

Table of Contents

This slide deck consists of slides used in 7 lecture videos in Week 4. Below is a list of shortcut hyperlinks for you to jump into specific sections.

- (page 2) [Week 4: Forms, GET, POST, and HTTP](#)
- (page 10) [Week 4: Building HTML Forms](#)
- (page 21) [Week 4: Forms and Cross Site Request Forgery \(CSRF\)](#)
- (page 30) [Week 4: CSRF Support in Django](#)
- (page 44) [Week 4: The POST Refresh Pattern](#)
- (page 49) [Week 4: Implementing POST Redirect in Django](#)

Charles Severance
www.dj4e.com

Form Processing

<https://samples.dj4e.com/getpost/>
<https://samples.dj4e.com/form/>



Forms gather
data and send
it to the server



The screenshot shows a web browser window with the address bar displaying `https://samples.dj4e.com/getpost/postform`. The page title is "Impossible POST guessing game...". Below the title, there is a text input field labeled "Input Guess" and a "Submit Query" button. The page content shows the result of a POST request: "Incoming POST data: guess=42".

samples.dj4e.com/getpost/post X

← → ↻ 🏠 ⓘ 🔒 `https://samples.dj4e.com/getpost/postform`

⚙️ Most Visited 🖼️ Drc 📁 SakaiCar 📁 Sakai 📁 Tsugi 📁 CRsera 📁 Teach 📁 LX

Impossible POST guessing game...

Input Guess

Submit Query

Incoming POST data:
guess=42

Forms GET vs. POST

Two ways the browser can send parameters to the web server

- **GET** - Parameters are placed on the URL which is retrieved.
- **POST** - The URL is retrieved and parameters are appended to the request in the the HTTP connection.

Utility Code – Dump a Dictionary

```
# Call as dumpdata('GET', request.GET)

def dumpdata(place, data) :
    retval = ""
    if len(data) > 0 :
        retval += '<p>Incoming '+place+' data:<br/>\n'
        for key, value in data.items():
            retval += html.escape(key) + '=' + html.escape(value) + '</br>\n'
        retval += '</p>\n'
    return retval
```

[dj4e-samples/getpost/views.py](#)

```
def getform(request):
    response = """<p>Impossible GET guessing game...</p>
    <form>
    <p><label for="guess">Input Guess</label>
    <input type="text" name="guess" size="40" id="guess"/></p>
    <input type="submit"/>
    </form>"""

    response += dumpdata('GET', request.GET)
    return HttpResponse(response)
```

[dj4e-samples/getpost/views.py](https://samples.dj4e.com/getpost/views.py)



<https://samples.dj4e.com/getpost/getform>

```
@csrf_exempt
def postform(request):
    response = """<p>Impossible POST guessing game...</p>
    <form method="POST">
    <p><label for="guess">Input Guess</label>
    <input type="text" name="guess" size="40" id="guess"/></p>
    <input type="submit"/>
    </form>"""

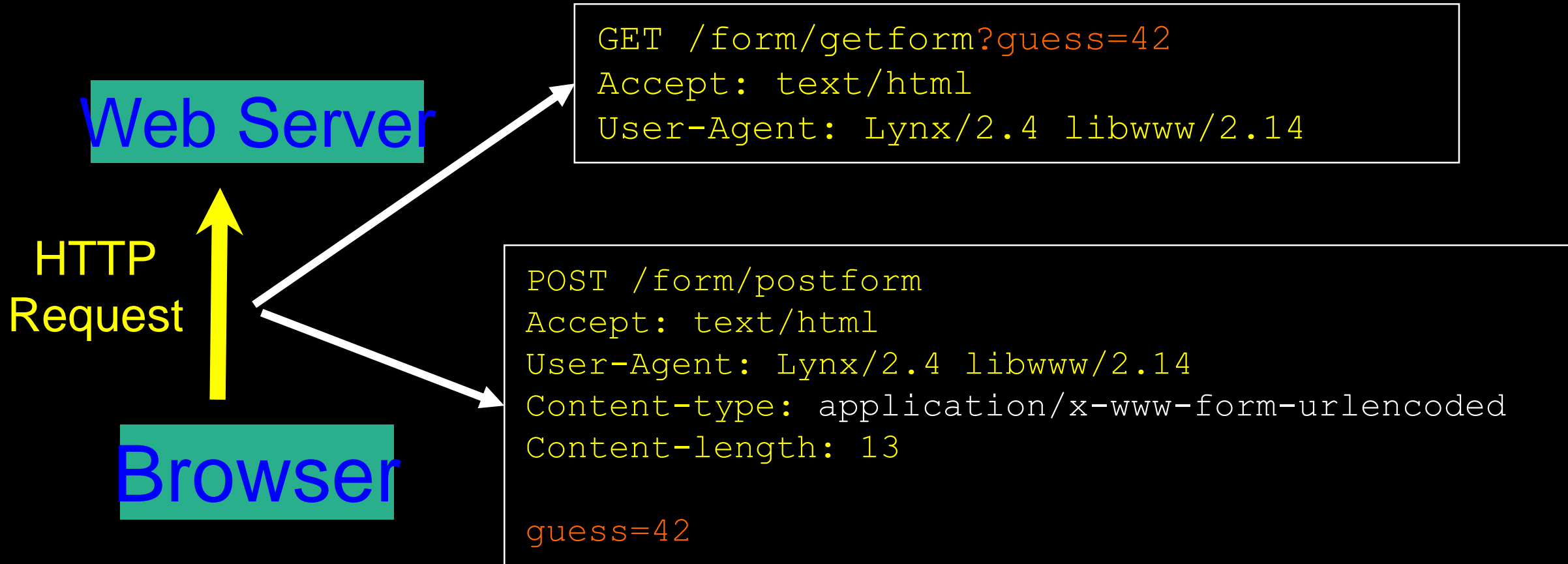
    response += dumpdata('POST', request.POST)
    return HttpResponse(response)
```

[dj4e-samples/getpost/views.py](https://github.com/dj4e/samples/blob/master/getpost/views.py)



<https://samples.dj4e.com/getpost/postform>

Passing Parameters to The Server



`<input type="text" name="guess" id="yourid" />`

Rules of the POST/GET Choice

- POST is used when data is being created or modified.
- GET is used when your are reading or searching things.
- GET should never be used to insert, modify or delete data.
- Web search spiders will follow GET URLs but generally not POST URLs.
- GET URLs should be “idempotent” - the same URL should give the “same thing” each time you access it. (i.e. bookmarkable)
- GET has an upper limit of the number of bytes of parameters and values (think about 2K).

FORMS in HTML

Pre HTML5 Input Types

- Text
- Password
- Radio Button
- Check Box
- Select / Drop-Down
- Textarea

<https://samples.dj4e.com/getpost/html4>
[dj4e-samples/getpost/templates/getpost/html4.html](https://samples.dj4e.com/getpost/templates/getpost/html4.html)

localhost:8000/getpost/html4

Pre HTML 5 input types...

Account:

Password:

Nick Name:

Preferred Time:

☐ AM

☒ PM

Classes taken:

☒ PY4E - Python for Everybody

☒ SI539 - Web Design

☐ SI664 - Web Applications

Which soda:

Which snack:

Tell us about yourself:

```
<p>Many field types...</p>
<form method="post">
  <p><label for="inp01">Account:</label>
  <input type="text" name="account" id="inp01" size="40" ></p>
  <p><label for="inp02">Password:</label>
  <input type="password" name="pw" id="inp02" size="40" ></p>
  <p><label for="inp03">Nick Name:</label>
  <input type="text" name="nick" id="inp03" size="40" ></p>
```

Account:

Password:

Nick Name:

Incoming POST data:

```
account=Beth
pw=12345
nick=nick
when=pm
...
```

dj4e-samples/getpost/templates/getpost/html4.html

```
<p>Preferred Time:<br/>
  <input type="radio" name="when" value="am">AM<br>
  <input type="radio" name="when" value="pm" checked>PM</p>
```

Preferred Time:

☐ AM

☒ PM

Classes taken:

☒ PY4E - Python for Everybody

☒ SI539 - Web Design

☐ SI664 - Web Applications

Incoming POST data:

...

when=pm

class1=on

class2=si539

...

```
<p>Classes taken:<br/>
  <input type="checkbox" name="class1">
    PY4E - Python for Everybody<br>
  <input type="checkbox" name="class2" value="si539" checked>
    SI539 - Web Design<br>
  <input type="checkbox" name="class3" value="si664">
    SI664 - Web Applications<br>
</p>
```

Preferred Time:

☐ AM

☒ PM

Classes taken:

☒ PY4E - Python for Everybody

☒ SI539 - Web Design

☐ SI664 - Web Applications

Incoming POST data:

...

when=pm

class1=on

class2=si539

...

```
<p><label for="inp06">Which soda:
  <select name="soda" id="inp06">
    <option value="0">-- Please Select --</option>
    <option value="1">Coke</option>
    <option value="2">Pepsi</option>
    <option value="3">Mountain Dew</option>
    <option value="4">Orange Juice</option>
    <option value="5">Lemonade</option>
  </select>
</p>
```

The values can be any string, but numbers are used quite often.



