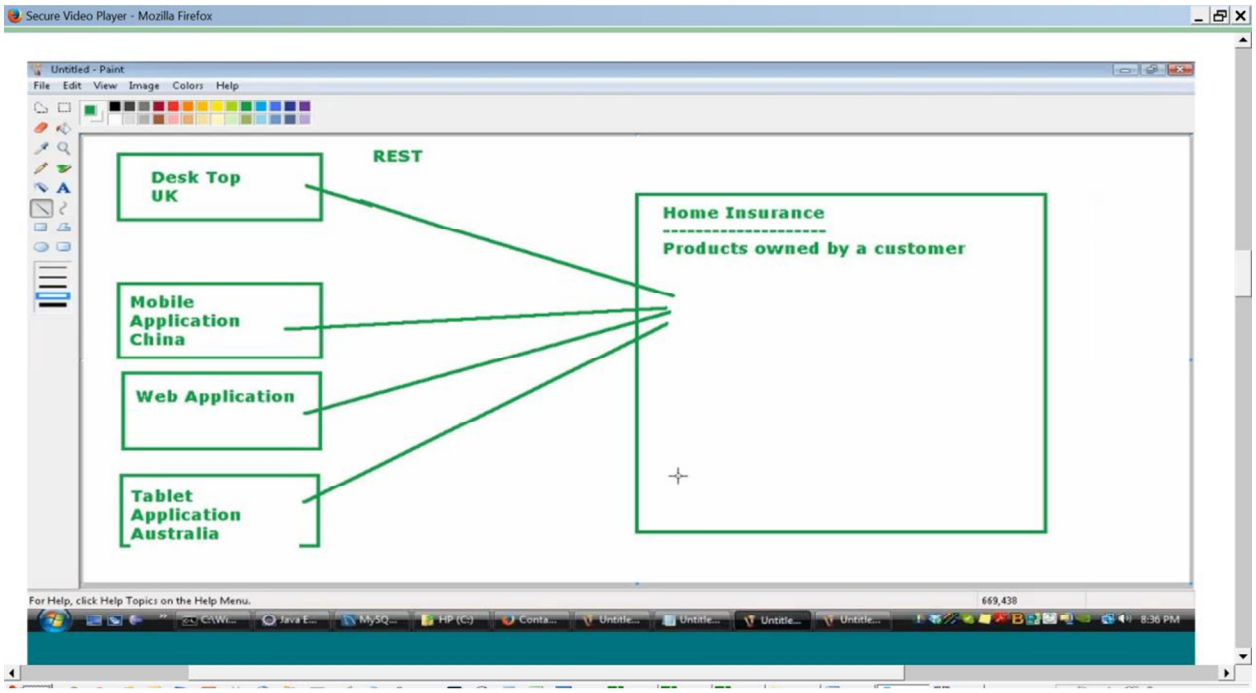
16-20140507: RESTful Web Services

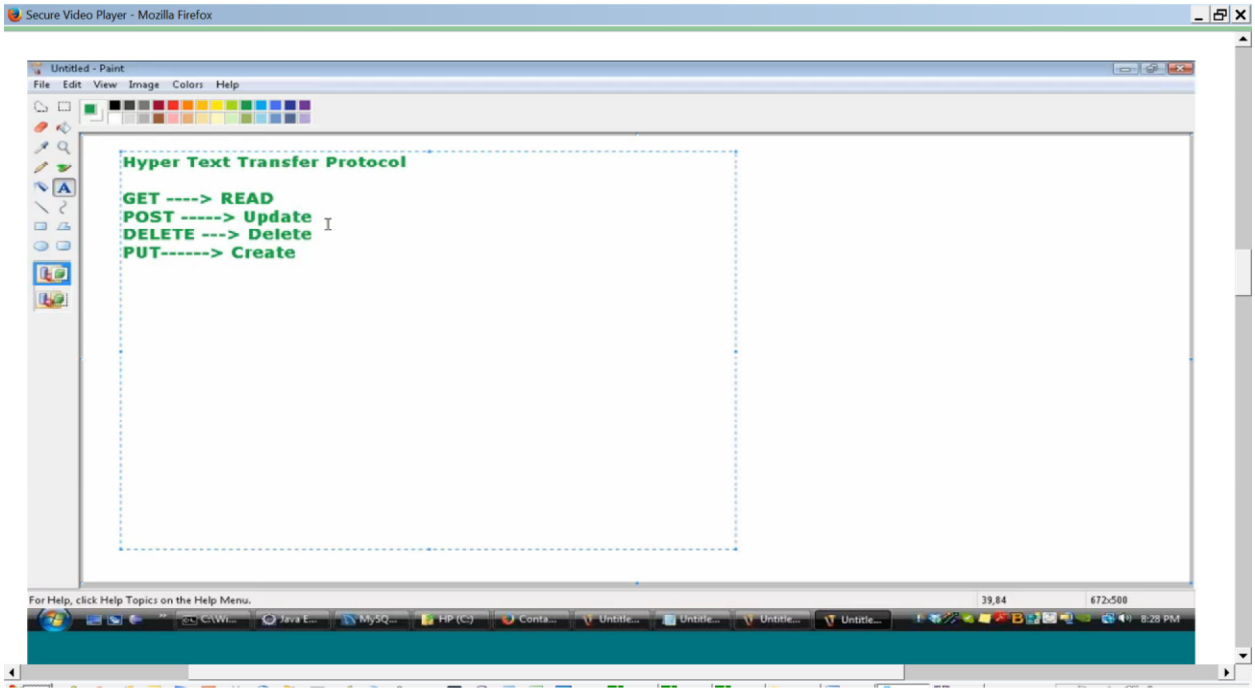
Video url http://vulab.com/portal/play/3693
start at 81:55 since 1st part continued the project starting in training of 20140505

