The Apache Software Foundation
Community Development
Mentoring Programme
India ICFOSS

Luciano Resende (lresende@apache.org)
Community Development, at Apache Software Foundation
Welcome!

- Get to know each other
  - What would you like to gain out of this workshop?
About me

Professionally

- Joined IBM Brazil in 2000 as Senior Consultant on Lotus Products
- Relocated to IBM US in 2001 to work on Lotus Organization
- In 2006, joined the IBM AIM Organization, and started working on new technologies for WebSphere Application Server in open source
- Left IBM in 2010 and joined the Platform & Architecture team at Shutterfly as Architect/Manager.
- In 2012 rejoined IBM as Senior Software Engineer at IBM Big Data & Analytics group

Open Source

- Started working on open source as part of my day job at IBM in 2006 working on Apache Tuscany.
- Currently chair for the Community Development PMC and Wink PMC.
- Have created and mentored various projects throughout Incubation
Day 1 Agenda

Day 1: June 21st 2013

- Learn about
  - Open Source
  - Open Source Licenses
  - Open Source Communities
  - Apache Software Foundation
  - How to get involved with an Apache project
  - Apache Incubator
    - Creating new projects
Day 2 Agenda

Day 2: June 22nd 2013

- Development hands-on
  - Setup development environment
    - Subversion
    - Git
    - Maven
- Development tools overview
- Hands-on exercises
Day 3 Agenda

Day 3: June 23rd 2013
- About Mentoring
- Expectations from students
- Working on project proposals
Post sessions Follow Up & Questions

- Interaction will be through mailing list and open source
- Students Mailing List:
  - students-subscribe@community.apache.org
  - students@community.apache.org

- This presentation is available for download at:
OPEN SOURCE
Topics

- General knowledge about open source
  - Importance of Open Source
  - How it got started
  - What is Open Source
  - License
  - Infrastructure
  - Community
Why Open Source ?
Why open source?

It’s “impossible to avoid”

Gartner 2007 Study:
By 2011, 80% of all commercial software will contain open source code

-“Open source impossible to avoid, Gartner says”, Network World

Forrester 2008 Study:
Study of 2,252 North American and European Software decision makers
done by Forrester: 66% are interested in open-source software.…
Open source seems to be a tactic for achieving the high priority
initiatives.

Web 2.0 technologies (such as blogs, wikis and RSS) and
Service Oriented Architecture among their major software initiatives.

-“Open source software: Just a Means to an End”, CIO magazine, March 2008
Why is Open Source important?

- Can be a major source of innovation!
  - It unites perspectives from a host of disciplines and brings ideas together from all around the world to
    - Rapidly solve business issues
    - Accelerate technological advancements
    - Stimulate economic growth
    - Enable new business models

- OSS is a good approach for driving emerging open standards
  - Popular open source projects can become the common implementations

- IT can benefit
  - Increased choice and flexibility
  - Lower costs
  - Quick response
Major Players
Serious Users

citibank

Nationwide

salesforce

PAYCHEX

NETFLIX

Microsoft

THE WHITE HOUSE
WASHINGTON

WELLS FARGO
Some Statistics

- $1.3 billion Red Hat 2012 revenue
- 3 million GitHub projects
- 80% commercial products with open source code
- 66% market share for Apache HTTP, Nginx and Lighttpd
- > 50% market share for Firefox and Chrome
- 50% blue chip companies fully committed to open source
- 29% growth rate for Android
HOW IT ALL STARTED?
GNU Emacs and GCC

Patch and Perl

1985 May - Original patch program written.

```
my $a = "foo";
if ($some_condition) {
    my $b = "bar";
    print $a;
    print $b;  # prints "foo"  # prints "bar"
}
print $a;
print $b;  # prints "foo"  # prints nothing
```
World Wide Web

The WorldWideWeb (W3) is a wide-area hypermedia information retrieval initiative aiming to give universal access to a large universe of documents.

Everything there is online about W3 is linked directly or indirectly to this document, including an executive summary of the project, Mailing lists, Policy, November's W3 news, Frequently Asked Questions.

What's out there?
Pointers to the world's online information, subjects, W3 servers, etc.

Help
on the browser you are using

Software Products
A list of W3 project components and their current state. (e.g. Line Mode, X11 Viola, NeXTStep, Servers, Tools, Mail robot, Library)

Technical
Details of protocols, formats, program internals etc

Bibliography
Paper documentation on W3 and references.

People
A list of some people involved in the project.

History
A summary of the history of the project.

How can I help?
If you would like to support the web..

Getting code
Getting the code by anonymous FTP, etc.
Linus - Unix = Linux

Deb + Ian = Debian
McCool hangs five

1994 May - NCSA loses its web developer.
The band gets together
1995 Jun 8 - PHP released publicly by Rasmus Lerdorf.
Mozilla escapes
Foundations coalesce
Wikipedia spins off

2010 Sep 28 - Several members of the OpenOffice.org project form a new group called "The Document Foundation".
WHAT IS OPEN SOURCE?
What Is Open Source?

- open source refers to software that is published under licenses that defines how source code can be made available to everyone to inspect, change, download, and explore as they wish.

- Open source is not a software methodology
  - It is a way of developing ideas and software collaboratively in the open

- Open source is about software license
  - Defines how the developers develop the software
  - Defines how the users consume the software and contribute back
  - Defines how the community interacts with one another
What characterizes an open source?

- The **community** of software users and developers interested to develop some idea in the open create the open source project.
  - The makeup of each open source project is identified by it’s philosophy, it’s culture and it’s character defines an open source project.
  - **Community Philosophy**: The software license used for the open source project captures defines its philosophy.
  - **Community Culture**: Where the open source project is hosted can define its community culture. For example Apache promotes a different culture than Eclipse.
  - **Community Character**:
    - Each project can run differently even within a given open source embodiment. The community decides on how to create, build and maintain the project.
    - The technology offered by a project can also form the character of the project. For a project that develops applications will be different than one that develops an infrastructure.
Open Source License

- License defines the community philosophy
  - open source refers to software that is published under licenses that defines how source code can be made available to everyone to inspect, change, download, and explore as they wish.

