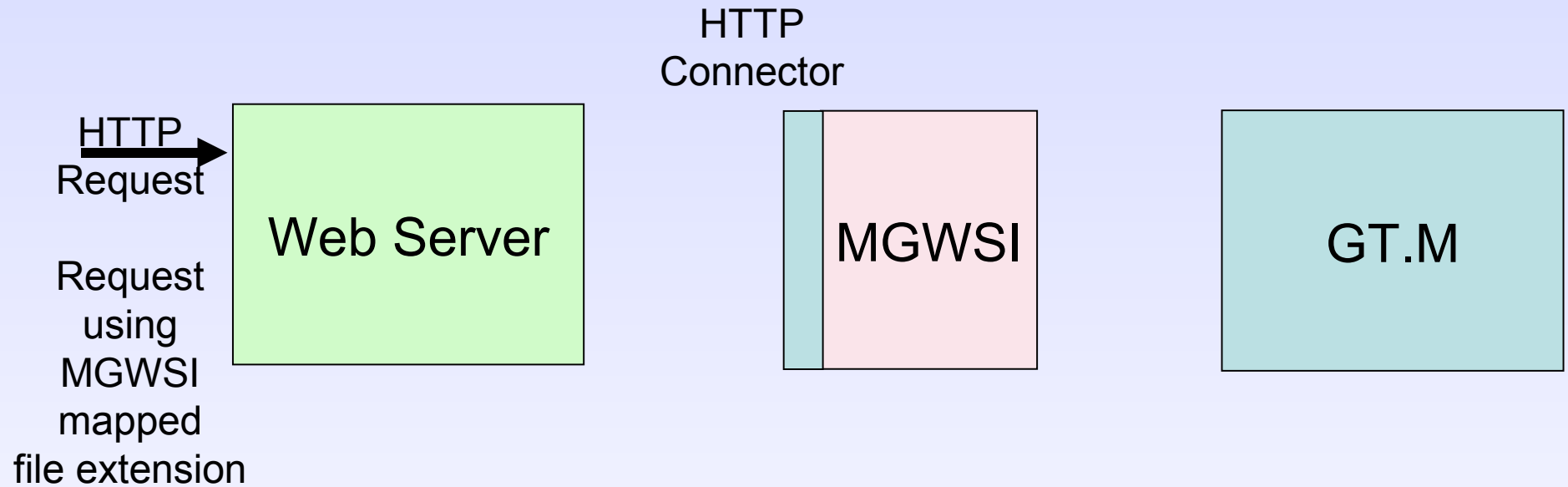
GT.M/REST Architecture



PHP page
creates the
required HTTP
response

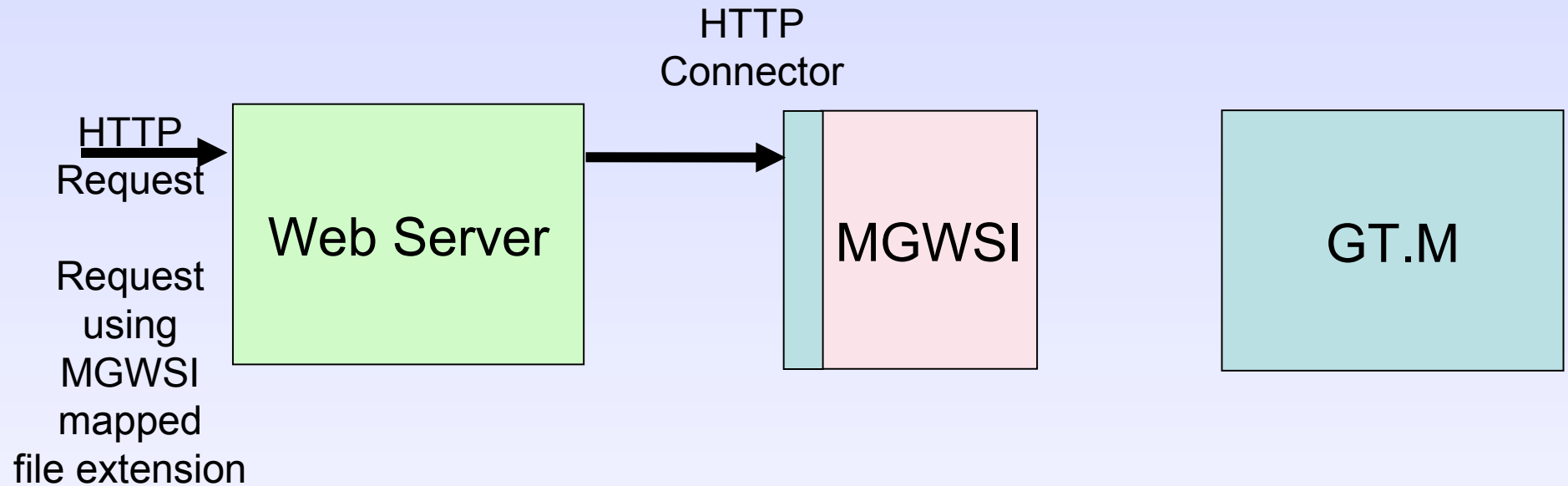
GT.M/REST Architecture

Next generation



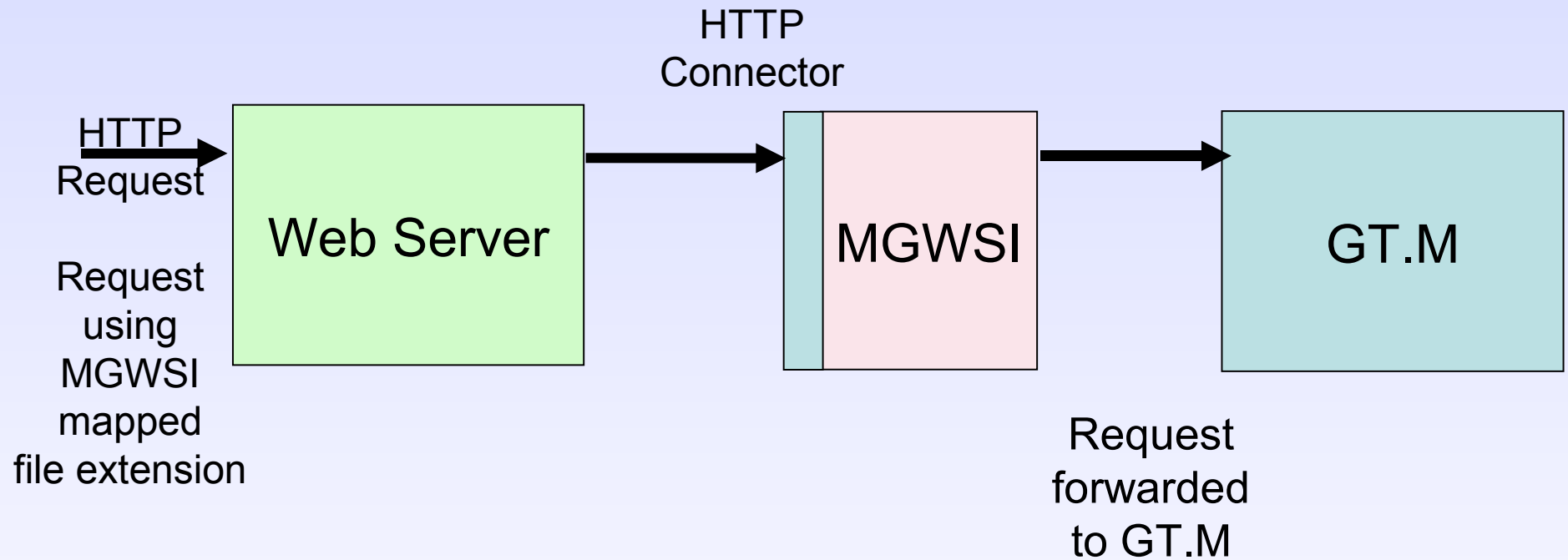
GT.M/REST Architecture

Next generation



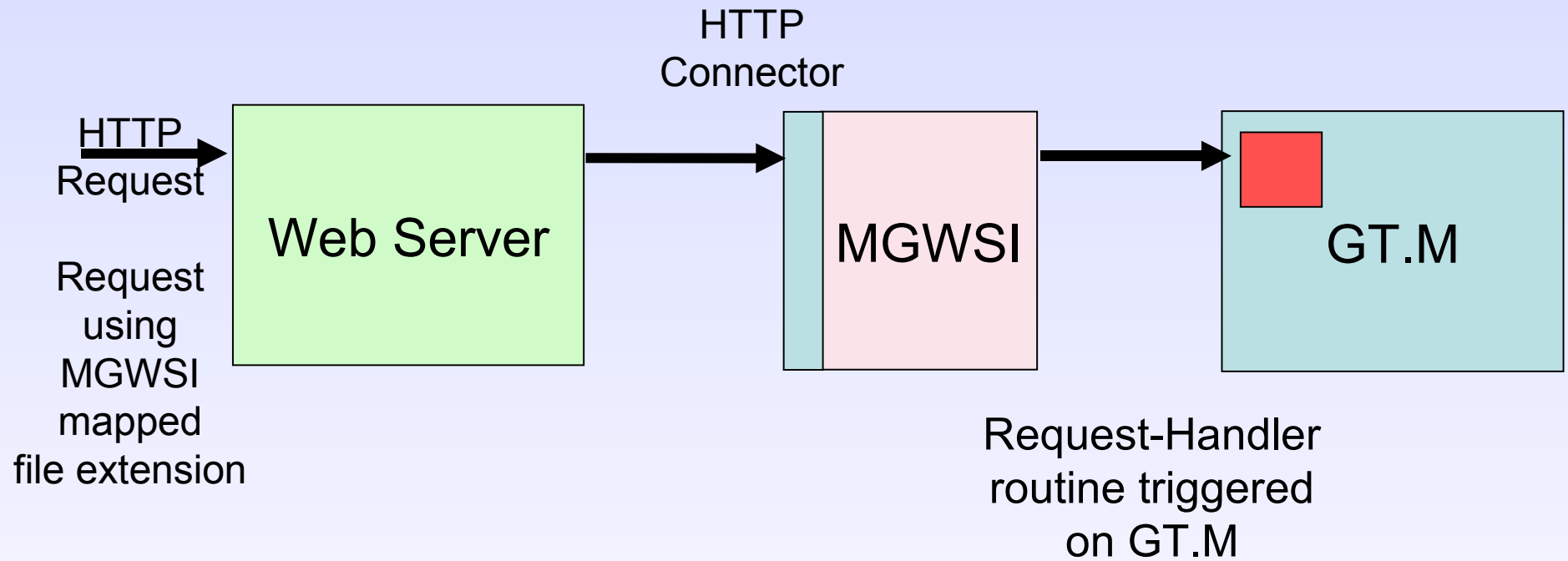
GT.M/REST Architecture

