Magento Performance Toolkit Customization

@SergiiShymko

Software Engineer, Magento

MagentoLive

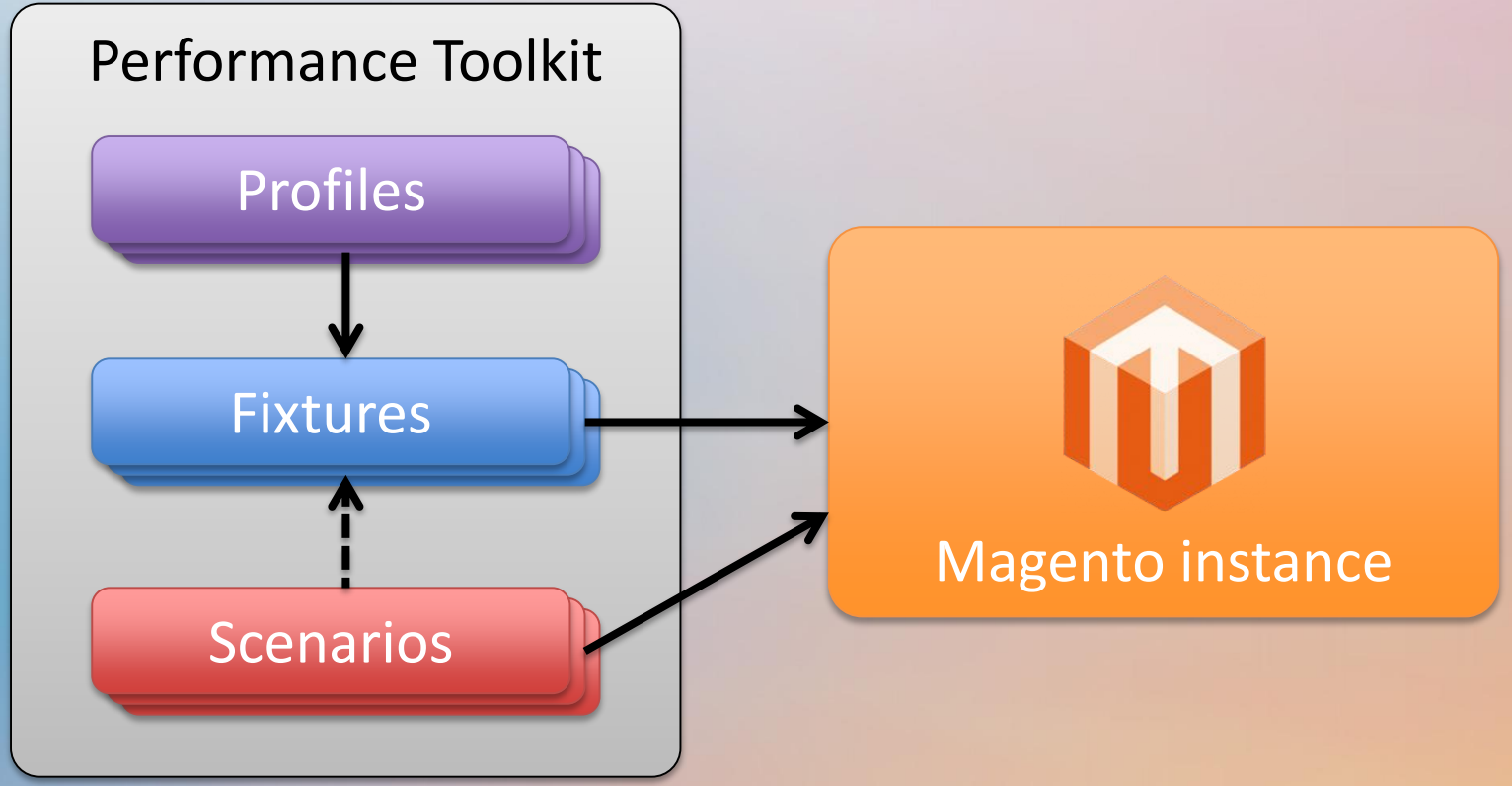
UK | 2015

Magento Performance Toolkit

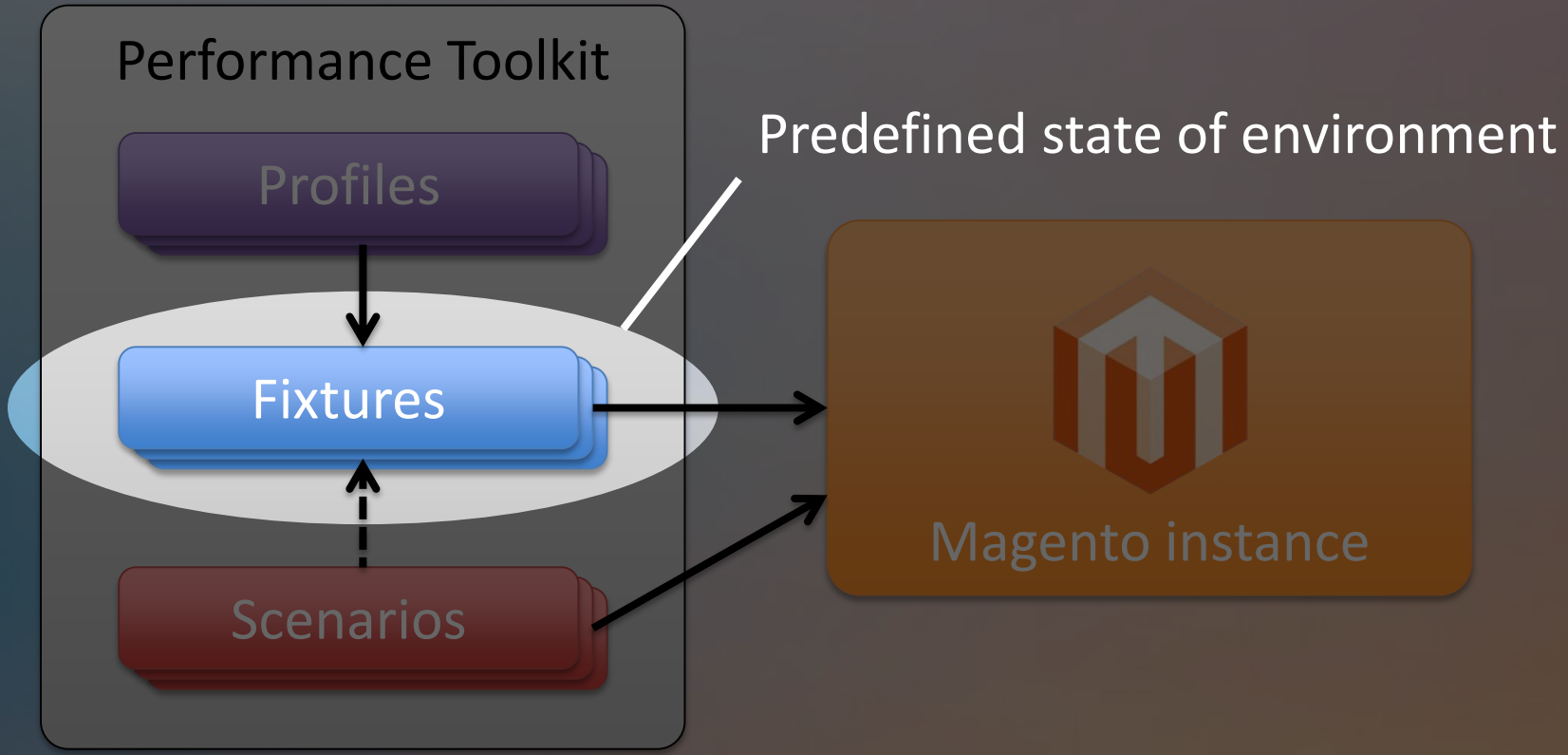
- Available at GitHub
  github.com/magento/magento-performance-toolkit
- Designed for EE 1.14, 1.13, 1.12
- Compatible with CE*
- Features:
  - Generation of sample data
  - Imitation of user activity
  - Comparing performance before/after changes

