5. rész: XXI-ik századi szoftver technológiák

Bakay Árpád
NETvisor kft
(30) 385 1711
arpad.bakay@netvisor.hu

Szoftvertechnológia a J2EE fejlesztés szolgálatában

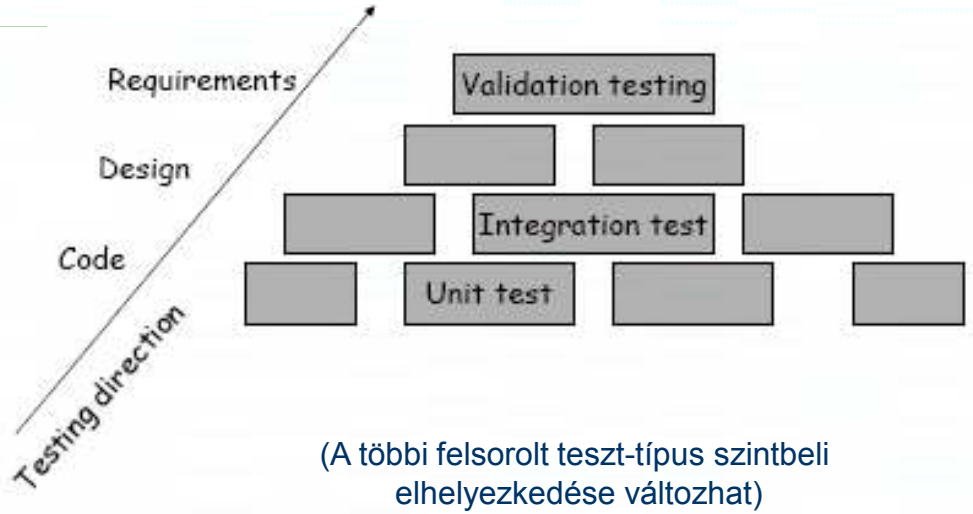
- Web alkalmzások tesztelése
- Build és konfiguráció-kezelés

Tesztelés filozófiai háttere

- Minden műszaki alkotást tesztelni kell!!!
  - A tervezési, kivitelezési hibák és a nem becsülhető körülmények miatt

“Optimism is the occupational hazard of programming; testing is the treatment”

Tesztek fajtái



(A többi felsorolt teszt-típus szintbeli elhelyezkedése változhat)

- Rendszer-szintű
  - Szoftver mellett más is!
- Validációs
  - Specifikációs
- Biztonsági
- Teljesítmény
- Stressz/töréstesztelő
  - 'extrém' feltételek mellett
- Integrációs
  - Modulok együtt
- Modul/unit tesz
  - Kis egységekre
  - A fejlesztő dolga!!!
- Regressziós
  - Utólag bevitt hibákra

Atomatizált tesztelés - emlékeztető

- Mire használható
  - Funkcionális tesztekhez – előre v. párhuzamosan megírt tesztcas-ek kellenek
  - Regressziós tesztekhez – tudja-e még, amit tegnap tudott?
  - Továbbá:
    - load testing: bír e 300 klienst?
    - performance testing: van-e olyan gyors, mint az előző verzió?

JUnit

- Egy keretrendszer tesztek írásához
  - A tesztek TestCase osztályok formájában készülnek

```
class myTest implements junit.framework.TestCase { .. }
```

  - A TestCase implementációban akárhány **`test`**... nevű metódus definiálható
    - További regisztráció nem szükséges, a framework gondoskodik a hívásukról.
  - JDeveloperrel, Eclipse-szel tökéletesen integrált
    - Hogy még kényelmesebb legyen
  - Az IDE integrációnak köszönhetően az eredményeket azonnal áttekinthető formában megjeleníti
    - Végeredmény: zöld vagy piros csík, hiba esetén részletezés
- Zömében modul-tesztek írásához használják
- **Fő a maximális kényelem, különben senki nem fog teszteket írni !**