- There are upward of 71 open source licenses which fall into 3 families:
  - Give me credit
  - Give me fixes
  - Give me everything
Open Source License Type – Give Me Credit

- Examples: AL, BSD, MIT
- Typical Characteristic:
  - Derivatives can sub-license
  - May have some conditions
    - No warranty
  - Credit to original authors required
  - Limited control by any one entity
  - Allows for commercial product development
  - Allows for competing services
Open Source License Type – Give Me Fixes

- Examples: Mozilla (MPL), Eclipse (EPL/CPL), LGPL
- Typical Characteristic:
  - Single entity control, still ‘business friendly’. If you modify the code, you need to make the modification available
    - File or derivative based conditions
    - Original author may have special rights
    - Differentiate between source and binary
    - Larger works can be under a different license
    - Encourages incorporation of code into larger works
    - Ensures direct development benefits all
Open Source License Type – Give Me Everything

- Examples: GPL (GNU General Public License)
- Typical Characteristic:
  - Also referred to COPY LEFT. If you use it, everything must be under this license
  - Derivative works remain under the license
  - Linked works may also remain under the license
  - Ensures all ‘down stream’ have the same rights
  - All direct development is contributed back
  - Contributors assured code remains open source
  - Encourages a full free software economy
  - Copyright holder retains much control
  - Limits commercial adoption: Forbids distribution for profit

- Software with this type of license cannot be included in Apache projects
Open Source License Type – Give Me Everything

- GNU AGPL v3.0
  - “AGPL fixes this “loophole” in GPL by saying that if you use the software over a network, you are bound by the copyleft. Other than that, the license is virtually the same as GPL v3.”

- What does it mean?
  - it is never required that applications are published/released.
  - The copyleft applies to internal AGPL Software, meaning that changes to these software needs to be published even though they have never been published/released and are only used internally.
Open Source: Most common Licenses

- Apache License, 2.0
- BSD licenses
- GNU General Public License (GPL)
- GNU Library or "Lesser" General Public License (LGPL)
- MIT license
- Mozilla Public License 1.1 (MPL)
- Common Development and Distribution License
- Eclipse Public License
- Artistic Licenses
License Compatibility

- **License compatibility** refers to the problem with software licenses which can contain contradictory requirements, rendering it impossible to combine code from such packages in order to create new software packages.

- Let’s consider the following scenario:
  - Code distributed with license A says: "modified versions must mention the developers in any advertising materials"
  - Code distributed with license B says: "modified versions cannot contain additional attribution requirements"

- These two licenses are considered **license-incompatible**. If someone combine a software package which uses license A, with a software package which uses license B, it would be impossible to distribute the combination because the two requirements cannot be simultaneously fulfilled.

License Compatibility

Apache License and third-party licenses

- **Authorized Licenses**
  - Apache License 2.0
  - Apache Software License 1.1
  - BSD (without advertising clause)
  - MIT/X11
  - ICU
  - University of Illinois/NCSA
  - W3C Software License
  - X.Net
  - zlib/libpng
  - FSF autoconf license
  - DejaVu Fonts
  - Academic Free License 3.0
  - Service+Component+Architecture +Specifications
  - OOXML XSD ECMA License
  - Microsoft Public License (MsPL)
  - Creative Commons Attribution (CC-A)
  - Creative Commons Copyright-Only Dedication
  - Python Software Foundation License
  - Adobe Postcript(R) AFM files
  - Boost Software License Version 1.0
  - Eclipse Distribution License 1.0
  - License for CERN packages in COLT

Apache License and third-party licenses

- **Excluded Licenses**
  - Binary Code License - BCL*
  - Special exceptions to the GNU GPL (e.g. GNU Classpath)*
  - GNU GPL 1, 2, 3
  - GNU LGPL 2, 2.1, 3
  - Affero GPL 3
  - NPL 1.0/NPL 1.1
  - QPL
  - Sleepycat License
  - Microsoft Limited Public License
  - Code Project Open License (CPOL)

Open Source Project Hosts

- Project umbrellas host the open source projects and can influence the overall culture of a community.

- Hosts provide infrastructure for open source project referred to as PRIM:
  - P (portal), R (repository), I (Issue tracking), M (mailing list)
  - Can provide legal governance

- There are three main types of public open source hosts:
  - Pure infrastructure
  - Vendor collaboration
  - Community focused
Project Host Type 1: Pure Infrastructure

- Examples: SourceForge, CollabNet, Codehaus, googlecode, github

- Provide the infrastructure
  - Sets overall rules (e.g. type of license permitted)
  - Each project governs itself

- Often many small projects
  - One or two developers, although may have lots of users
  - Fairly small codebase

- Provide an incubator role
  - Projects start under incubation and move to a more formal community when they grow
Project Host Type 2: Vendor Collaboration

- Examples: Eclipse, ObjectWeb, Mozilla

- Allows companies to collaborate
  - Specifically acknowledge the role companies have
  - Consider corporate needs

- Closer to commercial software development roadmap
  - More planning and oversight
    - (e.g.) have a architectural steering committee
    - (e.g.) have an official project management committee
  - Can be more conservative

- Eclipse recently created “Eclipse Project Incubator” as a place for innovation and investigation of new and alternative ideas.
Project Host Type 3: Community Focused

- **Example:** Apache Software Foundation
- **Non-profit corporation**
  - No staff, all volunteer
  - Elected membership

- **Primary goal is to foster open source communities**
  - Provide technical infrastructure
  - Provide legal oversight
  - Projects start under incubation and once they demonstrate they can run as a healthy Apache type project they graduate into an Apache top level project.

- **Technocratic Meritocracy**
  - People earn status by what they do

- **Project communities are very independent**
  - Project Management Committee (PMC) is a legal construct
    - Binding decisions e.g. to release software
  - Community decides direction and priorities
Examples of Open Source projects

- **Apache**
  - Web Server
  - Community initiated (apache.org)

- **Linux**
  - Operating System kernel
  - Individual initiated (Linus Torvalds)

- **Eclipse**
  - Universal Integration platform
  - Extensible application framework supporting solution based plug-ins
  - IT Vendor initiated (IBM and others)

- **Mozilla**
  - Browser and client technology
  - Hybrid (Netscape + community)
OPEN SOURCE COMMUNITY
HEART OF AN OPEN SOURCE PROJECT
Traditional Commercial Software

- Separate User and Developer Community
- Feedback provided through Beta and early adaptor programs
- Agile development model can help to provide more frequent feedback
Mixed Commercial – Share Code

- Code Is developed in the open and can be used readily
- Users provide feedback, but can’t modify the main repository
- Software is typically not free or a more advanced form of it is licensed

 USERS

“use the software”

 DEVELOPER S

“produce the software”
Open Source – Collaborative Environment

- Code is developed in the open and can be used readily.
- Users can become involved in the development of the software.
Start of an Open Source Project

- Open Source projects typically start with
  - A smaller developer community who have an idea to develop
  - One or more users who can use the software

"produce the software"
Who are the developers?

- A group of people interested to develop some idea in the open collaboratively.
  - Because ..
    - They want to create software that solves their business problems
    - Want to test new ideas in the market
    - Want To participate in new technology development
    - Want To test validity of standards through collaboration with users
    - Maybe they get paid by an employer, it’s their job.
    - Maybe it is just interesting to get involved in new technology!
      - Participation in a successful project can open the door to fame and recognition
  - .......
    - There is no magic answer. This makes it good challenge to analyze how to attract developers for each project

- IMPORTANT: Anyone interested can join in to help with the development
  - No invitation is required! Just get involved in what is interesting to you and share your thoughts with the community.
  - People from different backgrounds participate in open source. The key is to not be shy about not having a high command of a given language. Source Code and technology ideas become the common language.
Open Source Project

- Open Source Brings Users and Developers together to
  - Invent, develop, share experience, improve
  - Overtime, users join the developers community to influence what they use (care about).
A successful Open Source Community is an integrated user and developer community who together they

- Invent, develop, share experience, improve
Open source project grows as users get more involved and new developers join the project.