RESTful Web Services

1. Web services are headless web apps which means they have no UI but exchange data in XML or JSON formats for requests and responses. The back ends are the same with respect to domain, DAO and business classes, but controllers are slightly different that Spring MVC
2. RESTful web services – most widely used web services in last 6-7 years with advent of mobile devices
3. REST Web services use case. Imagine a home insurance company that has to track all products owned by customers and has desktop, mobile and tablet apps distributed globally. Desktop will have its own UI written in Java FX, Spring, etc. Mobile and tablet will have Android or Iphone app. All the apps need CRUD functionality for customers and their products.
4. REST stands for representational style.



5. Everything on Internet uses HTTP and is available on every device. How can HTTP be used to implement CRUD? Typically use HTTP GET to click on a link and POST to submit forms. GET is used to read data and POST is used to update or create data. However, HTTP has two more protocols methods that have not been used in last 20 years, DELETE and PUT. Developers saw the opportunity to use GET to read data, POST to update (or create), DELETE to delete and PUT to create (or update). These methods can be used to implement CRUD on any database table for all platforms without introducing a new protocol.



6. RESTful apps are simple. Example shown PizzaREST2014 service. With REST urls are mapped to Class methods with @Path annotation. These methods can receive objects coded in JSON or XML, update or query the database and return objects also coded in JSON or XML. Under the hood the framework automatically converts between java and JSON/XML. JSON by default is built into every browser. JSON stands for javascript object notation. REST methods will be annotated with @GET, @POST, @DELETE or @PUT to match the HTTP method on the url of their @PATH
7. Web apps can access web services and this is the current way. Spring MVC used mostly for desktop apps.
8. Apache crossfire CFX (http://cxf.apache.org/) used to build REST and SOAP web services.
9. (break start at 102:17, end 111:56)

Creating a RESTful Web Project Using Apache CXF in Eclipse

To create a REST web project create a maven project with CXF archetype for REST WS. It will be a completely configured project with a runnable example.

1. File->New->Maven Project
2. In New Maven Project popup click next
3. In Select Archetype popup filter for cxf and select the cxf-jaxrs-service archetype. (The cxf-jaxws-javafirst archetype is for SOAP web services.) Check cxf.apache.org downloads for latest version and add it to Maven archetype library in order to be able to select and use it for a project. Unselect "Show latest version of archetype only" to see additional selections.
4. After archetype has been selected click Next
5. Define group id as com.vulab.code.web and artifact id as the project name, V16-CustomerProductRESTWS-20140507. Define package name same as the goup id.Click Finish to create the project
6. If the new V16-CustomerProductRESTWS-20140507 project shows errors in pom.xml can comment out its <execution> section in
   <plugins> <plugin> <groupId>org.codehaus.mojo</groupId>
7. In src/main/java/com.vulab.code.web/ is HelloWorld.java and JsonBean.java.
8. How do we execute the HelloWorld web service? (141:00)

1. The GET method can be tested with browser
    i. Right click on project and run as -> run on server using tomcat
    ii. In web browser add /hello/echo/string and submit url to return a page with string printed in it
2. The GET method can also be tested with by running HelloWorldIT.testPing() as a Junit test after running the project on an app server (tomcat)
3. The POST method can be tested by running HelloWorldIT.testJsonRoundtrip() as a Junit test after running the project on an app server (tomcat). It can also be tested with an AJAX application running on an external device like an Android

Pom.xml dependencies in V16-CustomerProductRESTWS-20140507

[format: artifactId (groupId){pkg}]

18. cxf-rt-frontend-jaxrs (org.apache.cxf) { javax.ws.rs}
19. jackson-core-asl (org.codehaus.jackson)
20. jackson-jaxrs (org.codehaus.jackson)
21. jackson-mapper-asl (org.codehaus.jackson)
22. junit (junit)
23. spring-web (org.springframework)