JUnit - emlékeztető

- **Test Case:** pl. **`YyyTest.java`**
  - Jellemzően egy osztályra/modulra vonatkozó tesztek
  - Több **`testZzz()`** metódust tartalmaz
    - A név formátumából automatikusan azonosítja a teszt-metódusokat
- **Test Suite:** pl. **`XxxxAllTests.java`**
  - Egy egész modul (xxx) összes teszcase-jét sorban végighívja
- **Fixture**
  - Egy test adatkörnyezetét állítja elő
- **`assertPppp()`**
  - Ellenőrző metódusok, ezeket használjuk a test-metódusokban az eredmény ellenőrzésére
    - Ha egy assertion nem teljesül, a framework ezt hibaként regisztrálja
    - Pl.: `assertEquals(returnedList.getLength(), 15);`

HTTPUnit

- Kiterjesztés Web-UI-k szemantikai ellenőrzéséhez
  - A weblat tartalmát igen, de HTML képet nem ellenőrzi
    - Ez utóbbi amúgy is browser-függő, csak manuális teszt lehetséges…
- Junittal használjuk, de ez már nem modul-szintű
  - Az egész alkalmazás tesztje
  - Regressziós tesztekhez különösen hasznos
  - A teszteknek folytonosan követni kell a szoftver fejlődését
    - Folytonos karbantartás esetén ez elviselhető teher
- Éles, futó alkalmazások **felügyeletére** is használható
  - Pl. óránként egy szimulált tranzakció egy távoli kliensről, hiba esetén email és SMS az operátornak

HTTPUnit alapvető osztályok

- WebConversation class
  - WebRequest class
    - setParameter(name,value)
  - WebResponse class
    - Plain textként vagy parsolva ellenőrizhető
      - WebForm class
  - WebLink class
      - Click() method
- Frame-ek, táblázatok is támogatottak

Egyszerű példa

```
public void testWebapp() throws Exception {
      WebConversation wconv = new WebConversation();
      WebRequest  wreq = new GetMethodWebRequest(
       "http://telco.ikkk.inf.elte.hu:8888/PGY3TechSelectJSF/faces/login.jsp");
      WebResponse wresp = wconv.getResponse( wreq );
      assertTrue( "Cannot display login page", wresp.getText().indexOf( "laszt" ) != -1 );

// login page displayed
      System.out.println( wresp.getText() );              // a HTML-t stringként is olvashatjuk
      WebForm loginForm = wresp.getForms()[0];            // de az elemek (tag-ok) között is navigálhatunk
      wreq = loginForm.getRequest();                      // a form-hoz tartalmazó URL-re vonatkozó request
      wreq.setParameter( "form1:usrName", "ABCABCT" );
      wreq.setParameter( "form1:passwd", "p965" );
      wresp = wconv.getResponse( wreq );                  // elküldjük a login kérést

//main page displayed
       assertTrue( "Login not accepted", wresp.getText().indexOf( "Üdvözöljük" ) != -1 );
   }
```

Színkódok:
Junit API
HttpUnit API

JUnit / HTTPUnit fegyelmezett alkalmazásának előnyei

- A tesztek megírására és karbantartására fordítandó idő a kódolás idejének 30-100%-a
- Cserébe kapunk egy dupla biztonságot
  - vö: matek-egyenlet ellenőrzés behelyettesítéssel
- Rendszerint nem bonyolult kód (spagetti), viszont a kliens nézőpontból láttatja az alkalmazást
  - ez sok esetben jobb megértést tesz lehetővé

Népszerű verziókezelők (SCM, VCS)

- **First Tler**
  - **Subversion** **- ingyenes**
  - **CVS** **- ingyenes**
  - **Bazaar** **- ingyenes**
  - Perforce
  - Mercurial
  - StarTeam
  - CM Synergy
  - ClearCase
  - Visual Source Safe
- **Others**
    - Accurev
    - Aegis
    - Arch
    - BitKeeper
    - ClearCase Multisite
    - Code Co-op
    - Darcs
    - Monotone
    - OpenCM
    - PureCM
    - Serena PVCS / Dimension
    - Starteam Enterprise
    - Svk
    - Vesta
- SCM – Software Configuration Manager = VCS – Version Control System