Growth and adoption makes the project more stable and brings in innovations.
Why is Community Important?

![Graph showing the relationship between time (on the x-axis) and factor of success (on the y-axis), with two lines representing code with and without community. The line for code with community is inclined upwards, indicating increased stability over time, while the line for code without community is inclined downwards, indicating decreased stability. The graph includes the statement "Community = Stability."](http://www.cubiclemuses.com)
How to Grow a Community?

- Prerequisites
  - Attractive Code Qualities
  - Communication Infrastructure
  - People who believe in the project and help it grow
Attractive Code Qualities for Starting OS projects

- Directly used by and useful to developers
- Builds
- Follows common standards where applicable
- Modular and flexible
- Consistent
- Enough documentation to help new developers to get started
- Enough test examples to help new developers to get started and be confident to test their changes
- Can be improved
  - Does not have to be functionality complete (Tough concept to grasp)
    - Incremental, smaller checkins accompanied with discussions with the community get the community more involved and enables others to participate in building the software.
Communication Infrastructure

- Source code repository
- Issue Tracking
  - Defines how problems can be reported.
  - Provide a way to organize handling of many different ideas, feedbacks, etc.
- Website
  - First impression of the project is from the website. It is important to have a good website which clearly states objectives and introduces the visitor the to project
    - Most developers don’t find this a fun thing to do! Find ways to make it happen.
- Documentation
  - User documentation: Focuses on how to use the project
  - Developer documentation: Focus on how to get involved
    - Caution: ‘code talks’ does not work 100%. It is worth to spend time to share information
- Mailing list, newsgroup, or forum
  - Communication should be open
    - Engages everyone and solicits new ideas and participation
  - Communication should be archived
    - Archived information can be used to search for problems that were discussed before
    - Provides a reference for decisions that were made
How to attract a community?

- Good product and frequent releases
- Good documentation
- Examples of how to use the software
- Modular and flexible
  - Lowers the barrier to entry for developers. Let’s them focus on areas that they are interested in
  - Facilitates adoption by allowing users to pick and choose what they need. Lower footprint.
- Have an open, inviting environment
- Mentor new people to learn the project and feel comfortable to contribute
- But, that’s not enough!
  - People need to know about your project to download and use it or come to the site to participate.
    - Talk about the project and how it solves the given business problem through Conferences, Forums, Articles, …
Summary

- Open Source brings Users and Developers from all around the world together to invent, develop, share ideas, ...

- Community is the heart of Open Source
  - Community = stability

- Starting and growing of an open source project requires
  - Code
  - Infrastructure
  - And, dedication and effort to build the community
THE APACHE SOFTWARE FOUNDATION
THE STATE OF THE FEATHER
The ASF

- What is the ASF?
- What does the ASF do?
- Name some ASF projects you know?
  - Have you used/contributed to the project?
The ASF

- ASF == The Apache Software Foundation
- Before the ASF there was “The Apache Group” (~2006)
- But we needed a more formal and legal entity
- The ASF was incorporated in 1999
- Governed solely by members - 100% member based entity
The ASF

- Non-profit corporation founded in 1999
- 501(c)3 public charity
- Volunteer organization (The ASF doesn’t pay anyone to work on Apache projects)
- Virtual world-wide organization - Membership of Individuals
- Exists to provide the organizational, legal, and financial support for various OSS projects - let the developers develop
April 13, 1999 - The first official meeting of the Apache Software Foundation (not pictured).
Why call ourselves Apache?
No, it isn’t an attack helicopter
A Patchy Server? Nope, that’s just a pun.

A reference to our development philosophy:

“Characteristic of both Eastern and Western Apache, with the exception of the Kiowa Apache, was the lack of a centralized tribal organization. The band, an autonomous collection of small local groups within a given locality, was the primary political unit as well as the primary warring and raiding unit. The strongest headman of the local groups was recognized as an informal chief, and several bands might be united under one leader. Chieftainship was thus not generally hereditary.” Encyclopaedia Britannica
At the start

- There were only 21 members
- And 2 “projects”: httpd and Concom
- All servers and services were donated
Today...

- We have 468 members...(!)
- and 59 emeritus members
- >100 TLPs
- >30 Incubator podlings
- 36 Labs
- 16 in the Attic
- Tons of committers (literally)
  - (Over 3000 people)
Apache Mission

- Provide open source software to the public free of charge

- Provides a foundation for open, collaborative software development projects
  - supplies hardware
  - Supplies communication, and business infrastructure

- Create an independent legal entity to which companies and individuals can donate resources and be assured that those resources will be used for the public benefit
The ASF’s Mission

- Provide a means for individual volunteers to be sheltered from legal suits directed at the Foundation’s projects

- Protect the ‘Apache’ brand, as applied to its software products, from being abused by other organizations

- Provide legal and technical infrastructure for open source software development and to perform appropriate oversight of such software
Vision Statement

The Apache Software Foundation provides support for the Apache community of open-source software projects. The Apache projects are characterized by a collaborative, consensus based development process, an open and pragmatic software license, and a desire to create high quality software that leads the way in its field. We consider ourselves not simply a group of projects sharing a server, but rather a community of developers and users.
Vision Statement, Take 2

- Community created code
- Our code should be exceptional
Structure of the ASF - legal

- Member-based corporation - individuals only
- Members nominate and elect new members
- Members elect a board - 9 seats
- Semi-annual meetings via IRC
- Each PMC has a Chair - eyes and ears of the board (oversight only)
ASF “Org Chart”

Technical Oversight

Development

- PMCs (TLPs) ~100
- PMC Members ~2000
- Committers ~4500
- Contributors
- Patchers/Buggers
- Users

Organizational Oversight

Administrative

- Members ~430
- Officers ~100
- Board 9

Elects

Appoints

Reports

Creates & Updates
Staying true

- Policy still firmly in the hands of the ASF
- Use outsourced help where needed
  - Help volunteers, not replace them (if they want to do the work)
  - Only for administrative efforts
    - 3 FT sys-admins, 1 PT sys-admin
    - 1 PT Exec Assistant
- Infrastructure itself is a service provided by the ASF
Staying true

- Board/Infra/etc exists so projects and people don’t need to worry about it
- The ASF exists for the projects, the community and the membership.
- Our “best practices” have proven themselves time and time again to work.
- The board runs the foundation
  The PMCs run the projects
  (and rarely do the 2 interact)
Overview

- In general, PMCs are active and healthy!
  - Lots of releases
  - Lots of development
  - New committers and PMC members
- New podlings being added (eg: CloudStack)
- New projects being graduated (eg: OpenOffice)
- ASF continues to be recognized as FOSS leader
Protection

- Ensuring that IP is checked (oversight and governance)
- Tracking and handling any legal issues
- Protecting the ASF brand and trademarks
14 Years is a LONG time

- Over a decade of declarative discussion and development
- We must be doing something right
  - By maintaining our focus on the right direction - the projects and community
  - Not all things to all people, and we don’t want to be
  - Growth, improvement without speciation
- The “Apache Way” continues to be a sought-after topic at cons.
Apache License

- Apache License Philosophy
  - To allow the maximum use of our software for any purpose and by all people.

