1. Software Engineering Process
   1. Waterfall
   2. AIM (All-In-One Methodology )
   3. RUP (rational Unified Process)
2. Version control
   1. PVCS
   2. SVN branch
   3. fgfgf
3. Build and Deployment
   1. Ant
   2. Maven
   3. Jenkins (a.k.a Husdon) - – run maven or ant scripts, Cruise control for continuous integration
4. Tracking system
   1. Trac server
5. Testing
6. Recording:

<http://www.screentoaster.com/>

<http://softwaretestingwinners.blogspot.com/2009/03/tips-to-capture-unable-to-reproduce.html>

1. Testing data
2. Unit test - EasyMock, JMock
3. Integration Test
4. Performance Test
5. Regression Test
6. Web testing. Selenium

Firefox addons:

httpFox. Data tamper, firebug, Modify Headers, Error Console, RestClient, User Agent Switcher

1. Application Design
2. Package structure
3. Properties management
4. Multi-year support
5. Divide work into small units
6. Web Server
7. Cluster
8. Software cluster – multiple server instances in one box
9. Hardware cluster – multiple servers behind load balancer
10. Big IP – for statistics?
11. Security
12. DMZ
13. Firewall
14. Database
    1. Clustering (Oracle RAC)
    2. Backup/Create Script.
    3. Performance tuning.
    4. NoSQL database
15. Message Queue
    1. Active Message queue (Stomp, Openwire)
    2. Rabbit MQ (AMQP)
    3. Websphere MQ
16. Documentation
17. Functional Spec
18. Wireframe
19. Word doc Track Changes
20. Use case
21. High Level Design Spec
22. UML
23. Component Diagram
24. Details Design Spec
25. UML class diagram
26. Data flow
27. Page flow
28. Environment Setup document
29. Wiki
30. Build and Deployment Document
31. Build script
32. Production Operation
33. Initial setup
    1. Seed data
    2. folder structure
34. Network
    1. Email server
    2. FTP/SFTP server
    3. File system
35. Monitoring
    1. Log4j email, console, file, jdbc, jms, AsyncAppender, mongoDB

Asynchronous Logging:

Spring:

http://java.sys-con.com/node/2123093?goback=.gde\_46964\_member\_192634635

* 1. Jconsole
  2. SiteScope

1. System Cleanup
   1. Stranded jobs.
   2. Error message queue
   3. Invalid/unsysnchronized data
2. Documentation
   1. Production operation manual
   2. Trouble shooting guide
3. Security
   1. Jaas
   2. Container -managed security/ JEE security model

Declarative, user-based security model.

1. Tomcat

<http://www.coderanch.com/t/134555/Security/JAAS-vs-Container-Managed-Security>

1. Websphere
2. Weblogic
   1. Spring-security
   2. OpenId vs CAS

http://justarubyist.blogspot.com/2008/05/choices-choices-choices-cas-or-openid.html

* 1. Openid vs CAS vs Kerberos – Kerberos often mixed with LDAP
  2. Kerberos - primarily for client-server authentication. No password passed via network.

Verify identities of both client and server and thus against man-in-the-middle attack

* 1. Digest Access Authentication

No password transferred via network. Vulnerable to man-in-the-middle attack

* 1. LDAP – not an authentication protocol. Can be used for both authentication and authorization just like database does. LDAP server can be configured to use different

authentication scheme e.g. Kerberos, SASL. In JNDI LDAP sever, env.put(Context.SECURITY\_AUTHENTICATION, "simple"); instruct server to use simple level authentication.

Very much like JDBC. Connect to LDAP server and perform search/read/update

http://www.koders.com/java/fidFD0D61C53E871FB7F859FBB6057300DE5BA418FB.aspx

* 1. Cryptography

1. Hash function – use with salt(nounce) to prevent Dictionary attack
2. Symmetric key – fast. Problem with key distribution
3. Asymmetric key pairs – slow. Good for key distribution
4. MAC/HMAC to ensure message integrity.
   1. Nounce/salt. Used to prevent replay-attack
5. Batch
6. Job launch
7. Scheduled
   * 1. Unix Cron job
     2. Window Scheduler
     3. Quartz
8. Message driven
9. EJB message bean
10. Spring message container
11. Job Definition

Jobs with dependency on other jobs may be invoked using message. Thus jobs are loosely coupled in an asynchronous manner. Do not want to lump these jobs as steps of one job. Each job can be executed at different rate and failure of each job can be handled separately. e.g. job A download file form FTP server, job B processed files from job A and write to database. Database server and FTP server could fail for different reason, we do not want one job handle both failure. Define two jobs with job B invoked form message sent from job a makes system more robust.