Alkalmazás build az IDE-n túl

- Bonyolult alkalmazásokat nem lehet az IDE-ből megépíteni
  - Speciális előfeldolgozás, csomagolás igénye
    - (pl. generált kód, v. hitelesítés dig. aláírással)
  - Több projekt összeépítése esetén
- Még ha meg is lehet, akkor is szükséges egy script jellegű build tool
  - Pl. az automatizált éjszakai build&test-hez

ANT

- A Java standard build eszköze
  - v.ö: C/C++ make
- Egyetlen konfig file: build.xml
  - v.ö: Makefile
- Kész Java fordítási és futtatási parancsok (taszkok):
  - javac, jar, java
  - de más nyelvekre is használható

build.xml szerkezete

- Target: egy rész-cél
  - depends: más targetek, amiket előbb el kell érni
- Taszk: egy parancs
  - Funkciók: compile, archíve, run, deploy, file.ops, svn stb.
  - (Itt: echo, javac, jar, java)
- Változók használhatók: <property name=„my.var" value=„MyClass"/> , **hivatkozás:** ${my.var}

```
<project name=„myproj" default=„a”>
 <target name=„a" depends=„b,c”>
   <echo message=„Target a”>
   <java classname=„MyClass” classpath=„myJar.jar” >
 </target>
 <target name=„b”>
   <echo message=„Target b”>
    <javac srcdir=„.”>
 </target>
 <target name=„c”>
   <echo message=„Target c”>
   <jar destfile=„myJar.jar” basedir=„.” include=„*.class”>
 </target>
</project>
```

A default target az „a”

de: depedencia az „a” targetben

utóljára ez fut majd le

először

másodikként

„Haladó” funkciók

- RENGETEG built-in taszk (v.ö. köv lap)
- File include
  - A build.xml hivatkozhat más fileokra (ezek gyakran „build method librairy”-k)
- Plugin-ek
  - Plugin-elhetők Java osztályok, amelyek további task-okat definiálnak
  - Pl. Ivy, vagy szerver-specifikus remote deployer metódusok
- Logger, listener
  - A build folyamat ellenőrzésére, néha „debuggolására”
  - defaultLogger, colorLogger, stb.
- IDE Integration
  - Az Eclipse és a Jdeveloper tud ant-tal is buildelni
    (a beépített default build logika helyett)
- Al-processzek
    (az Ant meghív egy másik Ant-ot egy más könyvtárban)

Standard Ant Taszkok

Task típusok (3-10 taszk / típus)

Archive Tasks
Audit/Coverage Tasks
Compile Tasks
Deployment Tasks
Documentation Tasks
EJB Tasks
Execution Tasks
File Tasks
Java2 Extensions Tasks
Logging Tasks
Mail Tasks
Miscellaneous Tasks
.NET Tasks
Pre-process Tasks
Property Tasks
Remote Tasks
SCM Tasks
Testing Tasks
Visual Age for Java Tasks

Bővebben: http://ant.apache.org/manual/index.html

Dependenciák kezelése

- Problémák
  - Egy gépre a forrást letelepítve, azonnal fusson az ANT build
  - Ehhez library-k (jar-ok) kellenek, amiket össze kell keresgélni
    - Tiszta, megbízható forrásból (házi szerver v pl. ibiblio.org)
    - A JAR-ok többsége egy adott verzióban kell
    - Lehetnek köztük saját libraryk, amiket rendszresen frisstünk
    - A libraryknek maguknak is lehet függősége
      - Esetleg version confilict is felléphet