- Give me Credit type

- Requires Attribution via a NOTICE file
  - Credits to original author is required, derivatives can sub-license

- No Trademark Grant

- Includes patent grant

- A liberal open source software license
  - Business friendly
  - Easily reused by other projects & organizations
Apache Infrastructure

- Provide legal and technical infrastructure for open source software development and to perform appropriate oversight of such software

- Provide a foundation for open, collaborative software development projects by supplying hardware, communication, and business infrastructure
  - Portal (confluence and moin moin wikis, content management for website)
  - Repository for source, SVN and GIT for source control
  - Issue Tracking, JIRA and Bugzilla for bug tracking
  - Mailing Lists, all the communication
  - Also provides integration build infrastructure and many more
Apache Roles

- Apache Contributors/Committers are viewed as individual contributors, independent of their potential company affiliation.
- Individuals are recognized for their contributions and can assume more responsibility.
- Products are owned by project community.
- Emphasis on community stewardship.

Aaron Farr, http://www.slideshare.net
Apache Roles Definition (1 of 2)

- **User**
  - Uses the software and provides feedback in the form of bug or feature requests.

- **Contributor/Developer**
  - Contributes to the development of the project.
  - Takes the extra steps to participate in the project: Active on the mailing list, participate in discussions, provide patches, documentation, website updates, new features, etc.

- **Committer**
  - Committer is a developer that has been given write access to the code repository and has a signed Contributor License Agreement (CLA) on file. They have an @apache.org mail address
Apache Roles Definition (2 of 2)

- **PMC (Project Management Committee) Member**
  - Each project has a project management committee who makes decisions for the project, such as software releases and community related decisions.
  - A committer becomes a PMC member as a result of their involvement in the community matters related to the project.

- **PMC Chair**
  - The Chair of a Project Management Committee (PMC) is appointed by PMC Members.
  - The Chair is the interface between the Apache Board and the Project. He/she reports status of the project on a regular basis.

- **Apache Software Foundation (ASF) Member**
  - An ASF member is a person that was nominated by current members and elected due to merit for the evolution and progress of the foundation. Members care for the ASF itself.
The Apache Philosophy – “The Apache Way”

- **Meritocracy** – It’s about what you do, **those who contribute decide**

- **Peer to Peer**
  - We work with people, not companies.
  - Committers and members should hold one another with respect.
  - All votes hold the same weight.
  - Community over code.

- **Consensus decision making**
  - Most decisions made on mailing list without voting, sometimes by way of lazy consensus
  - Typically use Wikis for proposals and design sharing
  - Voting rules: Yes (+1)  Abstain (0)  No (veto or -1)
    - A -1 veto requires proposal of an alternate solution

- **Open, online communication**
  - Email lists are preferred form of communication. ← The most used and preferred
  - Most communication is publicly archived.
  - Most lists are open to any subscriber

- **Responsible oversight**
  - Security is mandatory
  - Ensure license compliance
  - Release only high quality software
  - No abuse of Apache brand or community
What is Karma?

- A Buddhist/Hindu conceptions of the sum of a person's actions which dictate his future (lives).
- Now, with regards to open source, your good actions and contributions come back to you as “karma” or “permission to a given resource” (e.g. when you become a committer, you receive karma to source code repository).
How to Become a User?

- Simply download and use the open source project.
- Report bugs
- Talk to the project community through user mailing list
  - Ask questions that will help you build your solution
- Provides feedback to the project
  - **Usage feedback is important** - Examples, here is how we are using the project. It is working well in these cases and it can be improved in these cases.
    - Helps the developers to create software that addresses real use case scenarios.
    - Creates a user community who can help each other to use the software properly
How to become a contributor/developer?

- Actively participate on the project mailing list by answering user questions, contributing to design discussions, etc.
- Start providing fixes (patches) and small enhancements to code, documentation, website.
How to become a committer?

- Projects recognize contribution and extends the committer role to high contributors. This means:
  - Write access to code and an @apache.org mailing address. It requires a signed CLA in place.

- One gains committer status through contribution as well as by gaining the trust of the PMC.
  - The criteria is really defined by the character of each project, that is what PMC believes is important to the project. What is important to that project. Some general characteristics:
    - Dedication
    - Attention to code quality and project guidelines
    - How well this person collaborates with the rest of the community
    - How openly they work
How to Get Involved with Apache?

- Apache is a non-profit organization run by volunteers
- Apache sponsors can provide monetary support:
  
  http://apache.org/foundation/sponsorship.html
  
  - It is the closest and most direct method for a corporation or individual to support the ASF
  - The ASF provides the infrastructure for the projects - the mailing lists, code repositories, bug tracking systems, etc. While all of the administrative effort is currently through unpaid volunteers, financial assistance to purchase bandwidth and keep servers running is required.

- If interested to get more involved with ASF, start with the above link
HOW TO GET INVOLVED WITH OPEN SOURCE PROJECTS?
Getting Involved
Getting Involved

- Quality Assurance
- Bug Fixers
- Tech Writers
- Level one support
- Issue Tracker
- Mailing List
- Web Site

- Be proactive
- Begin with the end in mind
- Put first things first
- Seek first to understand
- Think Win-Win
- Synergize
I have been hearing a lot about this open source project called Apache Tuscany. How can I learn more about it?

- Learn about the project
  - Project overview and subproject pages
  - Ask questions on the project mailing list

- Subscribe to the mailing list and talk to the community
  - Ask questions. Any questions is good

- Download distribution
  - Typically latest release is the best to start with

- Run getting started tutorial

- Run samples, demos
  - Use samples and demos to better understand how to use the project and learn details about its features
Project Overview: http://tuscany.apache.org

- **Service Component Architecture (SCA)**: An essential characteristic of SOA is the ability to assemble new and existing services to create brand new applications that may consist of different technologies. Service Component Architecture defines a simple, service-based model for construction, assembly, and deployment of a network of services (existing and new ones) that are defined in a language-neutral way. Tuscany implements the SCA Version 1.0 specifications. Learn more about SCA.

- **Service Data Objects (SDO)**: provides a uniform interface for handling different forms of data, including XML documents, that can exist in a network of services and provide the mechanism for tracking changes. Tuscany implements the SDO Version 2.0 specifications. Learn more about SDO.