Incoming POST data:

...

soda=0

snack=peanuts

...

```
<p><label for="inp07">Which snack:
  <select name="snack" id="inp07">
    <option value="">-- Please Select --</option>
    <option value="chips">Chips</option>
    <option value="peanuts" selected>Peanuts</option>
    <option value="cookie">Cookie</option>
  </select>
</p>
```



SI664 - Web Applications

Which soda: -- Please Select --

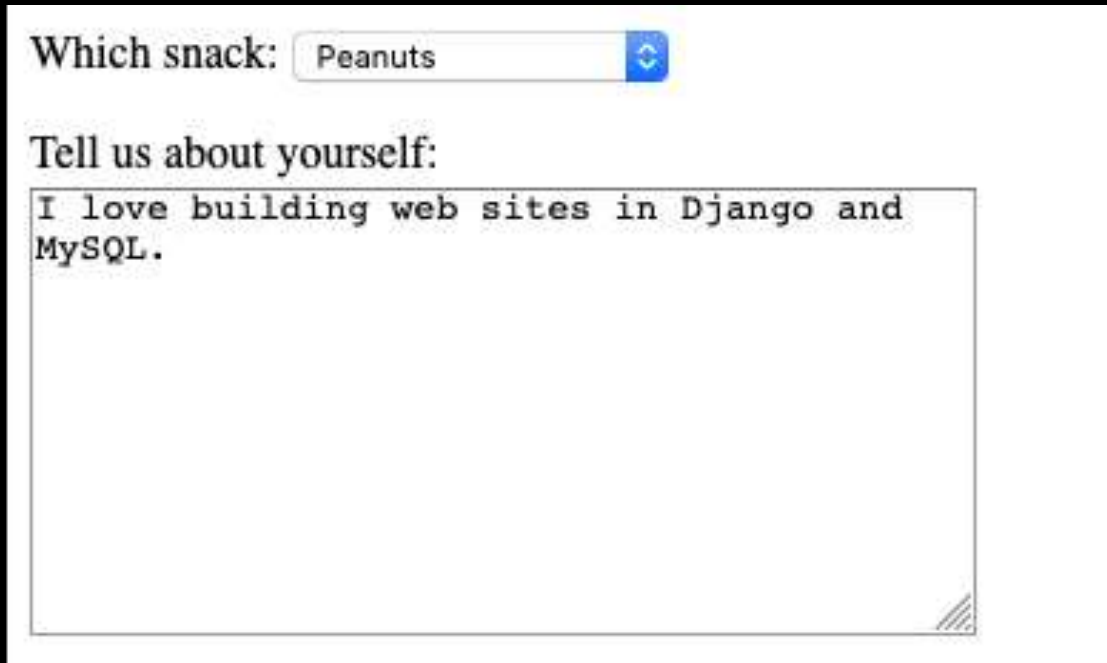
Which snack: Peanuts

Incoming POST data:

```
...
soda=0
snack=peanuts
...
```


dj4e-samples/getpost/templates/getpost/html4.html

```
<p><label for="inp08">Tell us about yourself:<br/>
  <textarea rows="10" cols="40" id="inp08" name="about">
    I love building web sites in Django and MySQL.
  </textarea>
</p>
```



Which snack:

Tell us about yourself:

I love building web sites in Django and MySQL.

Incoming POST data:

```
...
snack=peanuts
about=I love building
web sites in Django and
MySQL.
dopost=Submit
```

dj4e-samples/getpost/templates/getpost/html4.html

```
<input type="submit" name="dopost" value="Submit"/>
<input type="button"
  onclick="location.href='http://www.dj4e.com/'; return false;"
  value="Escape">
```

Tell us about yourself:

I love building web sites in Django and MySQL.

Submit Escape

Incoming POST data:

...

snack=peanuts

about=I love building
web sites in Django and
MySQL.

dopost=Submit

HTML5 Input Types

- HTML5 defined new input types
- Not all browsers support all input types
- They fall back to type="text"

<https://samples.dj4e.com/getpost/html5>

[dj4e-samples/getpost/templates/getpost/html5.html](https://samples.dj4e.com/getpost/templates/getpost/html5.html)

http://www.w3schools.com/html/html5_form_input_types.asp

Select your favorite color:

```
<input type="color" name="favcolor" value="#0000ff"><br/>
```

Birthday:

```
<input type="date" name="bday" value="2003-09-02"><br/>
```

E-mail:

```
<input type="email" name="email"><br/>
```

Quantity (between 1 and 5):

```
<input type="number" name="quantity"
  min="1" max="5"><br/>
```

Add your homepage:


```
<input type="url" name="homepage"><br>
```

Transportation:

```
<input type="flying" name="saucer"><br>
```

In-browser validation happens
when you press submit.

<https://samples.dj4e.com/getpost/html5>



The screenshot shows a web form with the following fields and values:

- Select your favorite color: A color picker showing blue.
- Birthday: A date input field showing 09 / 02 / 2003.
- E-mail: A text input field showing csev@umich.edu.
- Quantity (between 1 and 5): A number input field showing 2.
- Add your homepage: A text input field showing http://www.dr-chuck.com.
- Transportation: A text input field showing yes.
- Buttons: Submit and Escape.

Below the form, the incoming POST data is displayed:

```
Incoming POST data:
favcolor=#0000ff
bday=2003-09-02
email=csev@umich.edu
quantity=2
homepage=http://www.dr-chuck.com
saucer=Yes
dopost=Submit
```

Cross-Site-Request-Forgery (CSRF)

Security

CSRF Attack

- A rogue site generates a page that includes form that posts data to a legitimate site where the user is logged in via a session cookie
- The form is submitted to the legitimate site and the cookie is included
- The legitimate site accepts the request because of the cookie value
- Note that the rogue site does not need to know the cookie value – it just knows that the cookie will be sent on requests to the legitimate site

