A detailed walk through a CAS authentication

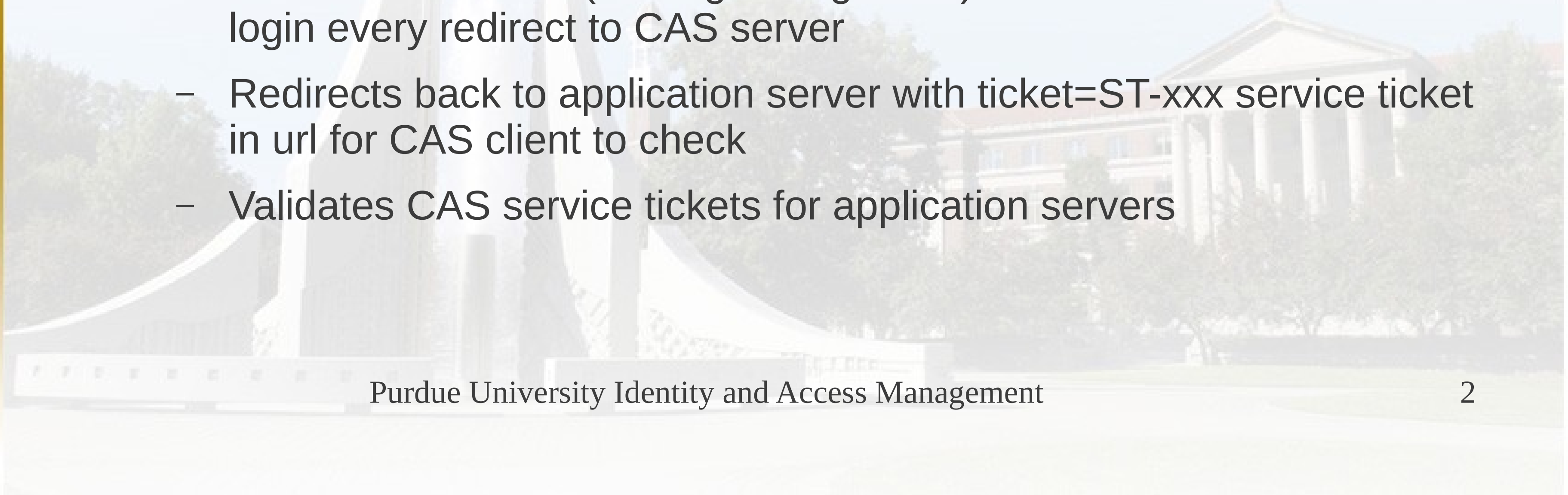
(and how to get your mits on the authenticated user)

- Welcome!
- First of all, what is CAS?
  - Web single sign on
  - Uses “federated” authentication, where all authentication is done by the CAS server, instead of individual application servers
  - The implementation is an open source protocol, open source Java server, and several open source clients
  - Purdue runs a CAS server, configured to authenticate with Purdue Career Account (https://www.purdue.edu/apps/account/cas)
  - As of 4/5/2011, 349 application servers are authorized to check CAS tickets
  - More can be found at:
    - http://www.jasig.org/cas
    - https://www.purdue.edu/apps/account/docs/CAS/CAS_information.jsp