- **Data Access Service (DAS)**: provides a simple SDO interface to relational databases. Learn more about DAS.

The above mentioned technologies provide a full infrastructure for developing and running SOA based applications. They are not dependent on one another and can be used independently.

**News**

- **June 08, 2008**: Tuscany has approved graduation from the Apache Incubator to a fully endorsed ASF top level project.
- **May 21, 2008**: Apache Tuscany is now an official ASF project!
- **May 11, 2006**: PyCon Italia Due! Giorgio Zoppo presents Python e Service Component Architecture covering Python in SCA.
- **April 27, 2008**: Tuscany Java SCA 1.2.1 released.
- **April 15, 2008**: Tuscany Java SDO 1.1 released.
- **Feb 05, 2008**: Tuscany Java SCA 1.1 released.
- **Nov 12-13, 2007**: SOA World Conference & Expo 2007 West.

For service composition & management

For Data Handling
How to ask questions?

Note: Send an empty email to get registered.

- You will get an email that you need to reply to. Simply reply, no text. This activates your subscription.
- You can always unregister yourself.
- Once registered, you will see other user questions once on this mailing list.
Documentation

How to develop applications

How to help with Tuscany development and extend it
I have started using Apache Tuscany. How can I get more involved?

- Check “Get involved” page. It explains some ways to get involved.

- Report a problem:
  - Report problems via JIRA
    - Describe problem and attach a test

- Provide input
  - Features that are important and provide use cases to help the community understand why something is important
  - Provide feedback based on your experience
  - Become a reference for the project
Get Involved Page

Apache Tuscany

Getting Involved

Apache Tuscany

A successful project requires many people to play many roles, for example write code, documentation, test, provide suggestions or feedback. These are all important.

Here are some suggestions for getting started and welcome to the project.

- Tuscany consists of several subprojects. Identify what areas you are interested in. Take a look at examples for that area.
- Subscribe to the mailing list. If you are interested to get involved at the user level, subscribe to user mailing list. If you are interested to get involved in the development of Tuscany, subscribe to the developer list. Remember that you can always unsubscribe later.
- Would like to contribute new ideas? Start with the mailing list and share your thoughts.
- Answer questions posted to the mailing list.
- Identify JIRA's in the area that you are interested in and provide patches.
- Contribute to the user or developer documentation or website. Create the Wiki page on the Tuscany Wiki, create a JIRA and attach the link.
- Identify known issues that you'd like to fix and provide a patch.
- Contribute to feature development. Just let the community know what you'd like to work on. It is as easy as that.
- If in doubt about where to start, send a note to the mailing list and mention your area of interest. Any questions is welcomed. We'd like you to be involved!
- Provide feedback: What is working well? What is missing? What can help you with your deployment. This helps us create software that addresses real pain points in SOA.

communication

Please note that a single mailing list is used for all the subprojects.

The Tuscany community also gets together on IRC. The IRC server is irc.freenode.net and the channel is Tuscany. Internet Relay Chat (IRC) Help is a good source of information on understanding IRC clients.

The following are some IRC chat clients:

- mIRC http://www.mirc.com/
- Trillian http://www.leviathanstudios.com/

Tuscany SCA

- SCA Overview
- SCA Java
- SCA Native
- SCA DXP

Tuscany SDD

- SDD Overview
- SDD Java
- SDD C++
- SDD J2EE

Tuscany DAS

- DAS Overview
- DAS Java
- DAS C++
Reporting Problems: Use JIRA system
JIRA System

How do I create a JIRA?

- If first time user, create a login account: User id and password
  - You define your own user id and password
- If not a first time user, login to create a JIRA or to update a JIRA
Creating a JIRA

- Provide a one liner summary that describes the issue
- Select the area that was affected
- Select the version of the code that you are using
- Provide necessary environment information
- Provide detailed description and any stack traces on the description
- Provide a test case to reproduce if can be done
Subscribing and using the mailing lists

- Different mailing list types
  - User – Used by users to ask questions or help each other
  - Development – Used by developers to discuss designs, releases, roadmaps. JIRA updates get routed to this mailing list
  - Committs – where code commits are sent to. A good way to see what is progressing in the project

- You can subscribe, unsubscribe or watch archive of these mailing lists. You cannot respond to emails from archive.

- Subscribing to Tuscany user list
  - `<list>-subscribe@<project>.apache.org`
  - `<e.g.>` [user-subscribe@tuscany.apache.org](mailto:user-subscribe@tuscany.apache.org)

- Unsubscribing from Tuscany user list
  - `<list>-unsubscribe@<project>.apache.org`
  - `<e.g.>` [user-unsubscribe@project.apache.org](mailto:user-unsubscribe@project.apache.org)

- Browsing Tuscany Mailing Archives
  - [http://www.mail-archive.com/](http://www.mail-archive.com/)
  - [http://apache.markmail.org](http://apache.markmail.org)
  - [http://mail-archives.apache.org/mod_mbox/](http://mail-archives.apache.org/mod_mbox/)
Asking questions on the project mailing list

- Feel VERY COMFORTABLE to ask ANY kind of questions to help you understand the project. There is no bad or good questions.
- Remember that the project is a community effort and most people working on the project are volunteers.
- The way you ask questions can play in how you get answers
  - You get quicker answer if you share information about how your investigation went and where you are blocked
    - Search archive mailing list and FAQ to see if you can find your answer.
    - Search mailing list archive for similar topics
  - Clear questions get clear answers
  - Here is a nice guide for how to ask questions: http://www.catb.org/~esr/faqs/smart-questions.html
Asking questions on the project mailing list

- Use meaningful, specific subject headers
  - Not helpful: HELP!
  - Better: Webservice Error when running App

- Be precise and informative about your problem
  - Describe the symptoms of your problem or bug carefully and clearly.
  - Do the best you can to anticipate the questions someone could ask back, and answer them in advance in your request for help.

- Volume is not precision

How to Ask questions the Smart Way : http://www.catb.org/~esr/faqs/smart-questions.html
I think Apache Tuscany is a great project to get involved with. I like the idea and the technology. How can I get involved?

How to Contribute?

- Identify what areas you’d like to get involved in
  - Find JIRAs (reported problems) to fix in that area
  - Share with the community that you’d like to help and which areas you are interested in. They’ll help you find pieces
  - Check out roadmap (if there is one)

- Download Code
  - Typically the ‘latest’ code branch is also referred to as trunk

- Read relevant developer documentation guides such as architecture doc, etc.

- Check out development guideline if any. It usually has good hints for how to build and get started

- Learn how the code works via samples,
Finding JIRAs to fix, good starting point
Anyone can read the code
Anyone can provide a patch
Only committers can write to the repository
There are different subprojects under Java (SCA, SDO, DAS, ..)

Download the code
• DAS – Data Access Object
• SCA – Service Component Architecture
• SDO – Service Data Object

• Developers Guide with general info on how to check out code and build is on: http://tuscany.apache.org/sca-java-development-guide.html
Exploring open source releases

How do I learn more about a release?