Next generation



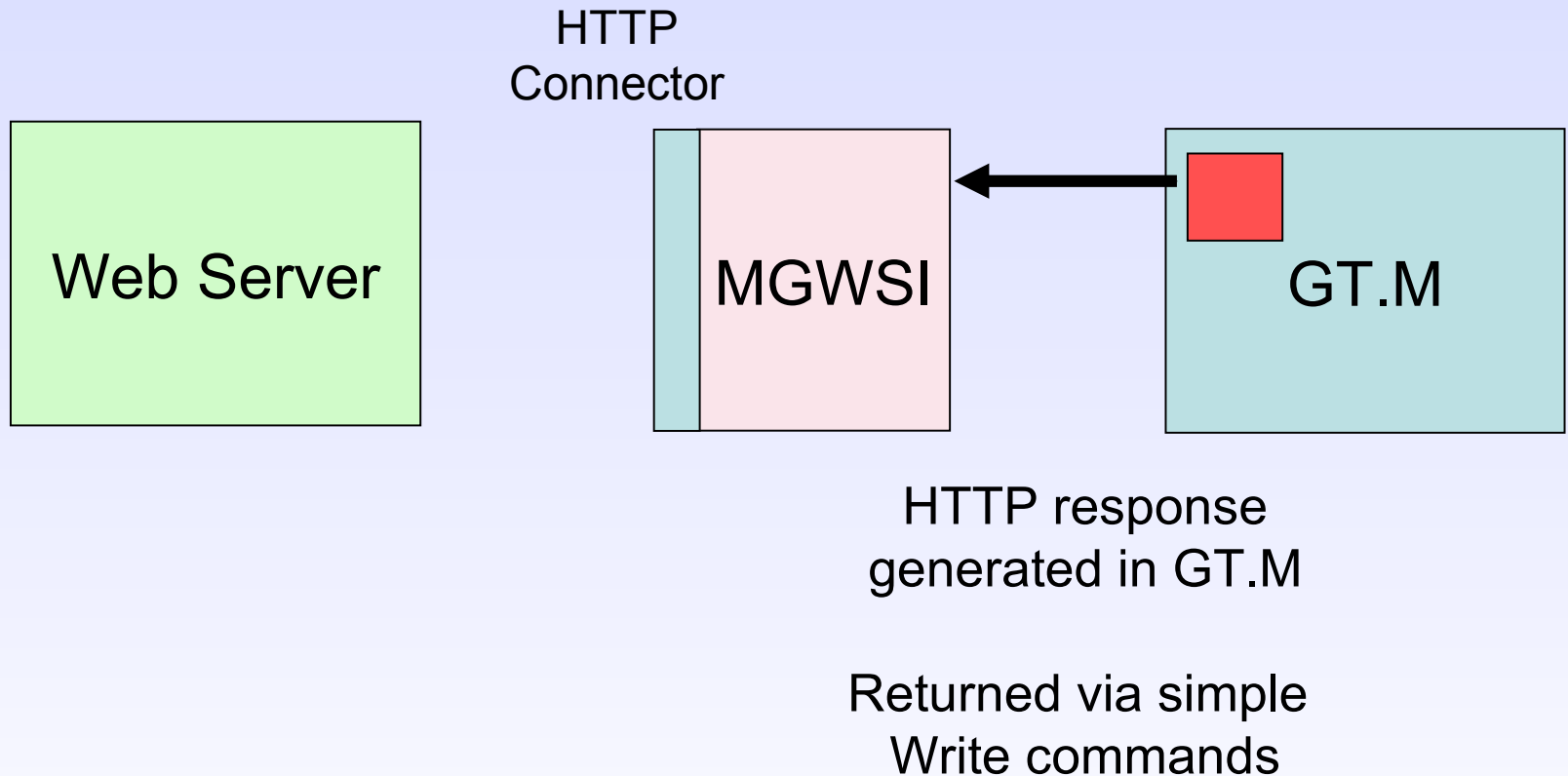
GT.M/REST Architecture

Next generation



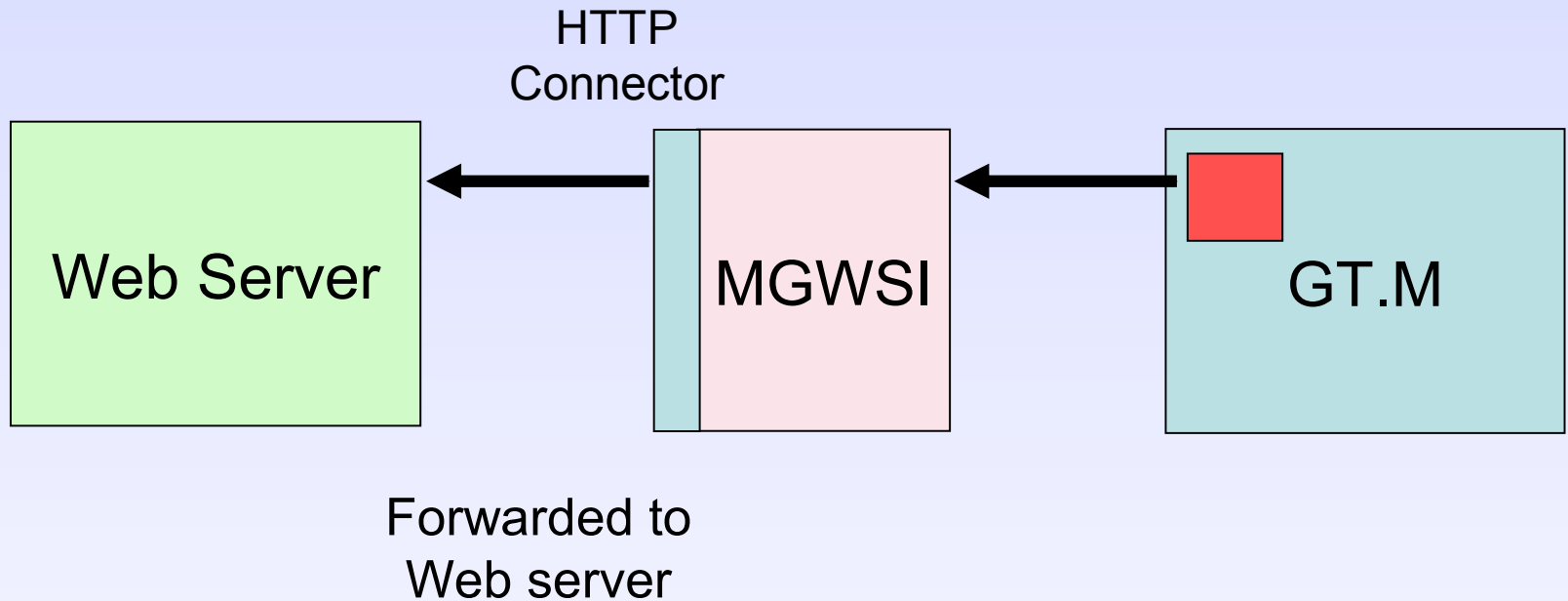
GT.M/REST Architecture

Next generation



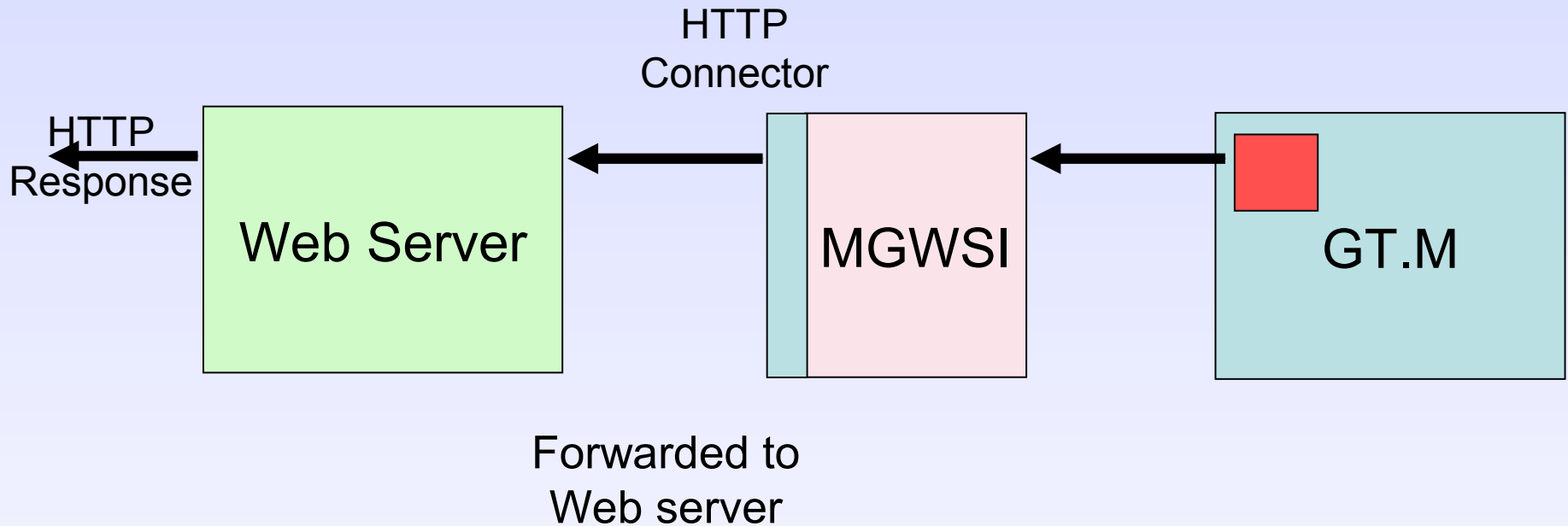
GT.M/REST Architecture

Next generation



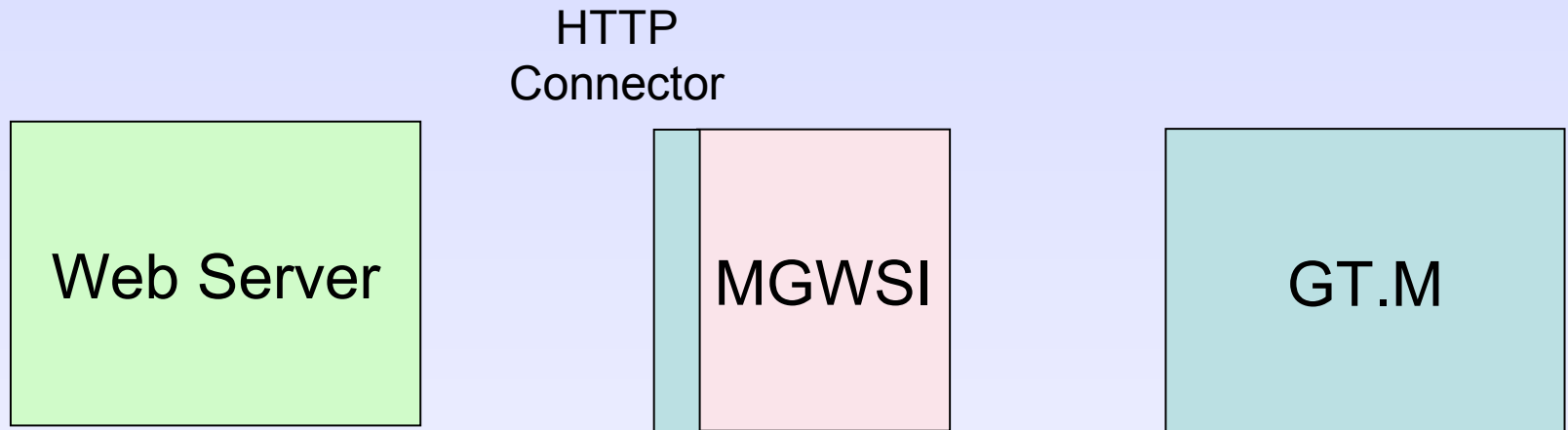
GT.M/REST Architecture