1. Spring Conetxt can be managed to have parent context and child context where parent context has all common information on job (e.g jobLauncher, jobRegister etc). child context has information about a single job definition. This is useful for multi-cycle support, where jobs from different cycle can have the same name. Property files can contain the same property name with different values for different years.
2. ctrl + shift + i, ctrl + shift + h
3. Java Concurrency

In multi-threaded environment, class member variables are not thread safe. Either use synchronized access or use ThreadLocal to guarantee thread safe. ThreadLocal is very useful fro framework implementation. ServletContet (or SecurityContext ) is stored as ThreadLocal so they are not pass around polluting method signatures. e.g FaceletContext.getCurrentInstance(), SecurityContextHolder.getCurrentInstance.

1. sdsad
2. Web application
   1. View technology
3. Freemarker
4. Velocity
5. Jsp
   1. Web framework
6. JSF
7. Tapestry
8. Struts
9. Spring MVC (Spring webflow)
10. GWT
11. Wicket
12. **Multi-cycle support**
    1. Model -> tag interface to enhance type safety
    2. DAO/Service -> interface with java generics
    3. Handler -> cycle neutral handler uses FacadeService wired with non-typed services
    4. Spring Context -> each servlet for one cycle with servlet-scoped Spring Context

Use customized SpringBeanFacesELResolver to switch SpringContext based on url pattern

* 1. Flow variable -> use a filter to switch webflow RequestContext
  2. Page EL expression -> Layout define facelet ui prarameter referencing correct cycle (if using one context). Referencing flow var is always cycle specific.
  3. Bundles -> use spring bundles. -> Layout define ui prarameter referencing correct cycle.

e.g <ui:param name="bundle1" value="#{cycle1112\_b1}"/> “cycle1112\_b1” is spring bean name of type spring resource bundle. #{bundle1[lbl.student]} on the page will resolve to bundle.

* 1. Page tag name space -> Have to update each page manually
  2. Page include path -> el expression
  3. Scenario: parent login flow and two subflow cycle1112/account, cycle1213/account.

**Solution 1** (works): 3 completely separate SpringContext including webflow definition and thus 3 DispatcherServlet listens to: /app, /app1112, /app1213.

Use customized SpringBeanFacesELResolver to switch SpringContext based on url pattern. Also use a servlet filter to switch webflow RequestContext based on url

Pros: webflow definition is separated for each cycle .

Cons: subflow is NOT supported. Form Login to cycle112/account is considered starting a new flow.

**Solution 2** (NOT WORKING because webflow uses DispatcherServlet SpeingContext as parent context. If all flows are defined by in non-cycle servlet, then all webflows uses that servlet’s context as parent).

3 separate SpringContext using 3 servelt. Only parent servlet has webflow definition and listens to /app. Other two servlet is used to initialize Spring Context without webflow definition and do not process any request (no url mapping).

Need customized SpringBeanFacesELResolver to switch SpringContext based on url pattern. But no filter needed to switch webflow RequestContext, which is defined centrally in parent context.

Pros: subflow is supported

Cons: webflow definition for each cycle is mixed.

http://localhost:8680/formProcessor/spring/jsf2sample;jsessionid=6553DB0F250EB2E83C0341D3DE926060

**TAP 2012-2013 Rollover**

Straighten out matching criteria

Upgrade Spring to 3.0, Spring Batch Incrementor

Spring batch Admin (Tomcat Security)

Cron expression validator.

Struts -> Weblfow + jQuery etc.

Authentication -> possible performance improvement?

Hibernate:

Load vs Get: <http://gmarwaha.blogspot.com/2007/01/hibernate-difference-between-sessions.html>

P405. Load will return a proxy if the entity is configured so. Get always hit database.

Lock vs Update:

P410. Update will always issue update statement. Lock may not force update depending lock mode.

The object updates made before or after attaching session matters to “lock” but not to “update”

Merge vs SaveOrUpdate: <http://blog.tallan.com/2008/09/18/hibernate-merge-vs-saveorupdate/>

Merge(Code code) will return a new Code object with merged state, the passed-in code object is NOT touched.

Persist vs Save:

Both will “attach” the instance and call database to save. Save will issue another sql to get the updated object back while persist not

**KeyStoreExplorer:**

**MongoDB:**

<http://docs.mongodb.org/manual/use-cases/pre-aggregated-reports/>

MongoDB stores BSON documents as a sequence of fields and values, not as a hash table. As a result, writing to the field stats.mn.0 is considerably faster than writing to stats.mn.1439.