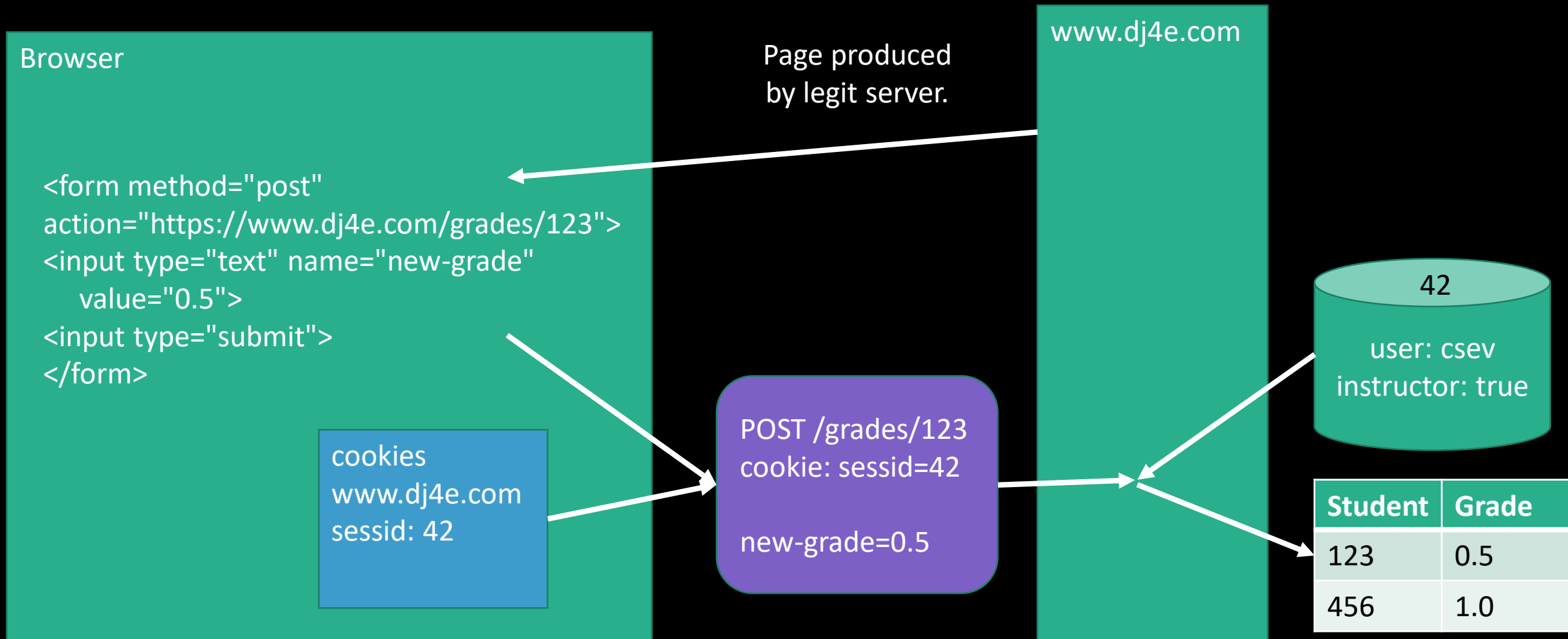
https://en.wikipedia.org/wiki/Cross-site_request_forgery

CSRF Defense

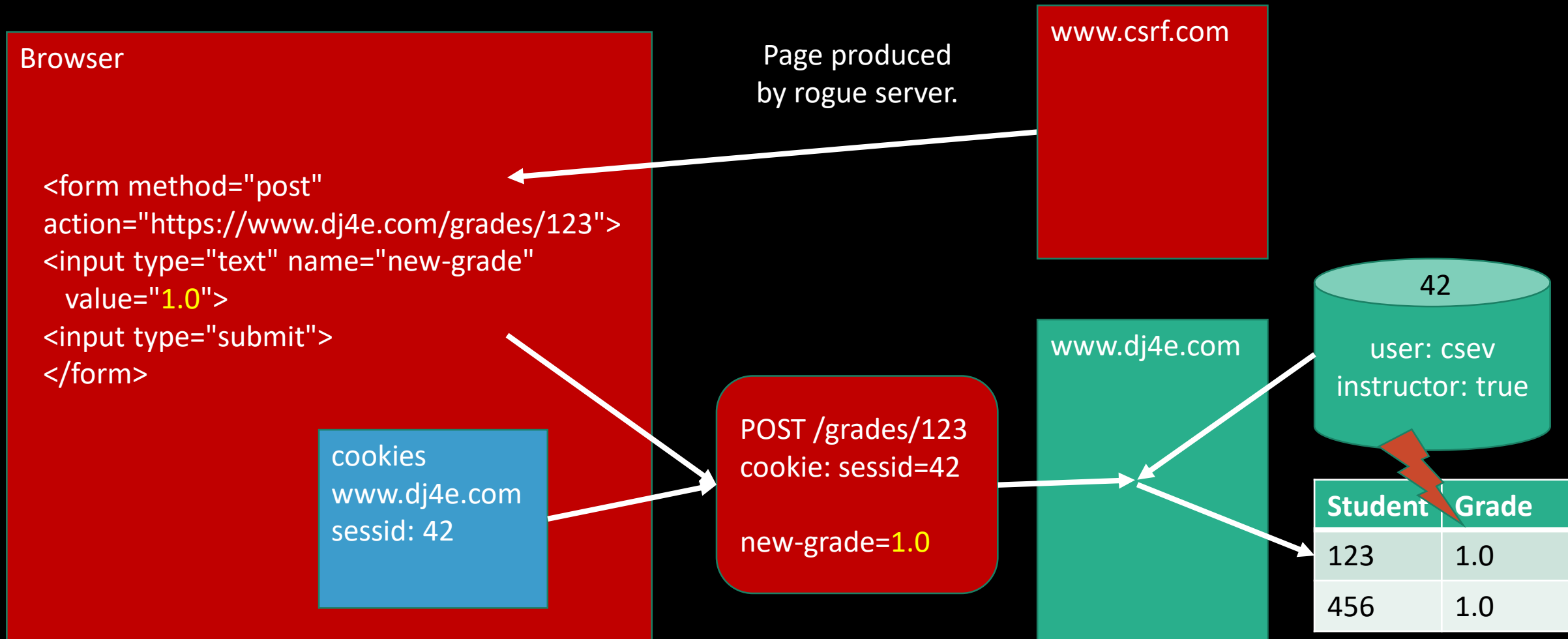
- The legitimate site chooses a large random number (the CSRF Token) and puts it in the session
- When the legitimate site generates a POST form, it includes the CSRF Token as a hidden input field
- When the form is submitted the CSRF Token is sent as well as the cookie
- The site looks up the session and rejects the request if the incoming CSRF Token does not match the session's CSRF Token

https://en.wikipedia.org/wiki/Cross-site_request_forgery

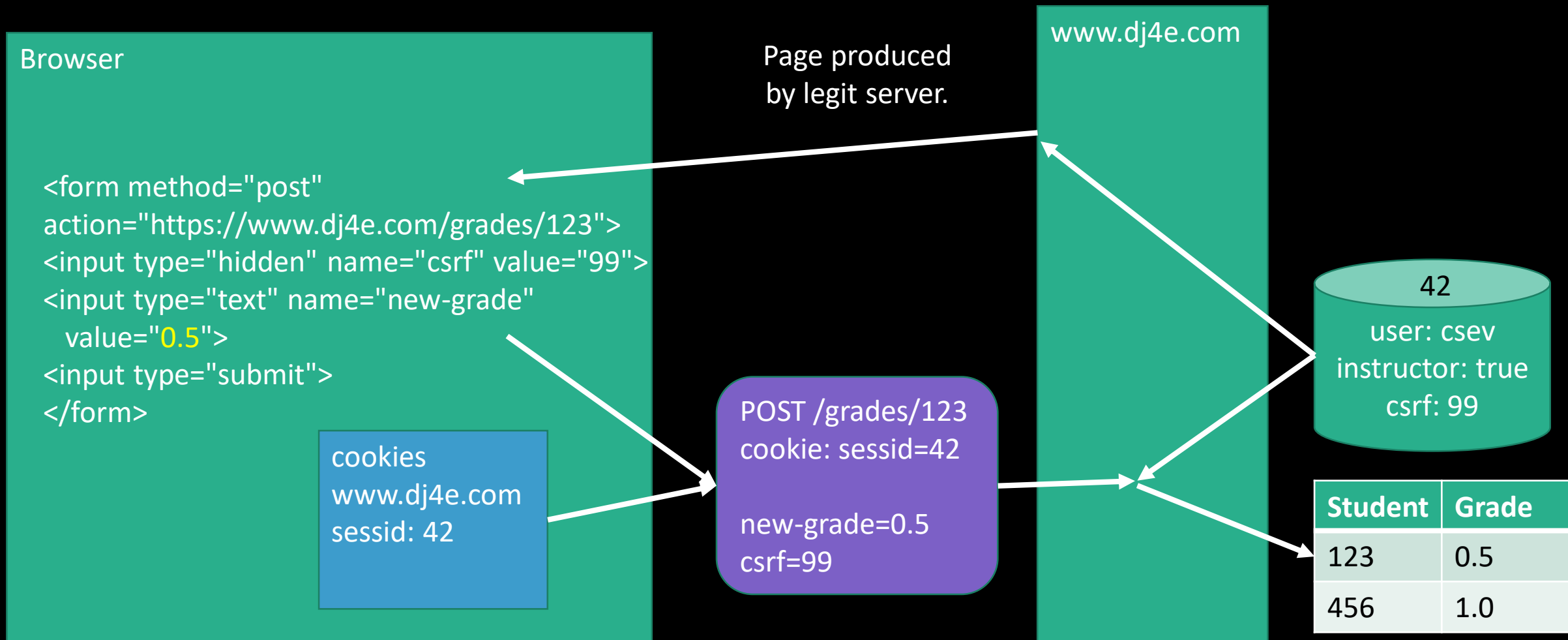
Scenario: Time to Change a Student Grade