Next generation



GT.M/REST Architecture

Next generation



Less “moving parts”, therefore:

- Higher performance
- Scalability limited only by GT.M, not PHP

EWD REST Server

- Fully documented in ewdMgr application
- Original mechanism for GT.M uses PHP interfacing
 - EWD provides a standard interface page for REST:
 - restServer.ewd
 - name/value pairs specify:
 - The name of the method you wish to invoke
 - The specific inputs your method requires

EWD REST Server

- Use `ewdMgr` to set up mapping between:
 - The exposed name of your method
 - Eg *getProduct*
 - The actual procedure that will be invoked
 - Eg *getProduct^myFunctions*
- Can then access the method with:
 - *<http://myGTMServer/php/ewdMgr/restServer.php?method=product>*

Specifying and accessing inputs

- Product multiplies 2 values
- You decide the names of the inputs and outputs
 - *value1, value2*
 - Produces a *result*
- *http://myGTMServer/php/ewdMgr/restServer.php?method=product&value1=5&value2=10*
- Your back-end method can access the inputs using EWD method *getRequestValue()*:
 - *S value1=\$\$getRequestValue^%zewdAPI("value1",sessid)*

Creating the output

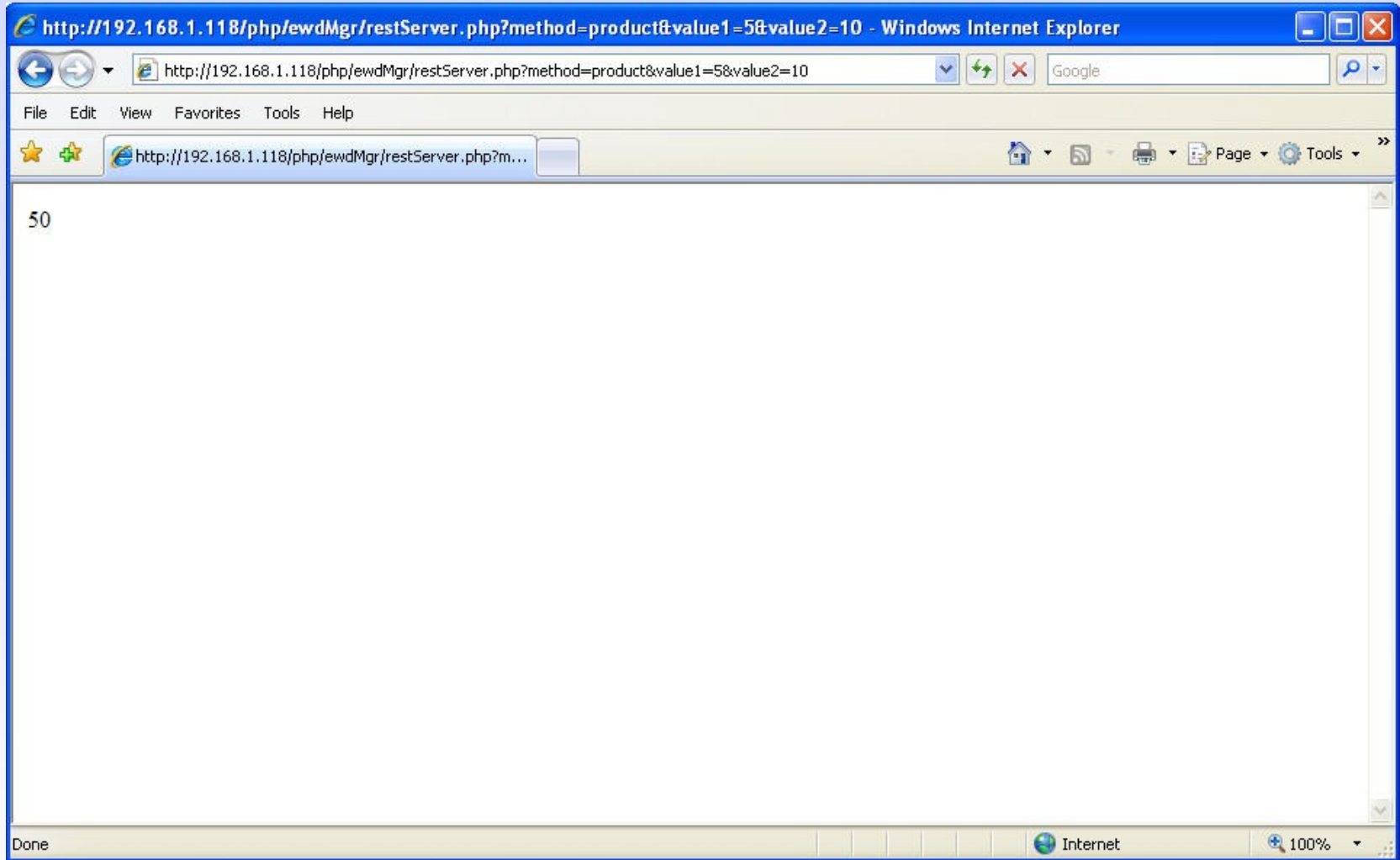
- Create an EWD session variable named *result*
- This can contain anything you like:
 - Text
 - XML
 - Single value
 - Multiple values
 - You determine the output format