** – after some tweaks ;-)*

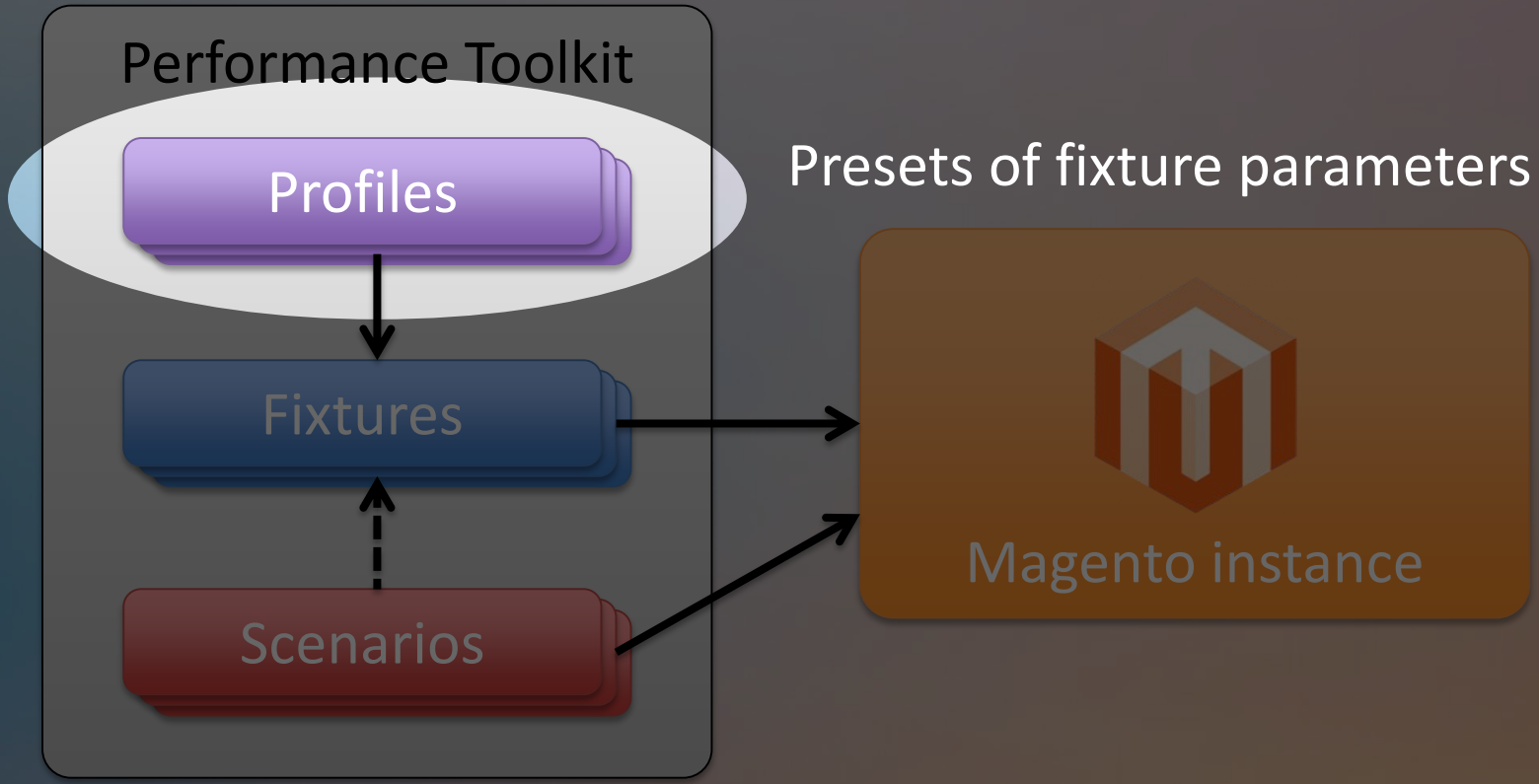
Architecture Overview



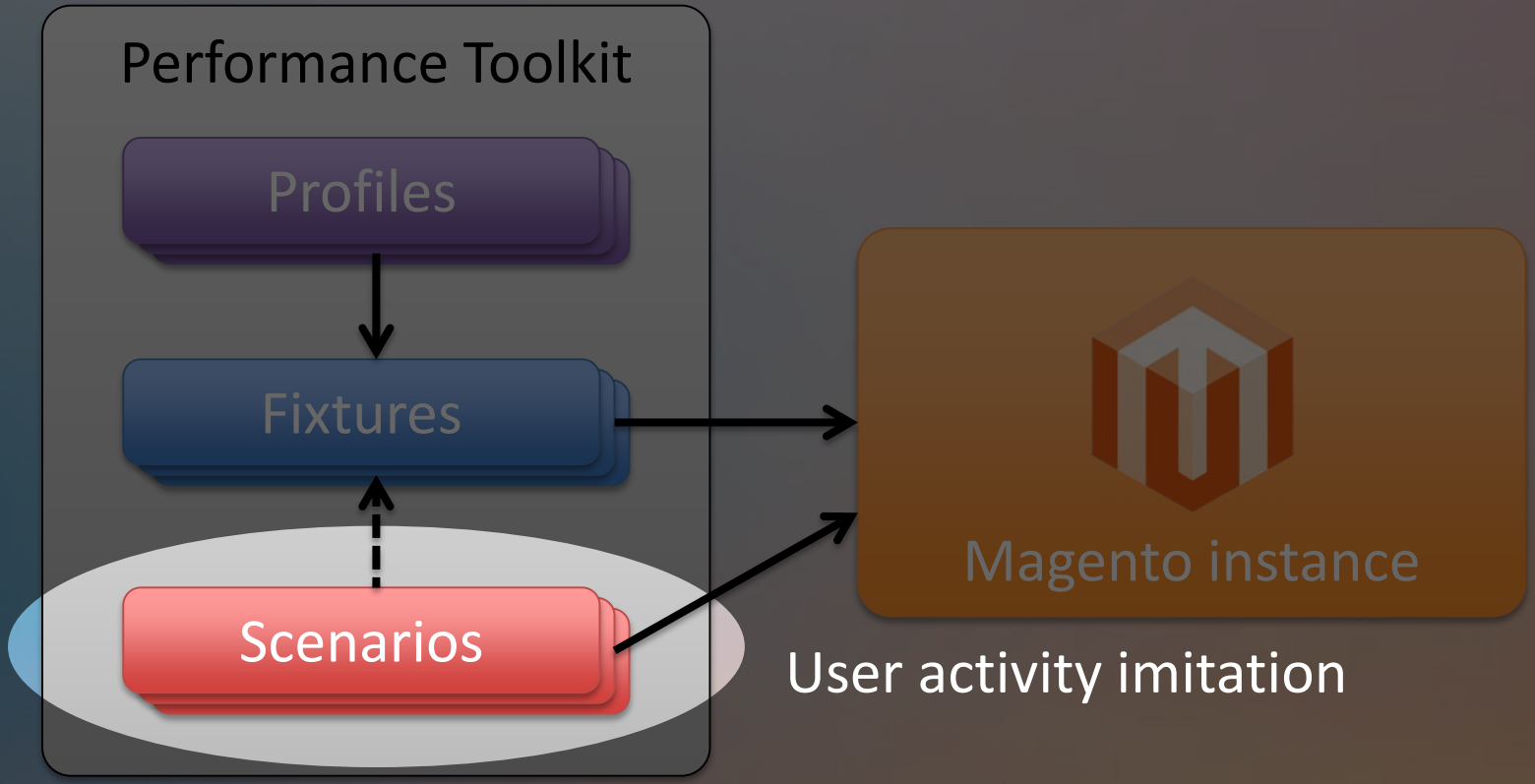
Architecture: Fixtures



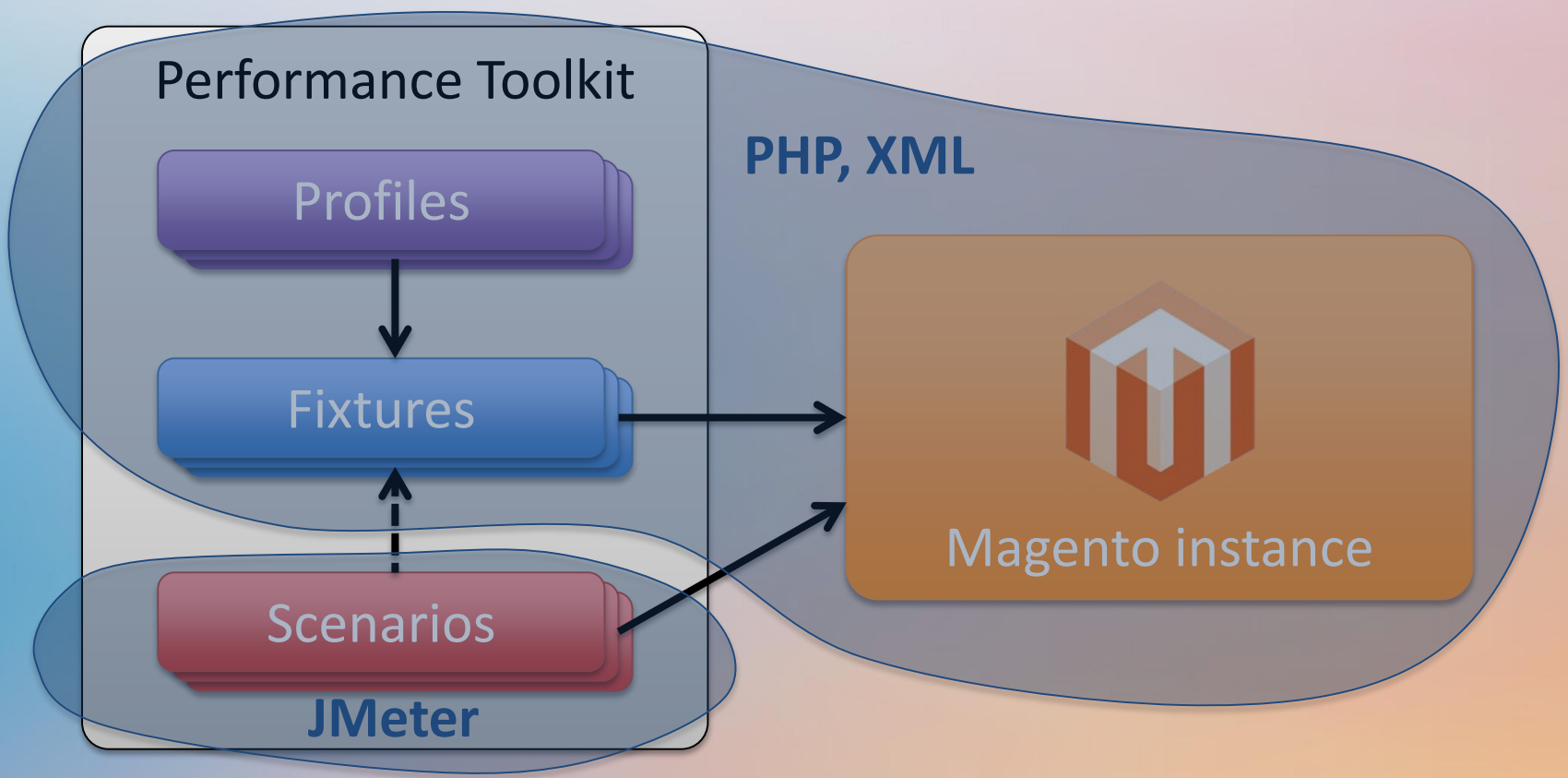