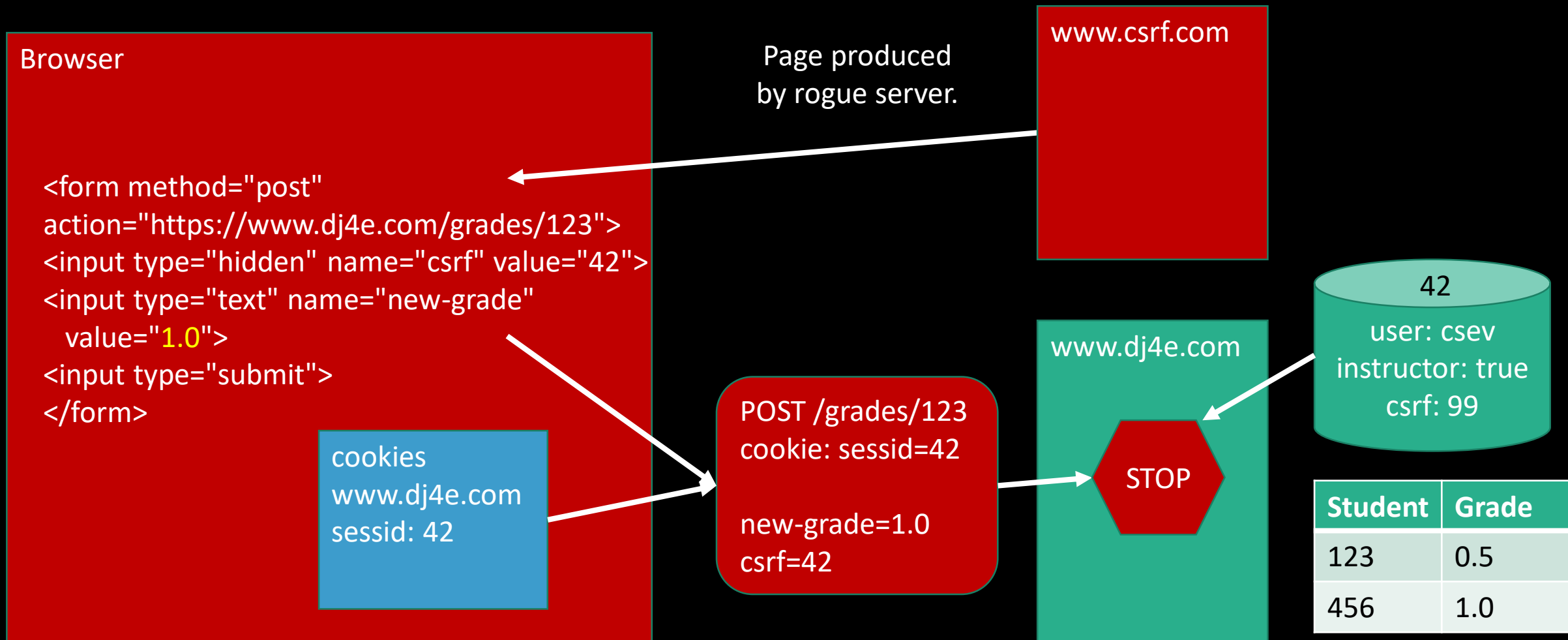
Attack (without CSRF)



With CSRF



CSRF Attack Blocked



4E 403 Forbidden

https://samples.dj4e.com/getpost/failform

Most Visited Drc SakaiCar Sakai Tsugi CRsera Teach LXP UMSI GMU

Forbidden (403)

CSRF verification failed. Request aborted.

Help

Reason given for failure:
CSRF token missing or incorrect.

In general, this can occur when there is a genuine Cross Site Request Forgery, or when [Django's CSRF mechanism](#) has not been used correctly. For POST forms, you need to ensure:

- Your browser is accepting cookies.
- The view function passes a request to the template's [render](#) method.
- In the template, there is a `{% csrf_token %}` template tag inside each POST form that targets an internal URL.
- If you are not using `CsrfViewMiddleware`, then you must use `csrf_protect` on any views that use the `csrf_token` template tag, as well as those that accept the POST data.
- The form has a valid CSRF token. After logging in in another browser tab or hitting the back button after a login, you may need to reload the page with the form, because the token is rotated after a login.

You're seeing the help section of this page because you have `DEBUG = True` in your Django settings and have set that to `False`, and only the initial error message will be displayed.

You can customize this page using the `CSRF_FAILURE_VIEW` setting.

4E 403 Forbidden

https://samples.dj4e.com

Most Visited Drc SakaiCar Sakai Tsugi CRsera Teach

Forbidden (403)

CSRF verification failed. Request aborted.

More information is available with `DEBUG=True`.

Enabling CSRF defense in Django

- Django has built in support to generate, use, and check CSRF Tokens
- Activated by default in `settings.py`

```
MIDDLEWARE = [  
    'django.middleware.security.SecurityMiddleware',  
    'django.contrib.sessions.middleware.SessionMiddleware',  
    'django.middleware.common.CommonMiddleware',  
    'django.middleware.csrf.CsrfViewMiddleware',  
    'django.contrib.auth.middleware.AuthenticationMiddleware',  
    'django.contrib.messages.middleware.MessageMiddleware',  
    'django.middleware.clickjacking.XFrameOptionsMiddleware',  
]
```

CSRF in forms

Remember.....

[dj4e-samples/getpost/views.py](#)

```
@csrf_exempt
def postform(request):
    response = """<p>Impossible POST guessing game...</p>
    <form method="POST">
    <p><label for="guess">Input Guess</label>
    <input type="text" name="guess" size="40" id="guess"/></p>
    <input type="submit"/>
    </form>"""

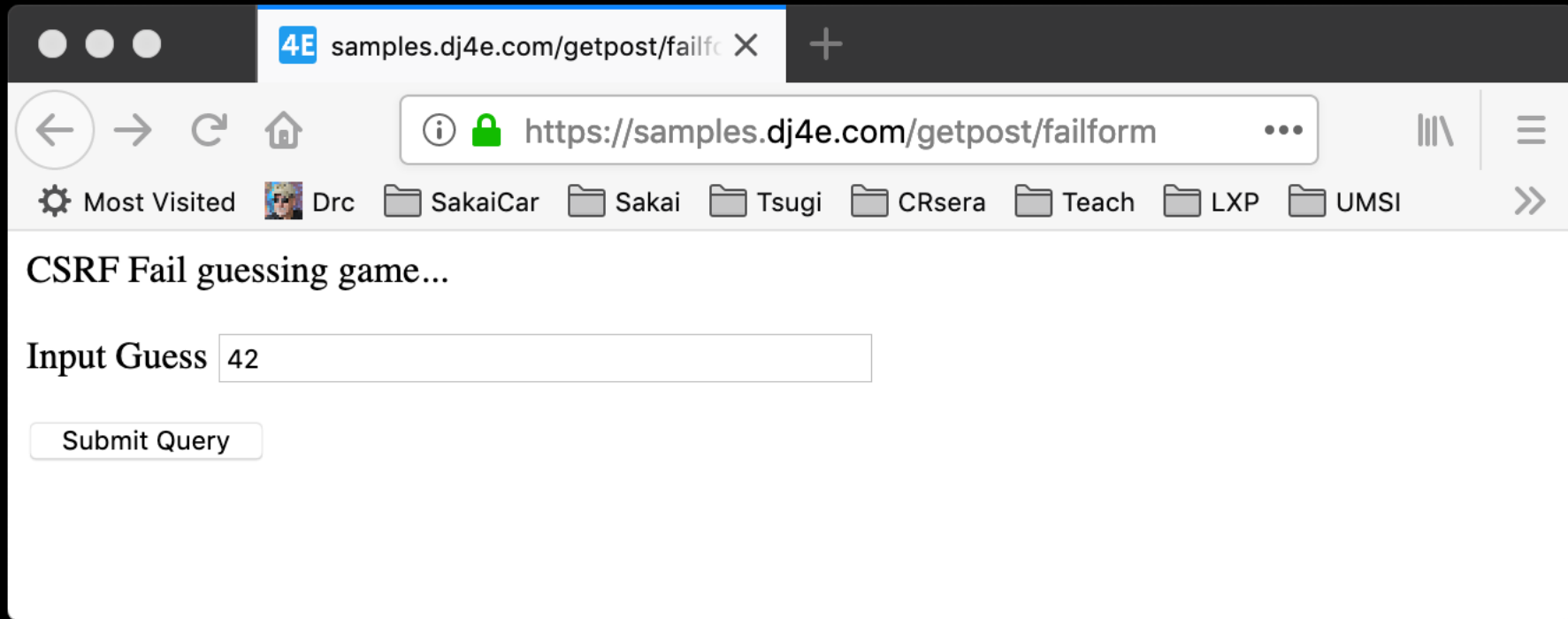
    response += dumpdata('POST', request.POST)
    return HttpResponse(response)
```