Creating the output

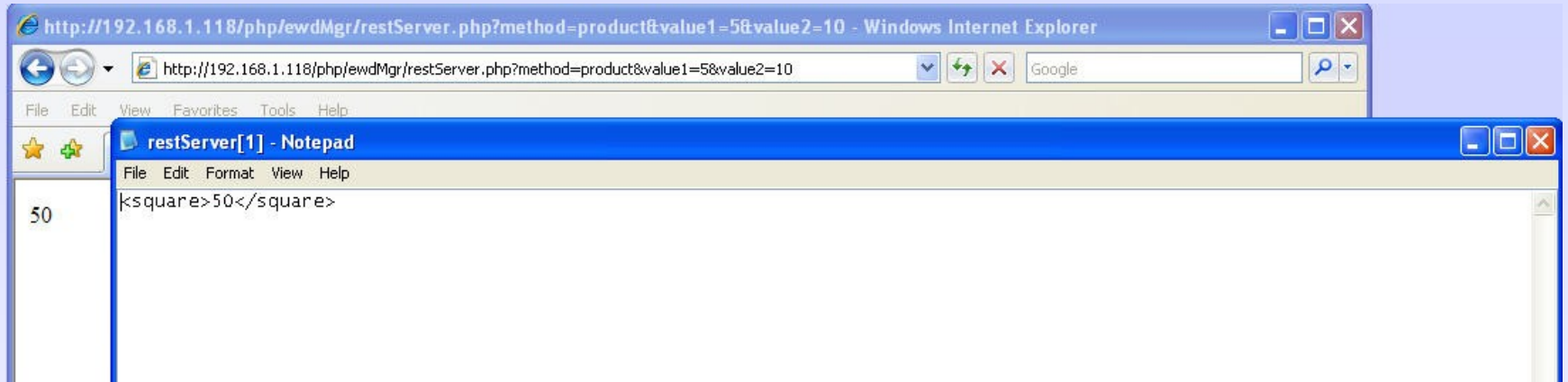
- In our *product* method:

```
product(sessid)
;
s value1=$$getRequestValue^%zewdAPI("value1",sessid)
s value2=$$getRequestValue^%zewdAPI("value2",sessid)
d setSessionValue^%zewdAPI("response","<square>_(value1*value2)_</square>",sessid)
s header("Content-type")="text/xml"
d mergeArrayToSession^%zewdAPI(.header,"ewd.header",sessid)
quit
```