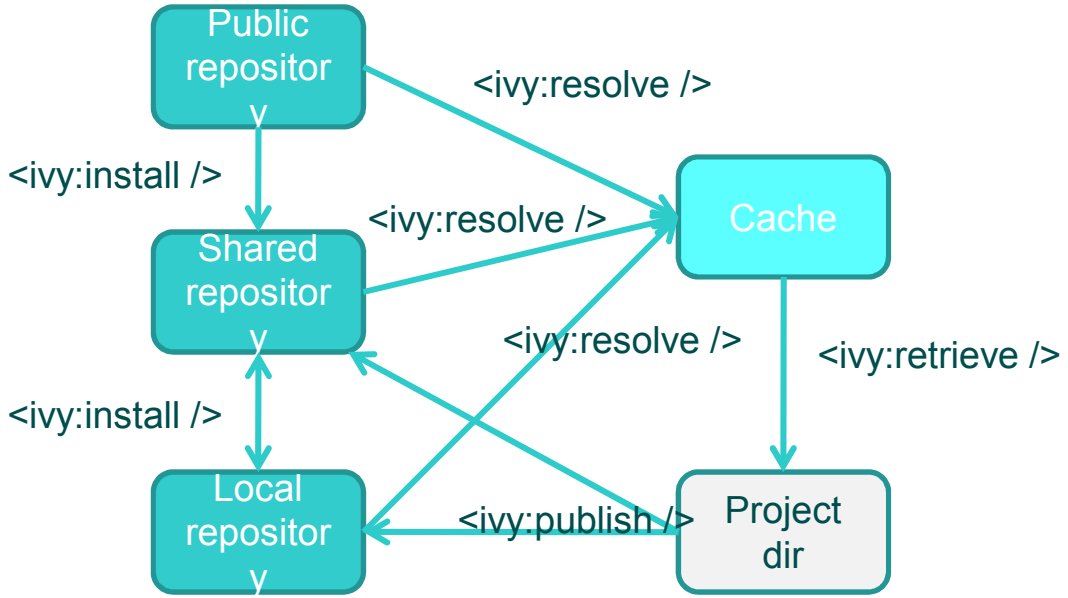
IVY - Dependency Management Tool

- Automatikusan beszerzi a szükséges JAR-okat
  - És az azok által igényelt JAR-okat is…
- Rendszerint az ANT-ba integráltan fut
  - Speciális ant task-ok-kal
  - Van command line felület is
- Többszintű hozzáférés az igénylt JAR-okhoz: pl. public, shared, local
- Version conflict resolution

IVY működése

- **<ivy:resolve>**
  - A szükséges library-ket cache-be gyűjti
  - A bekonfigurált repository-k valamelyikéből
- Ezután 2 út van
  - **<ivy:retireve>** A cache-ből átmásoljuk a build directory-ba
  - **<ivy:cachepath>** Olyan classpath-t készítünk, ami a cache-ből használja a libraryket
- **<ivy:publish>**
  - Jellemzően mások által is használt JAR fejlesztésénél
  - Felrakhatjuk vmelyik repositoryba

Ivy Repository alap műveletek



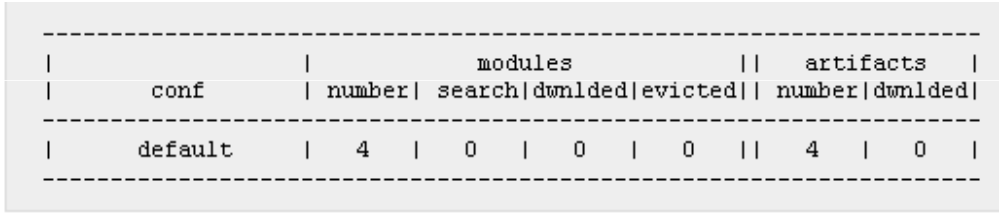
Szokásos tárolási szintek
Public: az interneten
http://ibiblio.org/maven, http://ivyrep.jayasoft.org
Shared: a vállalati szerveren
Local: a saját workspace-ben

Ivy parancsok az Ant build.xml-ben

```
<target name="resolve">
-- Initialize, load config files:
   <ivy:configure/>

-- resolve all files to cache
   <ivy:resolve />

-- copy all objects (default target is the 'lib' dir):
   <ivy:retrieve />
</target>
```



```
---------------------------------------------------------------------
|                  |            modules            ||   artifacts   |
|       conf       | number| search|dwnlded|evicted|| number|dwnlded|
---------------------------------------------------------------------
|      default     |   4   |   0   |   0   |   0   ||   4   |   0   |
---------------------------------------------------------------------
```