<https://samples.dj4e.com/getpost/failform>

```
def failform(request):
    response = """<p>CSRF Fail guessing game...</p>
    <form method="post">
    <p><label for="guess">Input Guess</label>
    <input type="text" name="guess" size="40" id="guess"/></p>
    <input type="submit"/>
    </form>"""

    response += dumpdata('POST', request.POST)
    return HttpResponse(response)
```


<https://samples.dj4e.com/getpost/failform>



4E 403 Forbidden

https://samples.dj4e.com/getpost/failform

Most Visited Drc SakaiCar Sakai Tsugi CRsera Teach LXP UMSI GMU

Forbidden (403)

CSRF verification failed. Request aborted.

Help

Reason given for failure:
CSRF token missing or incorrect.

In general, this can occur when there is a genuine Cross Site Request Forgery, or when [Django's CSRF mechanism](#) has not been used correctly. For POST forms, you need to ensure:

- Your browser is accepting cookies.
- The view function passes a request to the template's [render](#) method.
- In the template, there is a `{% csrf_token %}` template tag inside each POST form that targets an internal URL.
- If you are not using `CsrfViewMiddleware`, then you must use `csrf_protect` on any views that use the `csrf_token` template tag, as well as those that accept the POST data.
- The form has a valid CSRF token. After logging in in another browser tab or hitting the back button after a login, you may need to reload the page with the form, because the token is rotated after a login.

You're seeing the help section of this page because you have `DEBUG = True` in your Django settings and have set that to `False`, and only the initial error message will be displayed.

You can customize this page using the `CSRF_FAILURE_VIEW` setting.

4E 403 Forbidden

https://samples.dj4e.com

Most Visited Drc SakaiCar Sakai Tsugi CRsera Teach

Forbidden (403)

CSRF verification failed. Request aborted.

More information is available with `DEBUG=True`.

<https://samples.dj4e.com/getpost/csrfform>

```
from django.middleware.csrf import get_token

def csrfform(request):
    response = """<p>CSRF Success guessing game...</p>
    <form method="POST">
    <p><label for="guess">Input Guess</label>
    <input type="text" name="guess" size="40" id="guess"/></p>
    <input type="hidden" name="csrfmiddlewaretoken"
        value="__token__"/>
    <input type="submit"/>
    </form>"""

    token = get_token(request)
    response = response.replace('__token__', html.escape(token))
    response += dumpdata('POST', request.POST)
    return HttpResponse(response)
```

4E samples.dj4e.com/getpost/csrf X

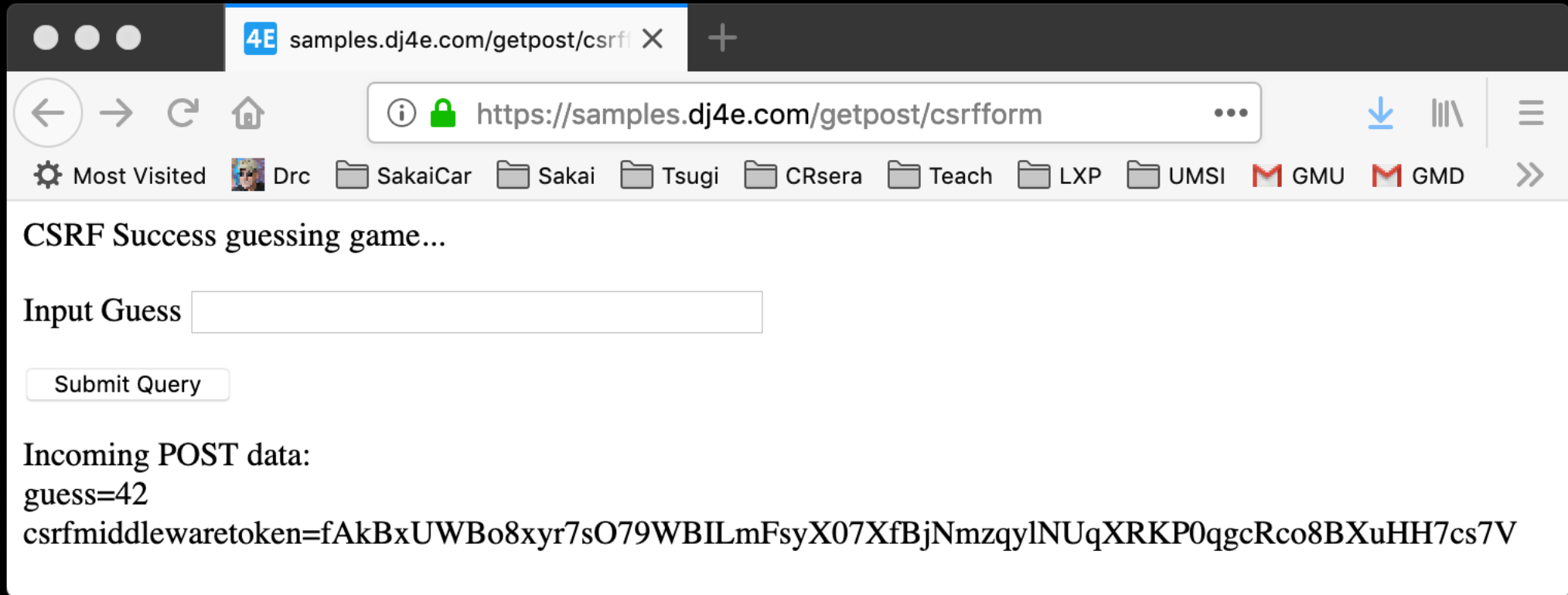
← → ↻ 🏠 <https://samples.dj4e.com/getpost/csrfform> ⌵ ⌵ ⌵

⚙️ Most Visited 🖼️ Drc 📁 SakaiCar 📁 Sakai 📁 Tsugi 📁 CRsera 📁 Teach 📁 LXP 📁 UMSI 📧 GMU 📧 GMD >>

CSRF Success guessing game...

