Guicing Up Selenium With Docker!

What is the concept?

- Tests should be deterministic, repeatable and accurately model the production environment
- Continuous Integration and Immutable Servers are the goal
- Docker containerization makes repeatable test environments easy to manage
- JUnit and Selenium provide familiar testing tools
- Giulius uses Guice dependency injection to makethe system modular and easier to manage

What the heck is Guilius?

- A collection of projects for loading configuration files, binding them using Guice and writing boilerplate-free JUnit tests of Guice code
- Created by long-time Java contributor, Tim Boudreau, based on work by Eelco Hillenius
- Builds up test environments using injected dependencies
- Container initialization can be integrated into test runs

Useful for?

- We built the system primarily for integration testing of web applications
- Our environment is based on Apache Open for Business and uses a lot of server side code written in Groovy
- Eases rapid interative testing and development
- Easily reset anything that stores state in the filesystem: databases, full text search, uploads, etc.

Benefits

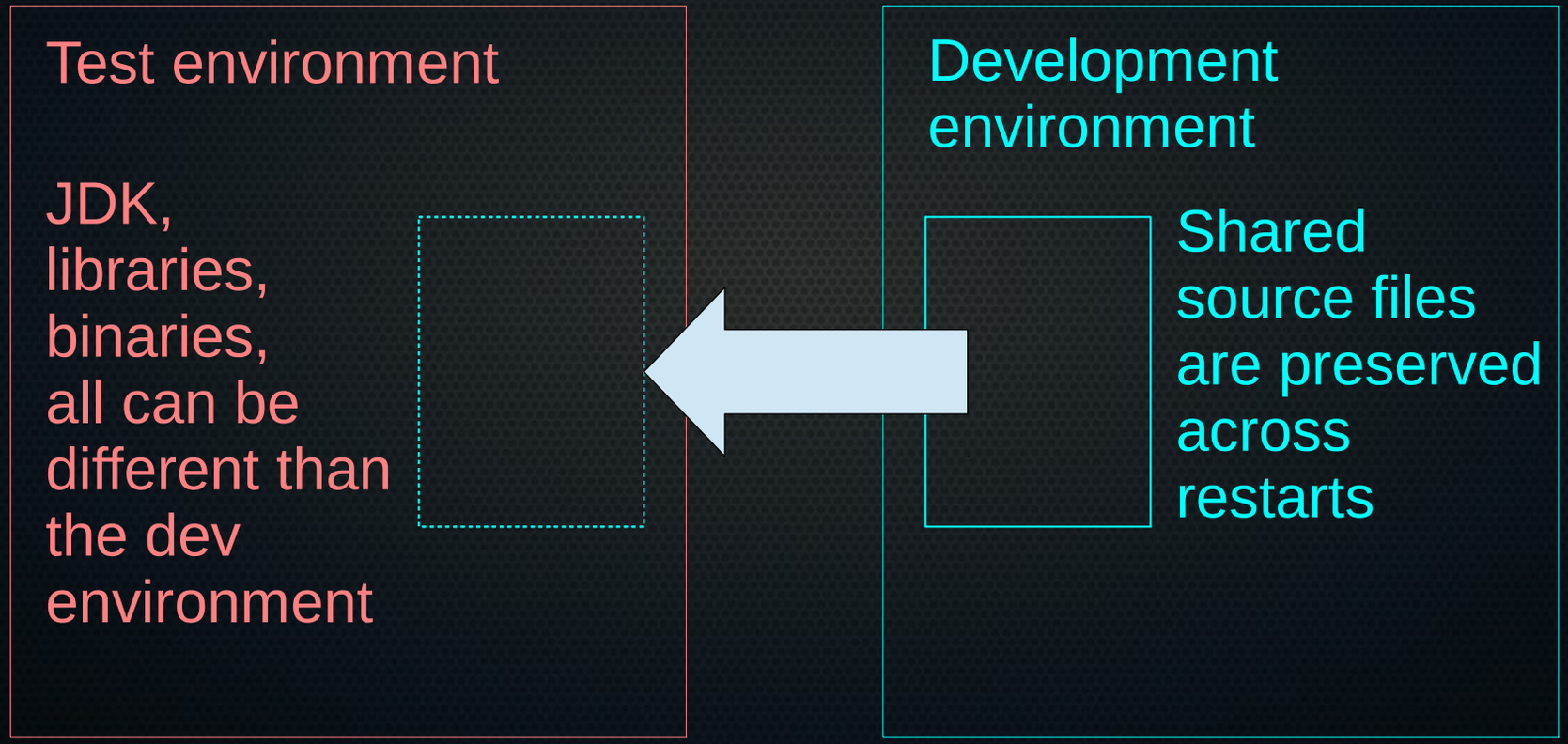
- Each test runs in a pristine copy of the application without any influence from previous runs
- All state is reset: legacy applications or anything with file system state is intact
- The container can be isolated and complex production network environments can be simulated
- Mock high level services such as email can be provided

Docker

- Popular and mature infrastructure for managing virtualization
- Overlay file system allows direct editing of source in the test container
- Allows the production environment to be simulated in the desktop environment
- Docker does impose its own concepts about how a system works (ie. no init.d)

Development Process

Docker: File system overlay



Guilius style injection

```
@RunWith(SeleniumRunner)
@TestWith([VideoModule, TestsModule])
@Fixtures([TestDatabase, ApprovedUser])
@Suites("cool-tests")
public class MakeSureCoolThingWorksTest {
… driver.findElement(By.byClass('cool-button')).click() …
… assert something … etc ...
}
```

Guilius style injection

bind(TestDatabase.class).to(ECommerceQ12014.class);

bind(ApprovedUser.class).to(GoodCreditUser.class);

bind(CreditProcessor.class).to(MockCreditProcessor.class);
… etc …

These can be rebound for the entire test system and will be dynamically injected into many tests at run-time.

Performance concerns and strategies

- Constantly rebuilding the container can be network intensive
  - Run a local mirror for packages
  - Design Dockerfiles and other initialization scripts to look for and use http_proxy
  - Design Dockerfile layers to build actively changing resources later in the process so that early steps can cache
- SSDs make a **huge** difference for rebuild performance, especially if the packages are cached on the local disk or network.

Demo

Future directions

- Ability to bind combinations of configured fixtures into a Docker snapshot so that expensive fixture construction can be preserved
- Analysis of fixture combinations to automate acceleration of large test suites through snapshot creation
- Use of other container technologies for both desktop and large-scale testing (ie. LXC, OpenStack, AWS, etc.)
- Integration with Continuous Integration (ie. Jenkins, GitLab CI, etc.)

Thank you!

- More info:
  - http://schu.es/guicing-up-selenium-with-docker
  - http://github.com/schue/selenium-guice
  - http://twitter.com/schue
  - ean@brainfood.com