A detailed walk through a CAS authentication

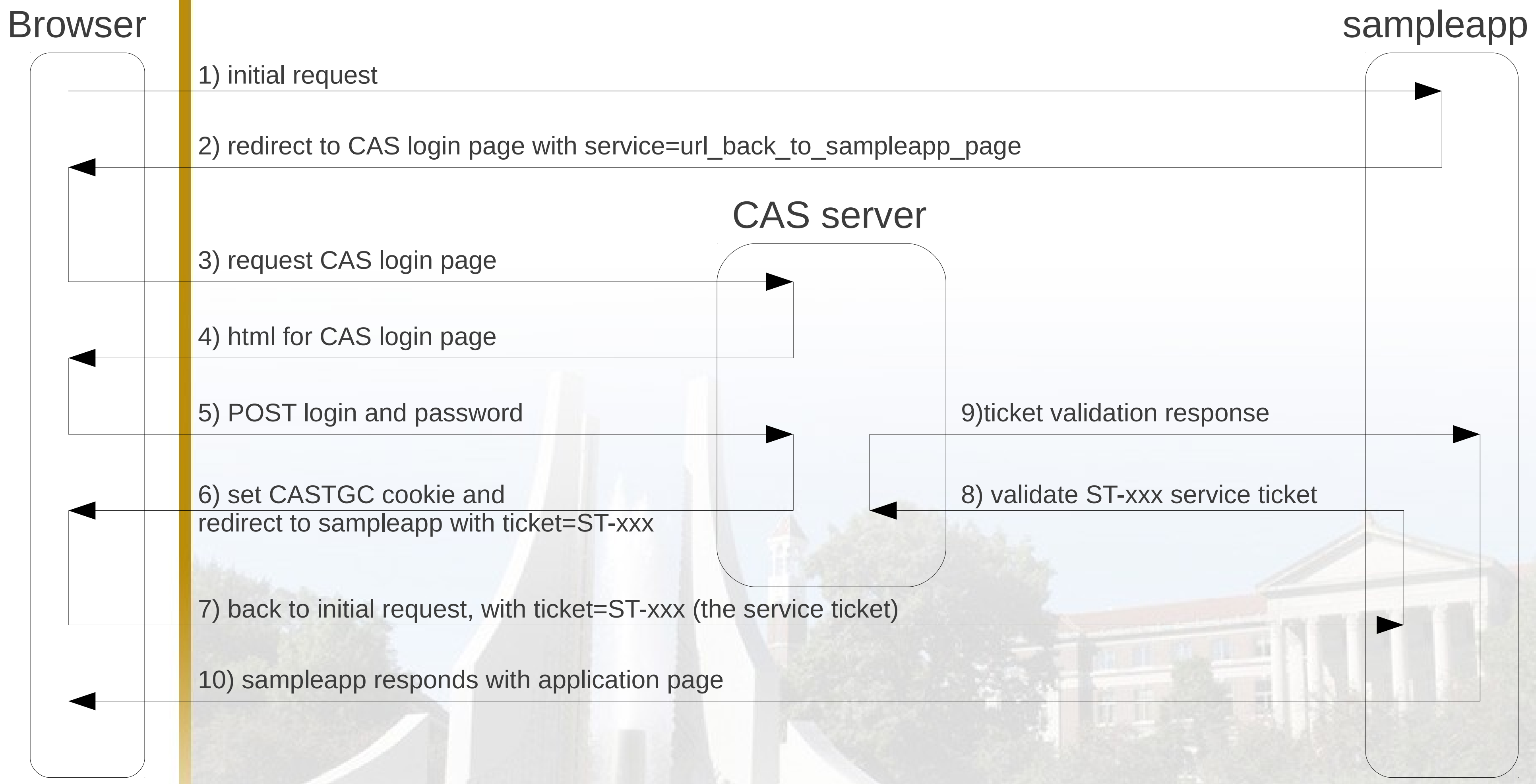
(and how to get your mits on the authenticated user)

- There are three machines in this game
  - a) Browser
  - b) Application server
    - Configured with a CAS client to require authentication for certain urls
  - c) CAS server (http://www.jasig.org/cas)
    - Serves CAS login web page and authenticates users
    - Issues TGT cookie (ticket granting ticket) so user does not have to login every redirect to CAS server
    - Redirects back to application server with ticket=ST-xxx service ticket in url for CAS client to check
    - Validates CAS service tickets for application servers



A detailed walk through a CAS authentication

(and how to get your mits on the authenticated user)

