Handling Server-Side Software Versioning: the "Smart Technology" Approach

Juris Ivanovs, Datorikas institūts DIVI
Krišs Rauhvargers, University of Latvia
Baltic DB&IS 2012
• Define server-side software contents
• Versioning problem
• Solution to the problem
• Continuous integration solution overview
• Build automation tool overview
• Database schema change handling
• Deployed version validation
• Results
A server-side software

Components

- Configuration
- JavaScript
- CSS
- Pictures
- Libraries and other files
- Desktop application/s
- Server code
- Database
Why is it a problem when doing it by hand?

- Many components – a lot to do
- All files must be up to date
- Including all database changes – tables, columns, indexes etc.
- Not easy to check that the new version is installed correctly
- Many users - can't afford mistakes
- Traceable changes
Difficult job to deploy **one instance**, but what if you have **many instances**?
Server-side software versioning problems (2)

Instances of the same version, but with different configurations and databases

Client 1
- Web application
  - User 1
  - User 2
  - User n

Client 2
- Web application
  - User 1
  - User 2
  - User n

Client n
- Web application
  - User 1
  - User 2
  - User n
Different variations of the same software (e.g. features must not be available to clients not paying for them)
How to handle these problems?

Automate – let the computer do all the routine/manual tasks
Solution requirements

- Must be able to run in different environments – *Windows*, *Linux*
- Adaptable for different web technologies – *PHP*, *.NET*, *Java*
- Build web and desktop applications
- Able but not limited to handling *Oracle* database schema changes
- Automatically validate the deployed software version – validate other people deployment
Existing solutions to the problem

All-in-one solutions

• Exist for specific platforms
  • *Microsoft Visual Studio Team Foundation Server* - .NET
  • *Phing* – *Ant* based, designed for *PHP* applications
• Difficult or even impossible to use in other scenarios that aren't predefined
"Building blocks" to the problem

- There are several tools that help to automate build and deployment – build automation, continuous integration servers
- Not much information can be found in literature about:
  - using these tools for an integrated solution
  - multi-platform solution
  - handling DB schema changes
  - validation of deployed versions
Idea of an integrated solution

Continuous integration server
• Execute and show build results
• Build configuration

Automated build
• Compilation and deployment
• Database comparison
• Creation of version descriptor

Validation of the version
• Deployment validation
• Presentation of results
Continuous integration server selection

Many exist, these were selected:

- CruiseControl
- Continuum
- Quickbuild
- Hudson
Automated build

Also many exist, these were selected:

MAKE

<APACHE ANT>

maven

gradle

a better way to build
Two approaches exist:

• Store any database changes like source code changes and apply these changes upon build, e.g. using *Liquibase*

• Use database comparison:
  • Tools exist, but not executable from command line or expensive
  • Tried to make our own tool to compare *Oracle* databases - difficult
• Using the *smart technology* idea about environment checks
  – Collect "requirements" for software to run
  – Create a tool that checks if each requirement is fulfilled in production environment
  – Run automated tests, when deploying (or re-deploying) software into new environment

• A checking tool created by ourselves (for PHP in our case) – check files and configuration
Overall architecture of the solution
• The approach has been successfully applied in several projects

• Time saved:
  • 2/3 of times is saved due to automation
  • approximately 1/3 of human effort is required for version deployment – hard to optimize more
  • In one of projects close to 1 man-month is saved during a year
Questions?