Input Guess

```
<p>CSRF Success guessing game...</p>
<form method="POST">
<p><label for="guess">Input Guess</label>
<input type="text" name="guess" size="40" id="guess"/></p>
<input type="hidden" name="csrfmiddlewaretoken"
  value="fSv596BjrYhRoBkJ008jWm0h3TrTxqiIj5x32K0vXgHaHjSlX33UCJfz52b0CVa2"/>
<input type="submit"/>
</form>
```



Django CSRF in Templates

```
<p>Guessing game</p>
```

[dj4e-samples/getpost/templates/getpost/guess.html](https://github.com/dj4e-samples/getpost/templates/getpost/guess.html)

```
{% if message %}
```

```
<p>{{ message }}</p>
```

```
{% endif %}
```

```
<form method="post">
```

```
<p><label for="guess">Input Guess</label>
```

```
{% csrf_token %}
```

```
<input type="text" name="guess" size="40" id="guess"/></p>
```

```
<input type="submit"/>
```

```
</form>
```

Utility Code for Guesses

[dj4e-samples/getpost/views.py](#)

```
# Call as checkguess('42')
def checkguess(guess) :
    msg = False
    if guess :
        try:
            if int(guess) < 42 :
                msg = 'Guess too low'
            elif int(guess) > 42 :
                msg = 'Guess too high'
            else:
                msg = 'Congratulations!'
        except:
            msg = 'Bad format for guess:' + html.escape(guess)
    return msg
```

```
class ClassyView(View) :
    def get(self, request):
        return render(request, 'getpost/guess.html')

    def post(self, request):
        guess = request.POST.get('guess')
        msg = checkguess(guess)
        return render(request, 'getpost/guess.html', {'message' : msg })
```

[dj4e-samples/getpost/views.py](https://github.com/dj4e/dj4e-samples/blob/master/getpost/views.py)

```
<p>Guessing game</p>
{% if message %}
<p>{{ message }}</p>
{% endif %}
<form method="post">
<p><label for="guess">Input Guess</label>
{% csrf_token %}
<input type="text" name="guess" size="40" id="guess"/></p>
<input type="submit"/>
</form>
```

[dj4e-samples/getpost/templates/getpost/guess.html](https://github.com/dj4e/dj4e-samples/blob/master/getpost/templates/getpost/guess.html)


```
class ClassyView(View) :
    def get(self, request):
        return render(request, 'getpost/guess.html')

    def post(self, request):
        guess = request.POST.get('guess')
        msg = checkguess(guess)
        return render(request, 'getpost/guess.html', {'message' : msg })
```

[dj4e-samples/getpost/views.py](https://samples.dj4e.com/getpost/views.py)



<https://samples.dj4e.com/getpost/classy>

Guessing game

Input Guess

```
<p>Guessing game</p>
```

```
<form method="post">
```

```
<p><label for="guess">Input Guess</label>
```

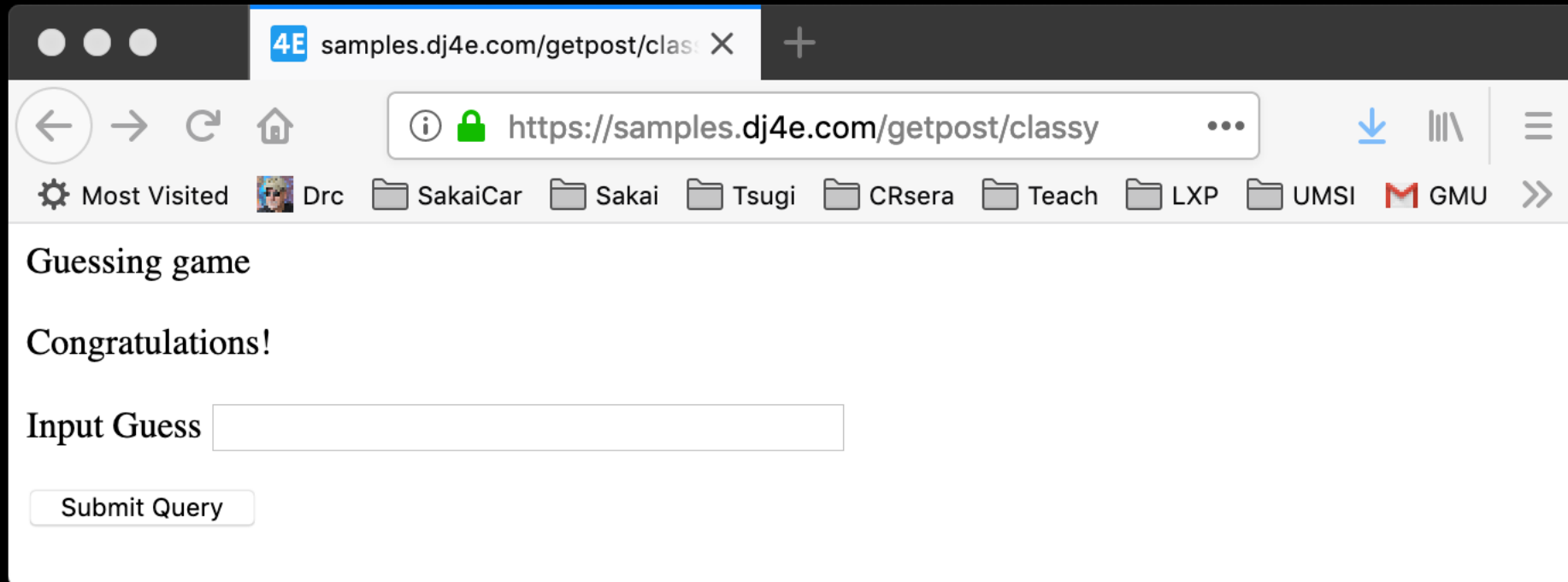
```
<input type="hidden" name="csrfmiddlewaretoken"
value="1oV2XIi9kNx710Lcu9V4rf0TmMsAZm9w5BX0QmHlQ5XqkIjODcQF7CfboVcH4R1Q">
```

```
<input type="text" name="guess" size="40" id="guess"/></p>
```

```
<input type="submit"/>
```

```
</form>
```

<https://samples.dj4e.com/getpost/classy>



Success!!!!

POST-Refresh ... Oops!

Remember this?



Success!!!!

POST / Refresh / 😞

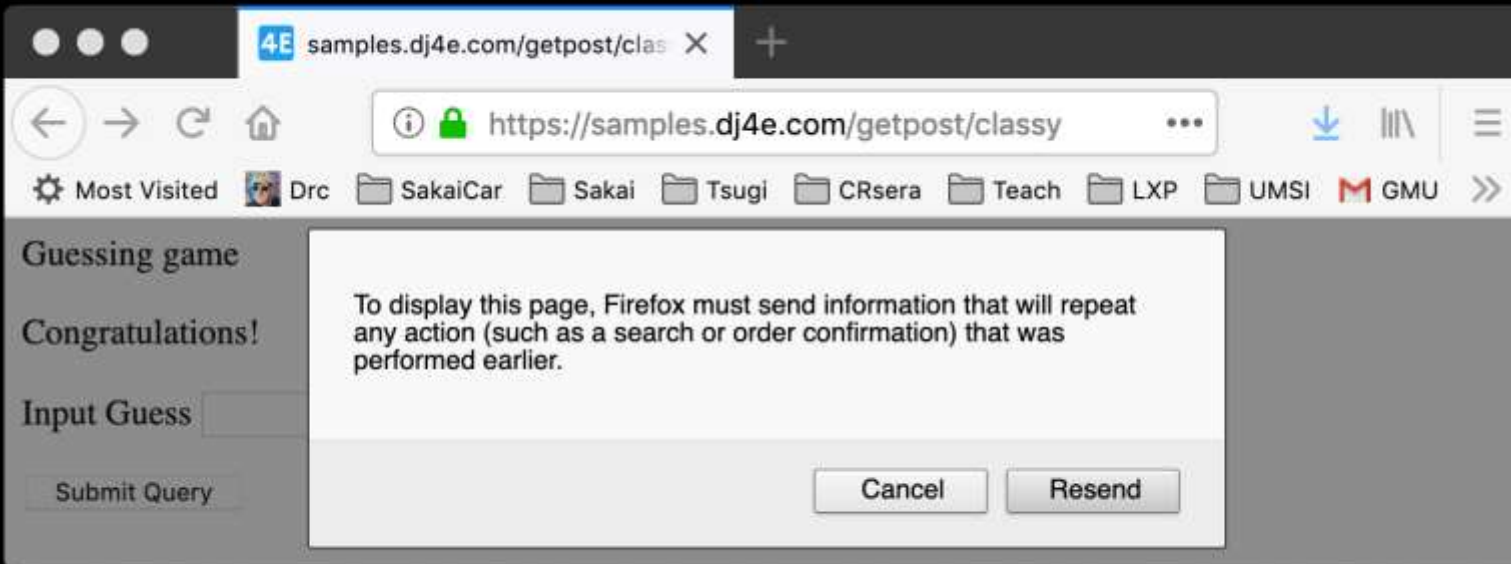
- Once you do a POST and receive 200 status + a page of HTML, if you tell the browser to refresh, the browser will re-send the POST data a second time.
- The user gets a browser pop-up that tries to explain what is about to happen.