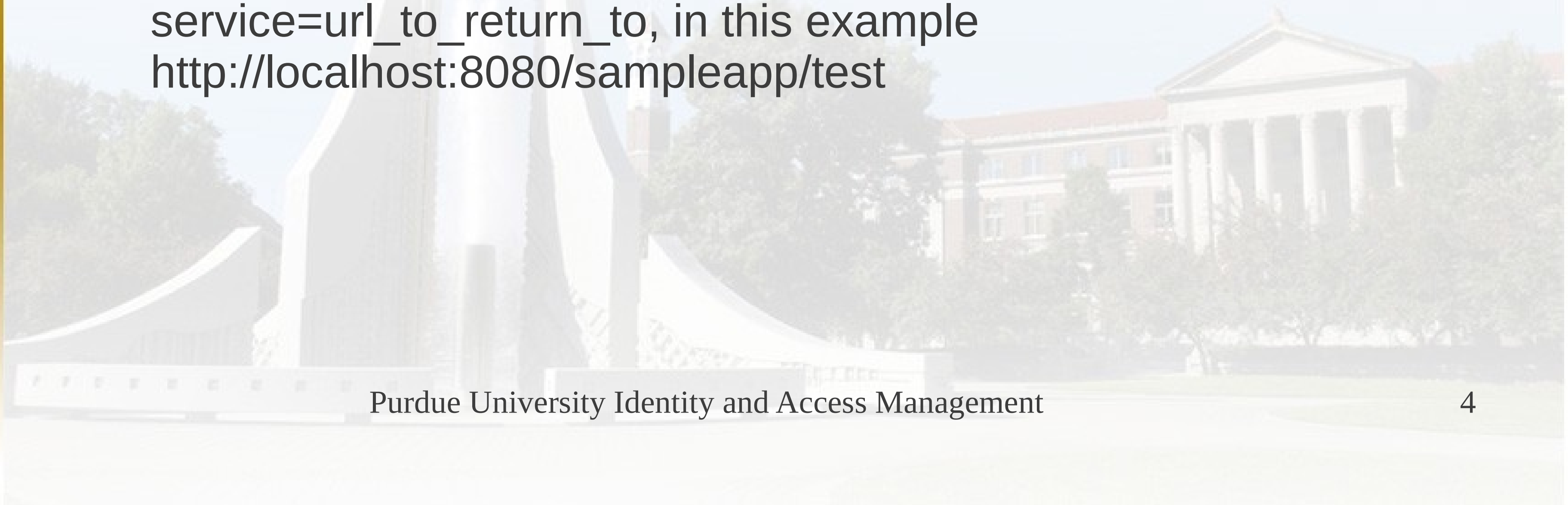


Browser
sampleapp
CAS server
1) initial request
2) redirect to CAS login page with service=url_back_to_sampleapp_page
3) request CAS login page
4) html for CAS login page
5) POST login and password
6) set CASTGC cookie and redirect to sampleapp with ticket=ST-xxx
7) back to initial request, with ticket=ST-xxx (the service ticket)
8) validate ST-xxx service ticket
9)ticket validation response
10) sampleapp responds with application page

A detailed walk through a CAS authentication

(and how to get your mits on the authenticated user)

- Step 1 – initial request
  - “sampleapp” application server is configured with a CAS client to require authentication for certain urls (in this example /test)
  - User with browser accesses /test on sampleapp
  - If browser does not already have session on sampleapp, sampleapp transfers control to the CAS client
  - If the CAS client does not see a ticket parameter in the request, user is redirected back to the CAS login page with service=url_to_return_to, in this example http://localhost:8080/sampleapp/test



A detailed walk through a CAS authentication

(and how to get your mits on the authenticated user)

- Step 2 – redirect to CAS login page
  - User is redirected back to CAS server for authentication
  - Application server (sampleapp) logs

```
2011-03-29 09:16:46,843 DEBUG
[org.jasig.cas.client.authentication.AuthenticationFilter] - <no ticket and no
assertion found>
2011-03-29 09:16:46,843 DEBUG
[org.jasig.cas.client.authentication.AuthenticationFilter] - <Constructed service
url: http://localhost:8080/sampleapp/test/>
2011-03-29 09:16:46,844 DEBUG
[org.jasig.cas.client.authentication.AuthenticationFilter] - <redirecting to
"https://www.purdue.edu/apps/account/cas-server-uber-webapp-3.4.6/login?service=
2F%2Flocalhost%3A8080%2Fsampleapp%2Ftest%2F">

application server access log:
0:0:0:0:0:0:0:1 - - [29/Mar/2011:09:16:46 -0400] "GET /sampleapp/test/ HTTP/1.1"
302 -
```

A detailed walk through a CAS authentication

(and how to get your mits on the authenticated user)

- Step 3 – browser requests CAS login page
  - CAS server checks for its CASTGC cookie (ticket granting ticket), if it's there, user is already authenticated via CAS, skip to step 6 and redirect back to sampleapp with a service ticket
  - If no CASTGC is present, serve browser the CAS login page
  - CAS server access log:

```
0:0:0:0:0:0:0:1 - - [29/Mar/2011:09:16:47 -0400] "GET /cas-server-uber-webapp-
3.4.6/login?service=http%3A%2F%2Flocalhost%3A8080%2Fsampleapp%2Ftest%2F HTTP/1.1"
200 6935
```

A detailed walk through a CAS authentication

(and how to get your mits on the authenticated user)