Ivy <resolve> status reportja az ANT outputban

Egyszerű build.xml példa

<ivy:cachepath/>: a fileokat itt a cache-ból használjuk

```
<target name="go"  description=" resolve dependencies, compile and run the project">
      <echo message="------------------------- Using ivy to resolve commons-lang 2.1..."/>
<!-- here comes the magic line: asks ivy to resolve a dependency on
           commons-lang 2.1 and to build an ant path with it from its cache  -->
      <ivy:cachepath organisation="apache"
              module="commons-lang" revision="2.1" pathid="lib.path.id"
inline="true"/>

       <echo message="------------------------ Compiling..."/>
       <mkdir dir="${build.dir}"/>
       <javac srcdir="${src.dir}" destdir="${build.dir}" classpathref="lib.path.id"/>

     <echo message="-------------------------- Running.."/>
      <java classname="example.Hello">
          <classpath>
             <path refid="lib.path.id"/>
             <path location="${build.dir}"/>
          </classpath>
       </java>
   </target>
```

Nem kell ivy.xml sem, mert a keresett modult a parancban explicit megadtuk

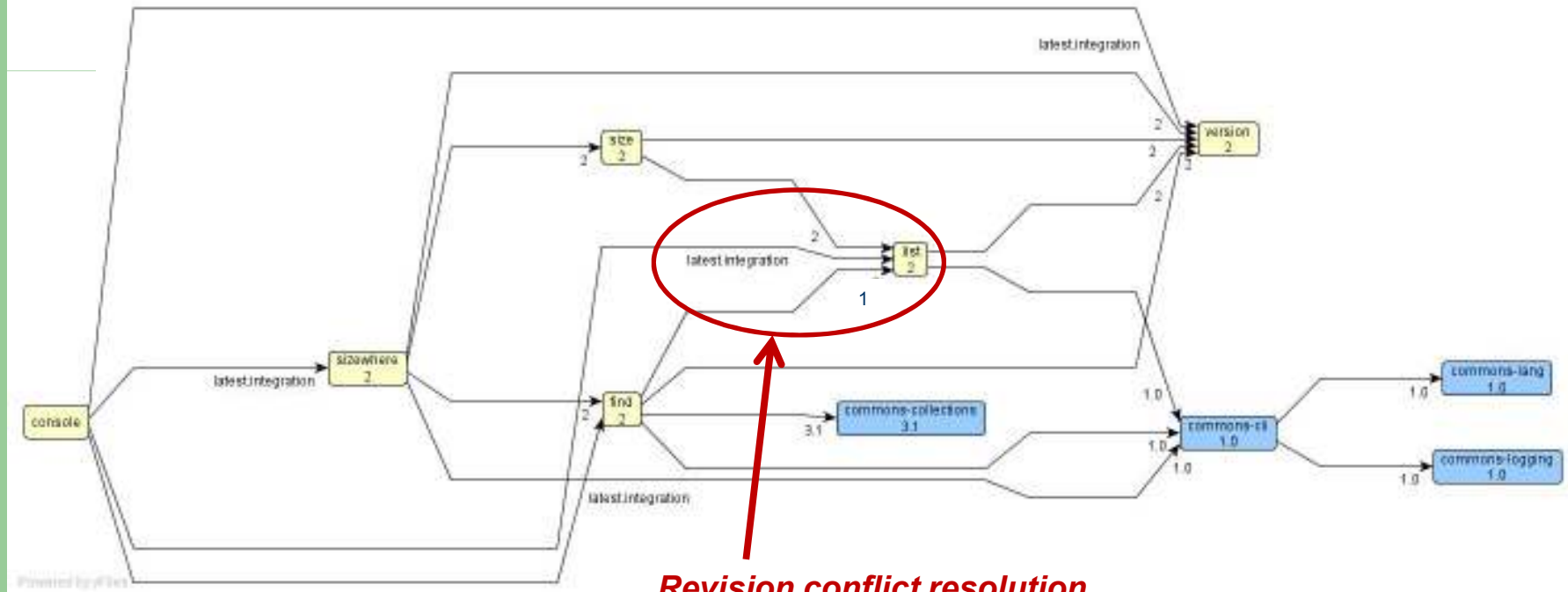
Tipikus modul-dependenciák

ivy.xml (a projekt **build.xml** mellett)

```
<ivy-module version="1.0">