- Download the release
  - There is a binary and source distribution for each supported platform
- Check important distribution artifacts
  - RELEASE_NOTES
  - CHANGES
- Learn how the code works via samples, tests
Tuscany Binary and Source distribution

Java SCA source distribution
- Demos
- Distributions
- Integration Tests
- Implementation Modules
- Samples
- Tools
- Tutorial

Java SCA binary distribution
- Demos
- Binary Dependencies
- Tuscany binaries
- Samples
- Tutorial
Running a sample from a distribution

- cd tuscany-sca-1.2.1-incubating\samples\calculator
- ant run
  - Buildfile: build.xml
  - run:
    - [java] 3 + 2=5.0
    - [java] 3 - 2=1.0
    - [java] 3 * 2=6.0
    - [java] 3 / 2=1.5
Contributing your fixes – Patch process

I have found this bug, and have a fix for it. How can I give it back to the Tuscany Project?

- Communicate to the community that there is an issue by creating a new JIRA issue
  - Bugs
  - New Features
  - etc
- Discuss any design issues, doubts, etc on the dev-list.
- Provide a patch
Contributing your fixes – Patch process

- Configure your svn properties
  - https://svn.apache.org/repos/asf/tuscany/java/etc/svn-props

- Checkout the code
  - https://svn.apache.org/repos/asf/tuscany/java/sca/
  - Use svn client or TortoiseSVN

- Modify a file(s)
  - Edit modules/pom.xml
  - New files need to be added
    - `svn add`

- Check modified files
  - `svn status`

- Generate patch
  - `Svn diff > my_updates.patch`

- Attach the patch to the related JIRA. If a JIRA does not exist, create one.
  - Make sure you grant Apache License
  - Make sure you mark the flag that says a patch is included

- Someone in the community will pick up the patch, review it and apply it or ask for more information.
Summary

- Getting involved in Apache Tuscany is easy
  - Check Get involved
  - Subscribe to the right mailing list and ask questions, provide your input and contribute
  - Plenty of choices:
    - Report problems via JIRA
    - Provide patches
    - Contribute samples, scenarios
    - Provide feedback on the project is working for you
    - Contribute code
    - Help with the website content, documentation
    - .......

- Any contribution is valued
List of Apache Projects

- Apache TLPs
  - [http://projects.apache.org/indexes/quick.html](http://projects.apache.org/indexes/quick.html)

- Apache Podlings

- Apache Labs

Home Work – Find a project of your interest

- Familiarize with the project resources
  - Website
  - Documentation
- Subscribe to development and user list
HOW TO START AN APACHE PROJECT?
Incubator Projects

- Apache projects start as incubator projects that are overseen by incubator PMC. They are called podlings.

- Once a project demonstrates it can behave as a ‘good’ Apache citizen, it graduates to a top level project
  - It is a worthy and healthy project: diverse and will sustain itself
  - It truly fits within the ASF framework
  - It “gets” the Apache way
    - Useful code
    - Working Communication infrastructure
    - Active developers
    - Dedicated users
About Apache Incubator

- Apache incubator is the entry path into The Apache Software Foundation (ASF) for projects wishing to be part of the foundation’s effort.
- Details about Apache Incubator at
- The Apache Incubator have two main goals:
  - Ensure all donations are in accordance with the ASF [legal standards](http://incubator.apache.org/legal)
  - Develop new communities that adhere to our [guiding principles](http://incubator.apache.org/community)
- Incubator PMC provides oversight for new projects on the following areas:
  - Responsible for the acceptance and oversight of new products submitted or proposed to become part of the Foundation
  - Ensures that the projects develop products according to the Foundation's philosophy and guidelines for collaborative development
  - Legal – ensures donation are in accordance with legal standards
  - Community – helps develops communities that adhere to guiding principles
  - Regularly evaluates products to determine if the product should be abandoned, continue to receive guidance and support, or proposed to the board for promotion to full project status as part of an existing or new Foundation PMC;
How to Start a New Incubator Project?

- **Detailed Guidelines at this link:**
  - [http://incubator.apache.org/guides/proposal.html#preamble](http://incubator.apache.org/guides/proposal.html#preamble)

- **Starts with a proposal… Formulate your proposal**
  - Find a champion. The Champion is a person associated with ASF already and helps move the proposal forward.
  - Identify mentors from ASF or ask for volunteers
  - Find a sponsoring organization for the project. This could be:
    - The Apache Board
    - The incubator
    - Another Apache Project
  - Post the proposal in plain text in a email whose subject is prefixed with [PROPOSAL] at general@incubator.apache.org
    - Example: [Proposal] Photo Share Project for incubation
    - If there is interest, the proposal gets discussed. Expect to revise your proposal with input from the general discussion. Be Responsive and inviting.
  - Once there is consensus on the proposal, post in a email proposal whose subject is prefix with [Vote].
    - Example: [Vote] Photo Share Project for incubation
  - When Incubator PMC approves the project, mentors help set up the infrastructure (PRIM) for the project.
Tips for Building a successful project

- Make sure there is enough help/resource to jump start and drive the project
- Set clear expectations (pathway)
- Community should understand the purpose and focus
- Don’t worry about duplication
- Be willing to break up
- Open communication - extremely important
- Be inviting and help others to come on board
- Promote the project (blogs, conferences, articles)
- Release often and regularly
- Good documentation and website
- Lower the barrier to entry
- Integrate with other Apache projects if it makes sense
- Watch how other successful projects behave and work
- Use your mentors and the incubator PMC effectively
- Focus on creating a diverse and stable community
- And, be Patient.
DAY 2
OPEN SOURCE TOOLS
OVERVIEW
Topics

- Install development environment tools
  - Subversion
  - Git
  - Maven

- Source Control Repositories
  - Subversion
  - Git

- Build and Dependency Management
  - Maven
Installing development environment tools

- Subversion
  - http://subversion.apache.org/packages.html

- Git

- Maven
  - http://maven.apache.org/download.cgi

- Eclipse

Also available at: http://10.0.0.3/icfoss

Wi-Fi: ICFOSS-ROBOTICS ICFOSS606
SUBVERSION (SVN) 101
Subversion

- Centralized source control repository

- Regular project structure
  - Trunk – current development
  - Branches – parallel development
  - Tags – snapshots (e.g. release tags)
Subversion useful commands

- Checkout source code
  - `svn co http://www.company.com/repo/project/trunk`
  - Ex. `svn co http://svn.apache.org/repos/asf/wink/trunk/ wink-svn`

- Update current checkout
  - `svn update`

- Check status of current checkout
  - `svn status`

- Check differences
  - `svn diff`
  - `svn diff >> changes.patch` (create a patch with the current diff)

- Rollback changes
  - `svn revert file`
  - `svn revert *`

- Add new file
  - `svn add file/directory`

- Commit files
  - `svn ci -m "Describe the contents of your commit"`
Subversion hands-on