- Step 4 – CAS server sends login page to browser
  - This is nice because application servers do not need to
    - maintain their own login page
    - maintain login/password credentials to do the actual authentication
    - even see the password, it's between the browser and CAS server

A detailed walk through a CAS authentication

(and how to get your mits on the authenticated user)

- Step 5 – browser POSTs login/password to CAS server
  - CAS server checks login and password, if authentication fails serve another login page to browser
  - Too many unsuccessful authentication attempts in a short period of time will result in a “lockout”, where authentication will always fail for a 15 minute lockout period
  - CAS server access log:

```
0:0:0:0:0:0:0:1 - - [29/Mar/2011:09:16:52 -0400] "POST /cas-server-uber-webapp-
3.4.6/login?service=http%3A%2F%2Flocalhost%3A8080%2Fsampleapp%2Ftest%2F HTTP/1.1"
302 -
```

A detailed walk through a CAS authentication

(and how to get your mits on the authenticated user)

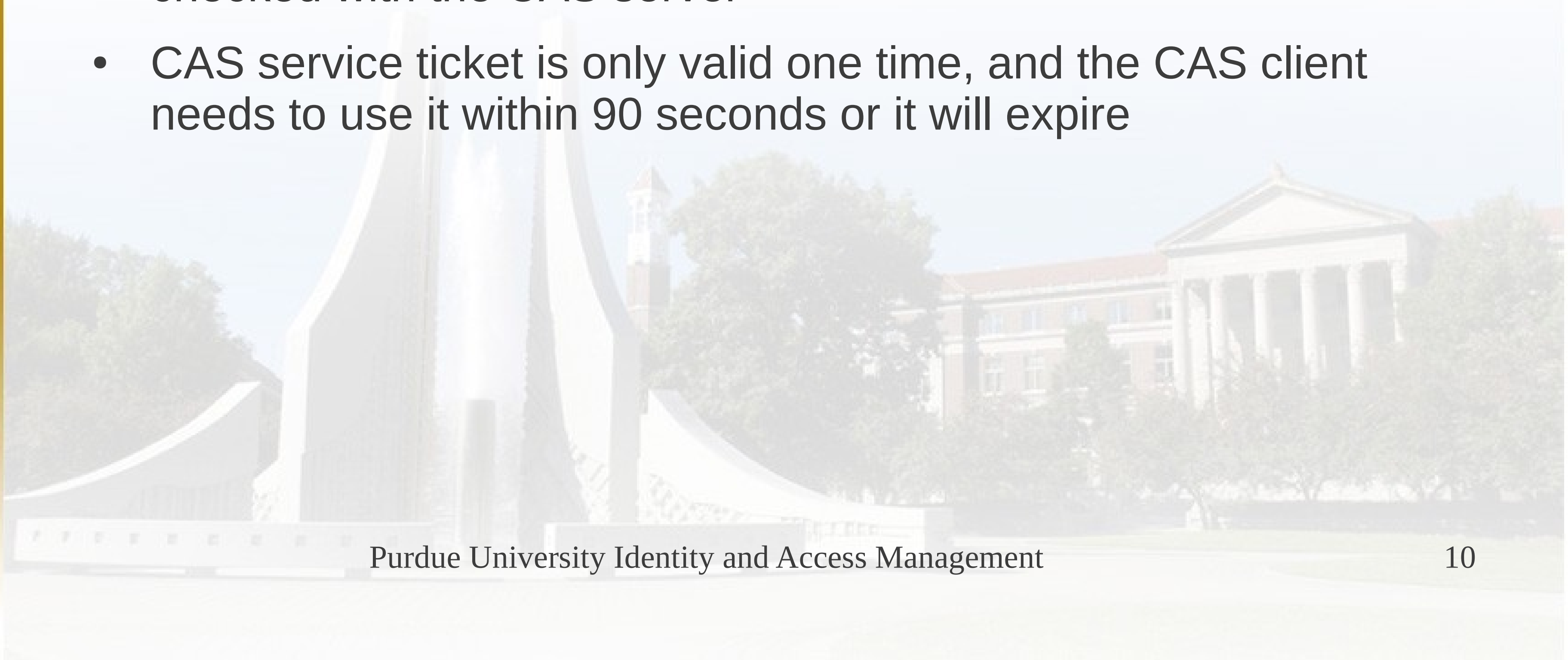
- Step 6 – CAS server redirects back to application server
  - A ticket granting ticket TGT-xxx is stored on the CAS server, and set as a CASTGC cookie
  - A service ticket is issued for the application (http://localhost:8080/sampleapp/test/) and sent as a parameter back to the application server

```
2011-03-29 09:16:52,208 DEBUG
[org.jasig.cas.web.support.CookieRetrievingCookieGenerator] - <Added cookie with
name [CASTGC] and value [TGT-1-wKQjkOhweJE6MMTNCqTwv6WojMDBL61GISejnyCfigrMFCumYu-
cas]>
2011-03-29 09:16:52,214 DEBUG [org.jasig.cas.ticket.registry.DefaultTicketRegistry]
- <Added ticket [ST-1-bdgbwHIReBonmaudvxJl-cas] to registry.>
2011-03-29 09:16:52,214 INFO [org.jasig.cas.CentralAuthenticationServiceImpl] -
<Granted service ticket [ST-1-bdgbwHIReBonmaudvxJl-cas] for service
[http://localhost:8080/sampleapp/test/] for user [jott]>
```