Make a POST

See Success

Press Refresh



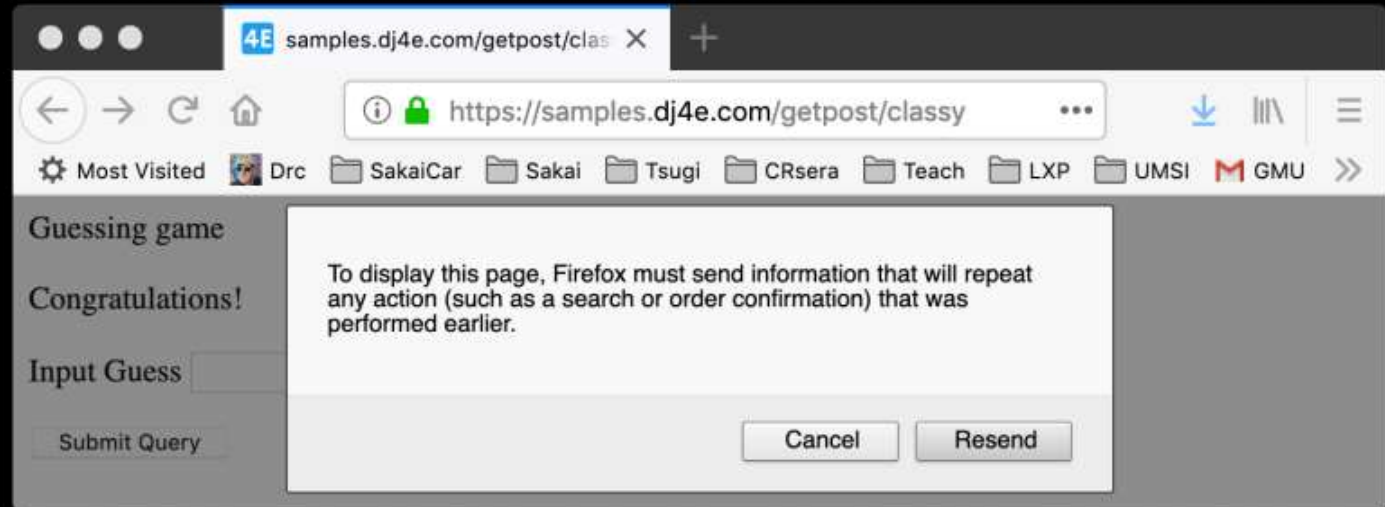
Yucky Message ☹️

Don't Allow Double Posts

- Typically POST requests are adding or modifying data whilst GET requests view data
- It may be dangerous to do the same POST twice (say withdrawing funds from a bank account)
- So the browser insists on asking the user (out of your control)
- Kind of an ugly UX / bad usability
- As developers we work so this never can happen

POST-REDIRECT-GET-Refresh

POST Redirect Rule

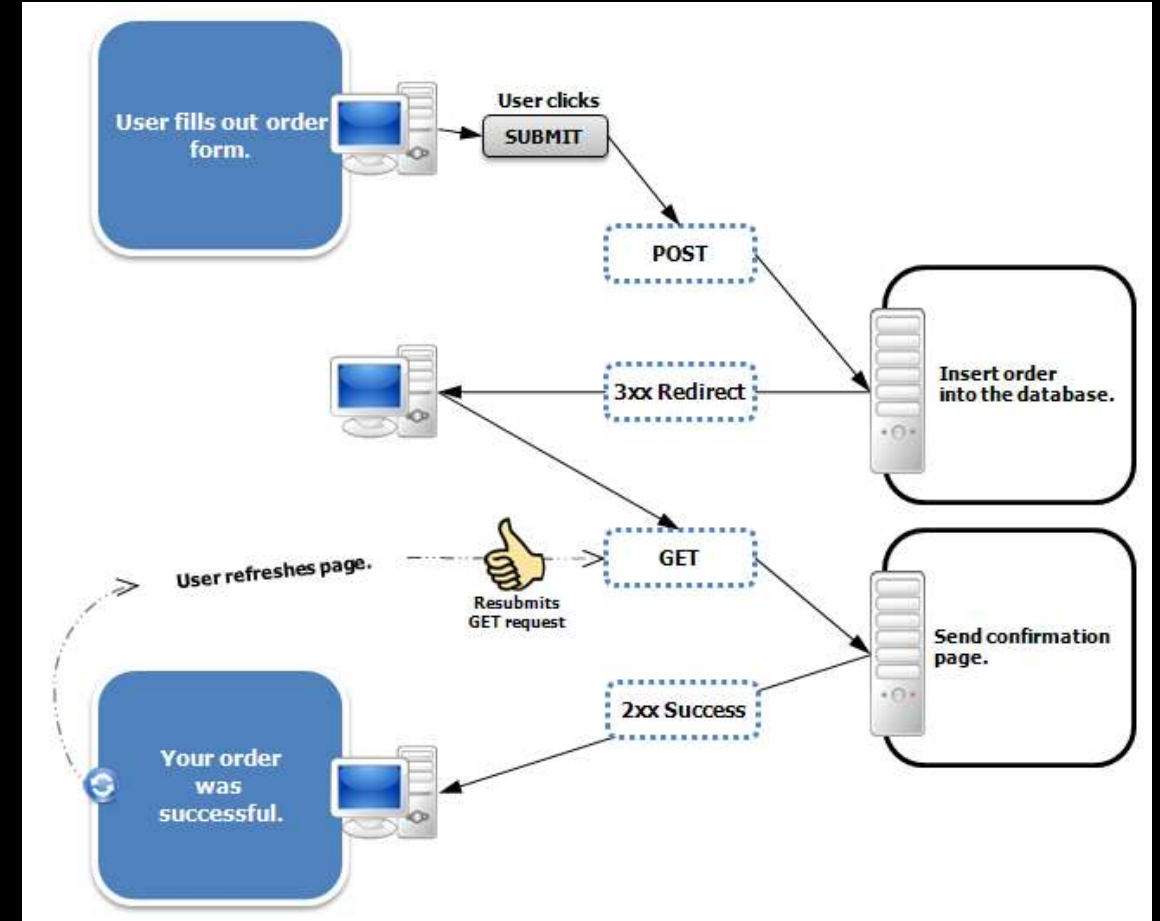
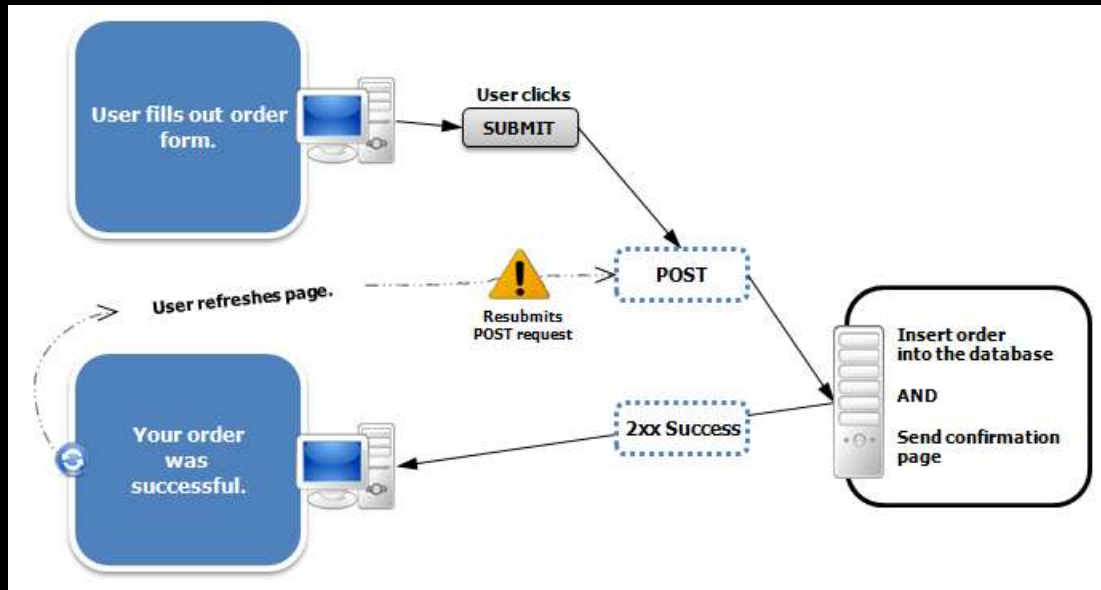


