***Next Meeting – Friday, June 27, 2014; 10:00 a.m. PST***

Next meeting will be using Zoom instead of CCCConfer

Join from a PC, Mac, iPad, iPhone or Android device:

Please click this URL to start or join. <https://zoom.us/j/195486446>

Or, go to <https://zoom.us/join> and enter meeting ID: 195 486 446

Join from dial-in phone line:

Dial: +1 (415) 762-9988 or +1 (646) 568-7788

Meeting ID: 195 486 446

Participant ID: Shown after joining the meeting

***Project Websites:*** <https://sites.google.com/site/pescedexchange/>

***And*** [***GitHub\_PESC\_CDS***](https://github.com/pescCDS/cdsWebserver)

Attendees:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Tim Calhoon | CCC Tech Center / Co-chair | X |  | Mark Cohen | Parchment / Co-chair |  |
| Doug Falk | NSC / Co-chair | X |  | Monterey Sims | U. of Phoenix |  |
| Lenny Robison | CCC Tech Center | X |  | Emmett Culley | CCC Tech Center |  |
| Darius Robinson | Parchment | X |  | Wes Owen | CCC Tech Center | X |
| Jason Weaver | Parchment | X |  | John DiPirro | CSIS | X |
| Dave Landry | Clearinghouse | X |  | Jam Hamidi | BC Campus | X |
| Matt Samuel | Clearinghouse | X |  | Jeff Elliott | Univ. of Missouri | X |
| Nick Nelson | U. of Phoenix |  |  |  |  |  |

**Agenda:**

* Discuss licensing and have the contributors get releases signed.

See <http://www.apereo.org/licensing>

* Individual Contributor License Agreement (ICLA)
* Corporate Contributor License Agreement (CCLA)
* Status of server hosting @ CCC Tech Center
* Status of development work
* Plans for next steps

**Notes, June 13, 2014;**

1. Tim Calhoon chaired the meeting
2. CCC Tech Center is completing the agreement of behalf of their developers
3. Tim encouraged others to complete agreements

See <http://www.apereo.org/licensing>

* Individual Contributor License Agreement (ICLA)
* Corporate Contributor License Agreement (CCLA)

1. Next Steps:
   1. Developers agreed to create a tech stack of jobs to pick from and collaborate on GitHub
   2. First tasks
      1. Build detailed for building in GitHub
      2. Build directory server and network server
         1. Syst Mgr Interface
         2. CDS Service
         3. Vendor interface
         4. Client interface
      3. Build IT requirements
      4. I believe Matt and Wes were beginning on this
   3. Use GitHub for issue tracking - decision
   4. Still need developer recommendations for splitting work components
   5. Based on developer input organize reasonable project plan

Q&A during meeting:

1.) Build Tool - The foundation has selected Maven as its build tool of choice, I assume we are using the latest version of maven? (maven 3.2.1 at the time of writing).  Do we have an artifact repository that we are deploying our built artifacts to?  I’d assume we don’t, but I want to check.

Maven 3

Also, are we using Jdk 1.6 or something newer?

Version 1.7

2.) Dependencies - Here are the questions I have about our dependencies

* First and foremost, which version of Spring Framework should we use?  Current stable version 4.0.5?  Was the current codebase built with 3.x in mind?

Try to remain current. Maintain list on GitHub with dependencies. Communications between developers.

* Next, Apache CXF, which is basically the core technology for creating the web services, what version should we use?  The current 3.0.0 version was just released last month.  Again was the current codebase built with an earlier version in mind?

Current version (non-beta)

* I need to know what version of DataNucleus/Hibernate to include in the pom?

Current version (non-beta)

* I need to know what version of MySql we are targeting so I can wire in the appropriate JDBC driver.

Current version (non-beta)

3.) Configuration - Here are the questions around configuration

* Are we defining JNDI resources in the Tomcat context?  Or are we doing direct jdbc configuration?  If so, I feel compelled to ask what connection pooling library we want to use? (C3P0? etc)

Yes.

* I’d assume everything should be annotation based, ie Hibernate annotations, Spring annotations, etc?

Yes.

* If we are using Liquibase, do we want to enable the maven hooks?  I’d assume we want to be able to build our database from scratch as needed.

Yes, enable Maven hooks.

When is comes to Spring Security, really this is quite a large question.  What types of security are we looking to support?  Just basic auth? Or something more robust?  This is less a technical decision, and while I don’t think we need to worry about it right away, it does directly impact how our spring configurations are wired together.

More of an implementers decision.