A detailed walk through a CAS authentication

(and how to get your mits on the authenticated user)

- Step 7 – browser re-requests url from application server, with a CAS service ticket
  - Application server still has not yet established a session, so CAS client takes control
  - CAS client sees a ticket parameter in the url, that can be checked with the CAS server
  - CAS service ticket is only valid one time, and the CAS client needs to use it within 90 seconds or it will expire



A detailed walk through a CAS authentication

(and how to get your mits on the authenticated user)

- Step 8 – application server checks CAS service ticket sent by browser in url
  - CAS client preparing to check service ticket:

```
2011-03-29 09:16:52,231 DEBUG
[org.jasig.cas.client.validation.Cas20ProxyReceivingTicketValidationFilter] -
<Attempting to validate ticket: ST-1-bdgbwHIReBonmaudvxJl-cas>
2011-03-29 09:16:52,232 DEBUG
[org.jasig.cas.client.validation.Cas20ServiceTicketValidator] - <Constructing
validation url: https://www.purdue.edu/apps/account/cas-server-uber-webapp-
3.4.6/serviceValidate?ticket=ST-1-bdgbwHIReBonmaudvxJl-cas&service=http%3A%2F
%2Flocalhost%3A8080%2Fsampleapp%2Ftest%2F>
```

  - CAS server access log:

```
127.0.0.1 - - [29/Mar/2011:09:16:52 -0400] "GET /cas-server-uber-webapp-
3.4.6/serviceValidate?ticket=ST-1-bdgbwHIReBonmaudvxJl-cas&service=http%3A%2F
%2Flocalhost%3A8080%2Fsampleapp%2Ftest%2F HTTP/1.1" 200 281
```

A detailed walk through a CAS authentication

(and how to get your mits on the authenticated user)

- Step 9 – CAS server responds to ticket check
  - CAS server response (notice the NEW attributes!):

```
2011-03-29 09:16:52,327 DEBUG
[org.jasig.cas.client.validation.Cas20ServiceTicketValidator] - <Server response:
<cas:serviceResponse xmlns:cas='http://www.yale.edu/tp/cas'>
     <cas:authenticationSuccess>
          <cas:user>jott</cas:user>
          <cas:attributes>
               <cas:email>jott@purdue.edu</cas:email>
               <cas:i2a2characteristics>0,3592,2000</cas:i2a2characteristics>
               <cas:lastname>Ott</cas:lastname>
               <cas:firstname>Jeffrey A</cas:firstname>
               <cas:fullname>Jeffrey A Ott</cas:fullname>
               <cas:puid>0012345678</cas:puid>
          </cas:attributes>
     </cas:authenticationSuccess>
</cas:serviceResponse>
>
```

  - You can test this now yourself against the new CAS server version 3.4.6 (which will become production in May 2011):

```
https://www.purdue.edu/apps/account/cas-server-uber-webapp-3.4.6/login
https://www.purdue.edu/apps/account/cas-server-uber-webapp-3.4.6/serviceValidate
```

A detailed walk through a CAS authentication

(and how to get your mits on the authenticated user)

- Step 10 – application server sends requested page
  - Some CAS clients (including the Java CAS client) can be configured to redirect the browser to the same url, but without the ticket parameter
  - Application server access log:

```
0:0:0:0:0:0:0:1 - - [29/Mar/2011:09:16:52 -0400] "GET /sampleapp/test/?ticket=ST-1-
bdgbwHIReBonmaudvxJl-cas HTTP/1.1" 302 -
0:0:0:0:0:0:0:1 - - [29/Mar/2011:09:16:52 -0400] "GET /sampleapp/test/ HTTP/1.1"
200 202
```