- The simple rule for pages intended for a browser is to never generate a page with HTML content when the app receives POST data and data has been modified
- Must cause a GET by redirecting somewhere - even a GET to the same URL - forcing the browser to make a GET after the POST

Review: HTTP Status Codes

- <http://www.dr-chuck.com/page1.htm> - 200 OK
- <https://samples.dj4e.com/getpost/failform> - 403 Forbidden
 - Post data without CSRF Token
- <http://www.wa4e.com/nowhere.htm> - 404 Not Found
- <http://www.drchuck.com/> - 302 Found / Moved
 - Also known as “redirect”

https://en.wikipedia.org/wiki/List_of_HTTP_status_codes



<https://en.wikipedia.org/wiki/Post/Redirect/Get>

```
class AwesomeView(View) :
    def get(self, request):
        msg = request.session.get('msg', False)
        if ( msg ) : del(request.session['msg'])
        return render(request, 'getpost/guess.html', { 'message' : msg })

    def post(self, request):
        guess = request.POST.get('guess')
        msg = checkguess(guess)
        request.session['msg'] = msg
        return redirect(request.path)
```

[dj4e-samples/getpost/views.py](#)

```
<p>Guessing game</p>
{% if message %}
<p>{{ message }}</p>
{% endif %}
<form method="post">
<p><label for="guess">Input Guess</label>
{% csrf_token %}
<input type="text" name="guess" size="40" id="guess"/></p>
<input type="submit"/>
</form>
```

[dj4e-samples/getpost/templates/getpost/guess.html](#)

<https://samples.dj4e.com/getpost/awesome>



4E samples.dj4e.com/getpost/awe X +

← → ↻ 🏠 ⓘ 🔒 https://samples.dj4e.com/getpost/awesome ⋮

⚙️ Most Visited 🖼️ Drc 📁 SakaiCar 📁 Sakai 📁 Tsugi 📁 CRsera 📁 Teach 📁 LXP 📁 UMSI >>

Guessing game

Input Guess

Enter guess and
press Submit

samples.dj4e.com/getpost/awe X +

https://samples.dj4e.com/getpost/awesome

Most Visited Drc SakaiCar Sakai Tsugi CRsera Teach LXP UMSI IMS Libre Privacy YT IEEE

Guessing game

Congratulations!

Input Guess

Submit Query

Inspector Console Debugger Style Editor Performance Memory Network Storage

Filter URLs || Persist Logs Disable cache No throttling HAR

Status	Method	Domain	File
302	POST	samples.dj4e.com	awesome
200	GET	samples.dj4e.com	awesome
200	GET	samples.dj4e.com	favicon.ico

Headers Cookies Params Response Timings Security

Response headers (524 B) Raw headers

cf-ray: 51e9f22c18fe7e31-DTW

content-type: text/html; charset=utf-8

date: Mon, 30 Sep 2019 23:23:41 GMT

expect-ct: max-age=604800, report-uri="ht....com/cdn-cgi/beacon/expect-ct"

location: /getpost/awesome

server: cloudflare

set-cookie: sessionid=w4eunuecd7uj1qlb1gs1...1209600; Path=/; SameSite=Lax

vary: Cookie

x-clacks-overhead: GNU Terry Pratchett

X-Firefox-Spdy: h2

x-frame-options: SAMEORIGIN

3 requests 15.67 KB / 3.66 KB transferred Finish: 407 ms DOMContentLo

samples.dj4e.com/getpost/awe X +

https://samples.dj4e.com/getpost/awesome

Most Visited Drc SakaiCar Sakai Tsugi CRsera Teach LXP UMSI IMS Libre Privacy YT IEEE

Guessing game

Congratulations!

Input Guess

Submit Query

Inspector Console Debugger Style Editor Performance Memory Network Storage

Filter URLs || ☒ Persist Logs ☒ Disable cache No throttling HAR

Status	Method	Domain	File
302	POST	samples.dj4e.com	awesome
200	GET	samples.dj4e.com	awesome
200	GET	samples.dj4e.com	favicon.ico

Headers Cookies Params Response Timings Security

Preview

Response payload

```
1 <p>Guessing game</p>
2
3 <p>Congratulations!</p>
4
5 <form method="post">
6 <p><label for="guess">Input Guess</label>
7 <input type="hidden" name="csrfmiddlewaretoken" value="9BbU8Z1f
8 <input type="text" name="guess" size="40" id="guess"/></p>
9 <input type="submit"/>
10 </form>
11
```

3 requests 15.67 KB / 3.66 KB transferred Finish: 407 ms DOMContentLoaded

GET after Redirect

samples.dj4e.com/getpost/awe X +

https://samples.dj4e.com/getpost/awesome

Most Visited Drc SakaiCar Sakai Tsugi CRsera Teach LXP UMSI IMS Libre Privacy YT IEEE

Guessing game

Input Guess

Submit Query

Refreshed GET

Inspector Console Debugger Style Editor Performance Memory Network Storage

Filter URLs || ☒ Persist Logs ☒ Disable cache No throttling HAR

Status	Method	Domain	File
302	POST	samples.dj4e.com	awesome
200	GET	samples.dj4e.com	awesome
200	GET	samples.dj4e.com	favicon.ico
200	GET	samples.dj4e.com	awesome
200	GET	samples.dj4e.com	favicon.ico

Headers Cookies Params Response Timings Stack Trace

Preview

Response payload

```
1 <p>Guessing game</p>
2
3 <form method="post">
4 <p><label for="guess">Input Guess</label>
5 <input type="hidden" name="csrfmiddlewaretoken" value="2XRSztgab4
6 <input type="text" name="guess" size="40" id="guess"/></p>
7 <input type="submit"/>
8 </form>
9
```

5 requests 31.01 KB / 6.48 KB transferred Finish: 4.06 min DOMContentL

The response to a POST must be a redirect

- Pass data to the GET – "flash message pattern"
- Session can be used for flash messages

```
class AwesomeView(View) :  
    def get(self, request):  
        msg = request.session.get('msg', False)  
        if ( msg ) : del(request.session['msg'])  
        return render(request, 'getpost/guess.html', {'message' : msg })  
  
    def post(self, request):  
        guess = request.POST.get('guess')  
        msg = checkguess(guess)  
        request.session['msg'] = msg  
        return redirect(request.path)
```

[dj4e-samples/getpost/views.py](#)

Summary

- HTML for Forms
- GET versus POST
- CSRF
- POST Redirect GET

Acknowledgements / Contributions

These slides are Copyright 2019- Charles R. Severance (www.dr-chuck.com) as part of www.dj4e.com and made available under a Creative Commons Attribution 4.0 License. Please maintain this last slide in all copies of the document to comply with the attribution requirements of the license. If you make a change, feel free to add your name and organization to the list of contributors on this page as you republish the materials.

Initial Development: Charles Severance, University of Michigan School of Information

Insert new Contributors and Translators here including names and dates

Continue new Contributors and Translators here

Additional Source Information

- Portions of the text of these slides is adapted from the text www.djangoproject.org web site. Those slides which use text from that site have a reference to the original text on that site. Django is licensed under the three-clause BSD license.