- Find an ASF project using subversion
- Play with the basic subversion commands
GIT 101
Git

- Distributed source control repository

- Regular project structure
  - Trunk – current development
  - Branches – parallel development
  - Tags – snapshots (e.g. release tags)
Git useful commands

- **Checkout source code**
    - E.g: `git clone git://git.apache.org/wink.git`
- **Update current checkout**
  - `git pull --rebase`
- **Check status of current checkout**
  - `git status`
- **Check differences**
  - `git diff`
  - `git format-patch >> changes.patch` (create a patch with the current diff)
- **Rollback changes**
  - `git reset --soft / --hard`
  - `git checkout file`
- **Add new file**
  - `git add file/directory`
- **Commit files**
  - `git add file // git commit -a -m"Describe the contents of your commit"`
  - `git commit -a -m"Describe the contents of your commit"`
  - `git push`
  - `git svn dcommit for git-svn bridge`
Git branch workflow

- Clone the repository for the first time
  `git clone git://git.apache.org/wink.git`

- Update the repository
  `git pull --rebase`
  `git svn rebase for git-svn bridge`

- Workflow
  `git checkout -b JIRA-101`
  `git commit -a -m "JIRA-101 – My updates"`
  `git commit -a -amend`
  `git rebase master`
  `git push`
  `git svn dcommit`
  `git checkout master`

- Updating a branch with changes from master
  `git rebase master`
Git hands-on

- Find an ASF project using GIT
- Play with the basic GIT commands
What is Maven

- A Java project management and integration build tool
- Based on the concept of XML Project Object Model (POM)
- Favors convention over configuration
  - Project Structure
  - Repository Layout
  - Etc
- Modular build platform
  - Extensible via plugins (a.k.a MOJO’s)
Maven Build Lifecycle (Phases)

Phases
- Validate
- Compile
- Test
- Package
- Install
- Site
- Deploy

Plugins + Goals
- Maven-java-plugin:compile
- Maven-test-plugin:test
- Maven-jar-plugin:jar

```
<project>
  ...
  <packaging>jar</packaging>
  ...
</project>
```
Maven Repository

- Repositories
- No more libs in source code
- Public repositories
  - Central, etc
- Internal repository (Nexus, Artifactory, etc)
  - Proxy repository
- Local repositories
- `<project>`
  
  ...  
  `<repositories>`
  `<repository>
   `<id>central</id>
   `<url>http://repo.internal.company.com:8081/releases</url>
  </repository>
  `<repositories>
  ...
  `</project>`
Maven Dependency Management

<project>
    ...
    <dependencies>
        <dependency>
            <groupId>org.apache.tuscany.sca</groupId>
            <artifactId>tuscany-binding-rest-runtime</artifactId>
            <version>2.0-M5</version>
            <packaging>jar</packaging>
        </dependency>
    </dependencies>
    ...
</project>

Maven Project Structure

- **Project (jar)**
  - src
    - main ..................... Project source code
    - Java ..................... Java Artifacts
    - Resources ............. Resources (e.g. xsd, composites, etc)
  - test .......................... Test codes (not added to final jar)
    - java ........................ Test Java Artifacts
    - resources ............. Test Resources

- **Project (webapp)**
  - src
    - main ..................... Project source code
    - java ..................... Java Artifacts
    - resources ............. Resources (e.g. xsd, composites, etc)
    - webapp ................ Web related resources (e.g. html, jsp, css, etc)
      - META-INF
      - WEB-INF
        - web.xml
      - Web resources
Maven Project Configuration

- Configuration is entered in XML format in a Project Object Model or POM
- Projects are structured in a hierarchy and project configuration is inherited by sub-projects
  - If no parent is specified, parent is called “Super” or “Parent POM”
Maven Project Configuration

<project>
  <parent>
    <groupId>com.company.application</groupId>
    <artifactId>parentProject</artifactId>
    <version>1.0.0-SNAPSHOT</version>
    <relativePath>../pom.xml</relativePath>
  </parent>
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.parent.application</groupId>
  <artifactId>artifactName</artifactId>
  <packaging>pom</packaging>
  <name>artifactName</name>

  <modules>
    <module>…</module>
    <module>…</module>
  </modules>
</project>

Modules must map to directory structure
Maven useful commands

- **Build**
  - mvn clean install
- **Build in offline mode**
  - mvn clean install -o
- **Build and force updating/downloading dependencies**
  - mvn -U clean install
- **Build without executing unit tests**
  - mvn clean install -Dmaven.test.skip=true
- **Build and report errors only at the end (fail at end)**
  - mvn -fae clean install
- **Build and don’t report errors (fail never)**
  - mvn -fn clean install
- **Execute only one test case**
  - mvn -Dtest=ComponentServiceReferenceTestCase

Clean is optional, but omitting it might cause unexpected issues.
Maven useful commands

- Put maven in debug mode
  - mvnDebug
- Identify 3rd party dependencies with maven dependency plugin
  - mvn dependency:analyze
  - mvn dependency:copy-dependencies
  - mvn dependency:tree
Maven with Eclipse

Maven Eclipse plugins

- The goal of the Eclipse m2eclipse project is to provide Apache Maven support in the Eclipse IDE, making it easier to edit Maven's pom.xml, run a build from the IDE and much more. For Java developers, the very tight integration with JDT greatly simplifies the consumption of Java artifacts.

- [http://download.eclipse.org/technology/m2e/releases](http://download.eclipse.org/technology/m2e/releases)
SUGGESTED HOMEWORK
Github Project

- Create a github java project
  - Create new repository
  - Create a maven java project
  - Add some business logic
  - Add tests to verify your business logic
  - Build
    - With tests
    - Without tests
Github collaboration

- Provide a patch for your colleague project
  - Do a simple change on your colleague project
  - Submit a pull request
  - Add a bit more change (simulating a update based on feedback)
  - Submit a pull request

- Owner should accept pull request and merge changes
  - If you can’t find an owner, I will create a simple project and you can use that
    - My github account : lresende
HOW MENTORING PROGRAMS CAN HELP YOU GET STARTED WITH OPEN SOURCE
Why Mentoring ?

Open Source Project

You
Why Mentoring

Open Source Project

Mentor

Mentorship:

- Experience
- Guidance
- Support

Baby

Cookie Monster

Mentorship:

- Experience
- Guidance
- Support

Baby

Cookie Monster

Mentorship:

- Experience
- Guidance
- Support

Baby

Cookie Monster

Mentorship:

- Experience
- Guidance
- Support

Baby

Cookie Monster

Mentorship:

- Experience
- Guidance
- Support

Baby

Cookie Monster

Mentorship:

- Experience
- Guidance
- Support

Baby

Cookie Monster

Mentorship:

- Experience
- Guidance
- Support

Baby

Cookie Monster

Mentorship:

- Experience
- Guidance
- Support

Baby

Cookie Monster

Mentorship:

- Experience
- Guidance
- Support

Baby

Cookie Monster
Why Mentoring?