  <info organisation="jayasoft" module="hello-ivy" />
  <dependencies>
    <dependency org="apache" name="commons-lang" rev="2.0" />
  </dependencies>
</ivy-module>
```

Verzió specifikációs szintaxis:

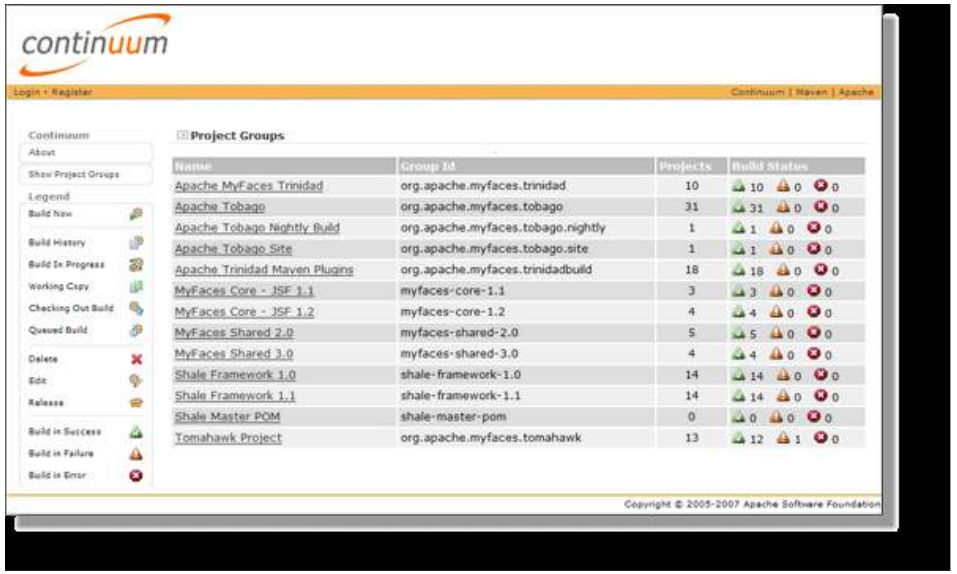
- latest.integration: a lefrissebb
- 1.3.8: pont ez
- 1.3.+: 1.3.x közül a legfrissebb
- [1.0,3.0], 1.0 és 3.0 között
- [1.0,) : minden az 1.0 felett



Revision conflict resolution

Build Management felsőfokon Continuous Integration

- **Az SCM, az ANT/IVY és a Junit/HTTPUnit együttes használata**
- A projekt folyamatos, rendszeres fordítása és teszt-futtatása
  - Azonnal kihozza a legtöbb hibát
  - Teljesítmény-figyelés
    - Nem lett e lassú?
- Keretrendszerek:
    - CruiseControl
    - Continuum
    - AntHill



Continuous Integration tanácsok

- Maintain a Single Source Repository.
- Automate the Build
- Make Your Build Self-Testing
  - Eszközök: *X*unit (Junit), vagy: FIT, Selenium, Sahi, Watir, FITnesse
- Everyone Commits Every Day
- Every Commit Should Build the Mainline on an Integration Machine
- Keep the Build Fast
  - Max. 10-20 minutes
- Test in a Clone of the Production Environment
- Make it Easy for Anyone to Get the Latest Executable
- Everyone can see what's happening
- Automate Deployment

Köszönöm a figyelmet!