A detailed walk through a CAS authentication

(and how to get your mits on the authenticated user)

- Java CAS client
  - https://wiki.jasig.org/display/CASC/CAS+Client+for+Java+3.1
  - Previous example used version 3.1.10
  - Looking at one CAS client will help understand how any of them will need configured
  - Next two slides show the web.xml to configure the Java CAS client for the previous example:

A detailed walk through a CAS authentication

(and how to get your mits on the authenticated user)

```
<filter>
     <filter-name>CAS Authentication Filter</filter-name>
     <filter-class>org.jasig.cas.client.authentication.AuthenticationFilter</filter-class>
     <init-param>
          <param-name>casServerLoginUrl</param-name>
          <param-value>https://www.purdue.edu/apps/account/cas-server-uber-webapp-3.4.6/login</param-value>
     </init-param>
     <init-param>
          <param-name>serverName</param-name>
          <param-value>http://localhost:8080</param-value>
     </init-param>
</filter>
<filter>
     <filter-name>CAS Validation Filter</filter-name>
     <filter-class>org.jasig.cas.client.validation.Cas20ProxyReceivingTicketValidationFilter</filter-class>
     <init-param>
          <param-name>casServerUrlPrefix</param-name>
          <param-value>https://www.purdue.edu/apps/account/cas-server-uber-webapp-3.4.6</param-value>
     </init-param>
     <init-param>
          <param-name>serverName</param-name>
          <param-value>http://localhost:8080</param-value>
     </init-param>
     <init-param>
          <param-name>redirectAfterValidation</param-name>
          <param-value>true</param-value>
     </init-param>
     <init-param>
          <param-name>exceptionOnValidationFailure</param-name>
          <param-value>false</param-value>
     </init-param>
</filter>
```

A detailed walk through a CAS authentication

(and how to get your mits on the authenticated user)

- Continued web.xml for Java CAS client configuration:

```
<filter>
     <filter-name>CAS HttpServletRequest Wrapper Filter</filter-name>
     <filter-class>org.jasig.cas.client.util.HttpServletRequestWrapperFilter</filter-class>
</filter>

<filter-mapping>
     <filter-name>CAS Authentication Filter</filter-name>
     <url-pattern>/test/*</url-pattern>
</filter-mapping>
<filter-mapping>
     <filter-name>CAS Validation Filter</filter-name>
     <url-pattern>/test/*</url-pattern>
</filter-mapping>
<filter-mapping>
     <filter-name>CAS HttpServletRequest Wrapper Filter</filter-name>
     <url-pattern>/test/*</url-pattern>
</filter-mapping>
```

A detailed walk through a CAS authentication

(and how to get your mits on the authenticated user)

- CAS is not just for web applications
  - Browsers hold CAS state with a cookie (called CASTGC that holds a CAS ticket granting ticket – TGT), but any client, such as a mobile app, can obtain and store a TGT
  - See https://wiki.jasig.org/display/CASUM/RESTful+API
  - Example:

```
POST a username and password to https://CAS_SERVER_URL/v1/tickets
(with “Accept: text/plain” as a header)

And if the login/password check out, the server sends back

201 Created
Location: https://CAS_SERVER_URL/v1/tickets/{TGT id}

If authentication fails, the server returns back a 400 code
```

A detailed walk through a CAS authentication

(and how to get your mits on the authenticated user)

- Initiatives for later this year:
  - Ability to use Boilerkey for CAS authentication:
    - If Boilerkey is used, CAS server will expose an extra attribute returned by the ticket check that indicates that the authentication was a Boilerkey authentication

    `https://www.purdue.edu/apps/account/IAMO/Purdue_CareerAccount_BoilerKey.jsp`
  - Separate mobile app CAS login page
  - Application server administrators will be able to manage CAS ticket check server lists via web page
  - Check for more at:
    - https://www.purdue.edu/apps/account/docs/CAS/CAS_information.jsp

A detailed walk through a CAS authentication

(and how to get your mits on the authenticated user)

- Thanks for your attention!
- Questions?
- Purdue Identity and Access Management can be reached at accounts@purdue.edu
- Please fill out an evaluation at http://www.itap.purdue.edu/boilerweb/survey