- What to expect from the mentor
  - Will help you get familiar with the open source community, and help you interact with the existent community
  - Your mentor might also help with technical questions/issues, but this might not be a complete requirement
What mentoring programs are available?
Google Summer of Code
Goals

- Inspire students to engage in open source development and also give them more exposure to real-world software development scenarios

Eligibility Requirements

- Must be 18 years old by program announcement/start
- Must be a student.
Google Summer of Code

- Restrictions
  - Code only
  - Summer only
  - Students only
Google Summer of Code

- How it works?

Project Ideas

Mentoring & Evaluations

Project Proposals

Open Source

Google

OpenOffice.org

Students working on computers
Google Summer of Code

- Google Summer of Code web page
  - [http://code.google.com/soc/](http://code.google.com/soc/)

- Google Summer of Code FAQ

- Google Summer of Code discussion group
  - [http://groups.google.com/group/google-summer-of-code-discuss](http://groups.google.com/group/google-summer-of-code-discuss)

- IRC Channel
  - IRC Freenode #gsoc channel
Google Code-In
Google Code-In

- Goals
  - Inspire young students to engage and contribute to open source communities

- Eligibility Requirements
  - Must be between 13-18 years old
  - Must be a student.
Google Code-In

- Google Code-In web page
  - http://code.google.com/opensource/gci

- Google Code-In FAQ
Apache Mentoring Programme
Apache Mentoring Programme

- Goals
  - Foster a greater community around the various Apache Projects
  - Introduce The Apache Way to people new to Apache and open source in general
Apache Mentoring Programme

- Restrictions?
  NO RESTRICTIONS ...
  - Any contribution
  - Any time
  - Anyone
Apache Mentoring Programme

- How it works?

Project Ideas → Project Proposals

Mentoring & Evaluations
Apache Mentoring Programme

- Finding a local mentor

Nearby Apache People

Brought to you by the Apache Community Development Project.

Community Development Site > Find People

People Near You

- Sander Tomme 11.4 km
- Jean-Sebastien Delfiao 32.7 km
- Ted Dunning 52 km
- Chris Hostetter 122 km
- Jeff Genender 1507 km
- Brian McCallister 1523 km
Apache Mentoring Programme

- Community Development web page
  - http://community.apache.org

- Mentoring Programme web page
  - http://community.apache.org/mentoringprogramme.html

- FAQ
  - http://community.apache.org/newbiefaq.html
ASF & ICFOSS
Mentoring Programme
Programme Information

- Mentoring Programme Page
### Schedule

- **June 20**: Local Workshop at ICFOSS headquarters in Kerala - India
- **June 24 - July 12**: Would-be student participants discuss application ideas with ASF projects
- **July 15**: Students project proposal submission opens
- **July 19**: Students project proposal submission deadline
- **July 22 - August 02**: Mentors review and rank students project proposals
- **August 05**: Accepted student proposals announced
- **August 05**: Students start coding their project proposal with Mentor guidance
- **September 09**: Mentors and students can begin submitting mid-term evaluations
- **September 13**: Mid-term evaluation deadline
- **October 21**: Suggested 'DCUT' date, where students should then scrub code, enhance documentation.
- **October 28**: Mentors and students can begin submitting final evaluations
- **November 1**: Final evaluation deadline
Finding Apache Projects

- List of Apache Projects
  - [http://projects.apache.org/indexes/quick.html](http://projects.apache.org/indexes/quick.html)

- List of Incubator Projects
  - [http://incubator.apache.org](http://incubator.apache.org)
Finding Project Ideas

- ICFOSS Project Ideas
  - [http://s.apache.org/icfoss2013ideas](http://s.apache.org/icfoss2013ideas)

- Google Summer of Code Project Ideas
  - [http://s.apache.org/gsoc2013ideas](http://s.apache.org/gsoc2013ideas)
The workflow

1. Select an area of interest (e.g. Cloud, Big Data, Services)
2. Find an Apache Project on that area
3. Find JIRA (ideas) for that Project
4. Contact Mentor that submitted the idea
5. Start creating your project proposal
6. Discuss/Review project proposal with mentor (using dev list)

Additional resources:
- List of Incubator Projects: http://incubator.apache.org
- ICFOSS Project Ideas: http://s.apache.org/icfoss2013ideas
- Google Summer of Code Project Ideas: http://s.apache.org/gsoc2013ideas
Creating your project proposal

- **Start as early as possible**
  - The earlier you start working on your proposal the higher is your chance to get accepted.

- **Focus on quality**
  - Make one good proposal, instead of few half baked ones

- **Get involved in the project community**
  - Subscribe to the development list and discuss your proposal with your mentor and others on the community

- **Be descriptive when writing the proposal**
  - Describe in detail the idea being proposed, and how is your approach to implement it.
  - Describe the technical details (this shows you have a clear idea of what needs to be done). Some high level system diagrams is a plus.

- **Write a timeline**
  - Include a detailed schedule of your project
Examples of project proposals

- Searching artifact across SCA Domains
  - [https://cwiki.apache.org/confluence/display/TUSCANYWIKI/Searching+artifacts+across+SCA+domain](https://cwiki.apache.org/confluence/display/TUSCANYWIKI/Searching+artifacts+across+SCA+domain)

- GIMPLE to GRAPHITE transformation
  - [http://students.fim.uni-passau.de/~grosser/gcc_soc/](http://students.fim.uni-passau.de/~grosser/gcc_soc/)

- Nmap Scripting Engine – Infrastructure Manager
Thank You !!!
Notice

- This presentation contains contents from “Secret Life of Open Source” slides, originally created by “Ted Husted”.

- This presentation contains contents from “The State of the Feather” slides, originally created by Jim Jagielski, which are licensed under a Creative Commons Attribution 3.0 Unported License.
ADDITIONAL INFORMATION
HOW COMPANIES USE OPEN SOURCE
How companies use open source

- Internal forking
  - Pros
    - Quicker to produce fixes for internal environments/deployments.
  - Cons
    - Requires deep knowledge of the source code
    - Can’t easily move to new releases
  - If a must, at least maintain a internal branch for the fork
How companies use open source

- Partial forking, where fixes are provided back to community
  - **Pros**
    - Quicker to produce fixes for internal environments/deployments.
    - Fixes are provided to the community.
    - Provide a path for moving to new releases.
  - **Cons**
    - Depending on how often codes are contributed back, things can get very apart.

Internal fork

Release 1.x

Trunk

Release 2.x
How companies use open source

- Working on the community
  - Pros
    - Always in sync with latest fixes
  - Cons
    - In general can be a bit slower depending on the community
    - Can introduce regressions, which can be mitigated by test coverage in the community source

Diagram:
- Internal code drop
- Release 1.x
- Release 2.x
- Trunk