Try running it in a browser



View/Source



HTTP Verbs

HTTP Verb	Meaning
<i>GET</i>	Read
<i>POST</i>	Create, update, delete
PUT	Create, overwrite, replace
DELETE	Delete

HTTP Verbs

- In fact their use is very informal
- Can, in theory, use GET to save, delete etc
- Browsers only understand GET and POST

Securing REST

- Best example is Amazon S3
- If you need to secure REST services, reverse engineer their security
- Add a hash of headers + data as another HTTP header:

```
StringToSign = HTTP-Verb + "\n" +  
                Content-MD5 + "\n" +  
                Content-Type + "\n" +  
                Date + "\n"
```

- *getServerValue^%zewdAPI()* allows you to get HTTP header values

Securing REST

- Need to create the equivalent of this Caché function

```
getSignedString(string,secretKey)
;
new hash
;
s hash=$System.Encryption.HMACSHA1(string,secretKey)
QUIT $System.Encryption.Base64Encode(hash)
QUIT ""
```

- secretKey is provided by Amazon when you register

Securing REST

- Creating the Authorization Header:

```
getAuthorisationHeader(headerString,s3Key,s3SecretKey)
;
new header
;
s header="AWS "_s3Key_": "
s header=header_ $$getSignedString(headerString,s3SecretKey)
QUIT header
;
```

Securing REST

- Server must reverse the process:
 - Extract the various headers
 - Apply the same hashes using the user's key and secret key
 - Secret key known by client and server
 - Secret key is ***never*** transmitted!
 - If the hashes match, run the method
 - Note that date/time is to nearest second
 - Can be used to discriminate repeat requests from same user provided at least 1 second apart

Request for help

- Easy access from GT.M to standard encryption libraries
- Probably openSSL
 - Don't want/need everyone re-inventing this wheel
 - These libraries (SHA-1, MD5, HMACSMA1, etc) are fast becoming standards in HTTP/REST communications

Handling SSL

- GT.M can only handle unencrypted plain text
- Need a proxy to handle the SSL negotiation and encryption
- We tend to recommend Delegate
 - Already installed and configured in EWD Virtual Appliance
 - We have a document available on installing Delegate
 - Assigned a TCP port
 - GT.M forwards requests and receives decrypted response via that port
 - Delegate looks after SSL side of things

GT.M as a REST Client

- EWD includes basic HTTP Client
 - `httpGET^%zewdGTM`
 - Just GET method currently supported

Conclusions

- REST is a great way to expose GT.M methods and data
 - Lightweight, flexible, quick, easy
- Fast and scalable
- EWD makes it even easier
- REST can be secured
 - Amazon's S3 security can be easily reverse-engineered
 - SSL available via proxy such as Delegate

New feature requests

Rob Tweed

M/Gateway Developments



New features/stuff needed

- Encryption routine library
- Facilitating new users:
 - Simple, package-based installer, complete with default database:
 - 2 minutes to “Hello World” without any questions asked
 - Caché users:
 - Better orientation information for Caché users
 - Caché integration to allow easy migration and/or mixed environments (global mapping?)
 - New users:
 - Technology-specific guides (eg GT.M for Python users)
- Slicker return interface for OS call-outs

Current piped file interface

```
fileInfo(path,info)
```

```
  n line,temp
```

```
  k info
```

```
  s temp="temp"_$p($h,"",2)_.txt"
```

```
  i '$$fileExists^%zewdAPI(path) QUIT
```

```
  zsystem "ls -l "_path_">"_temp
```

```
  o temp:(readonly:exception="g fileDateNotExists")
```

```
  u temp
```

```
  r line
```

```
  s info("date")=$p(line," ",6,8)
```

```
  s info("size")=$p(line," ",5)
```

```
  c temp
```

```
  s ok=$$deleteFile^%zewdAPI(temp)
```

```
  QUIT
```

```
fileDateNotExists
```

```
  s $zt=""
```

```
  i $p($zs,"",1)=2 QUIT
```

```
  QUIT
```



In-memory pipe?

- Pipe output to a GT.M variable?

`zoscall ("ls -l /usr/ewdapps", pipedOutputVariable)`

Then simply parse output in variable, rather than opening file, parsing contents, deleting file

More debate!

- Let's see more use of comp.lang.mumps
- More discussions, questions, debate
- If comp.lang.mumps is quiet, people think the language is dead!
- Also:
 - www.outOfTheSlipstream.com
 - The EWD Community Group:
 - <http://groups.google.co.uk/group/enterprise-web-developer-community>