Architecture: Profiles



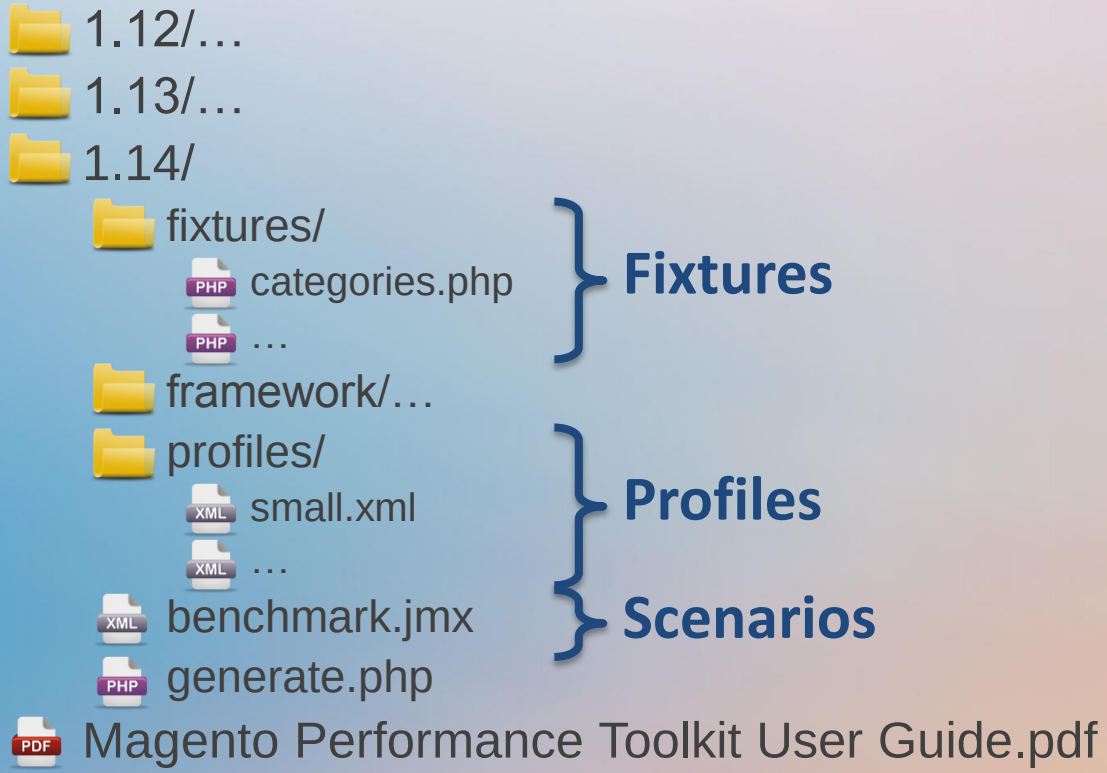
Architecture: Scenarios



Technologies



Repository Structure



Usage

1. Install JMeter
2. Install JMeter plugins Standard and Extras
3. Locate Magento instance
4. Extract toolkit files to
   `<magento_dir>/dev/tools/performance_toolkit/`
5. Generate fixture data
   `php -f generate.php -- --profile=<profile_path>`
6. Run JMeter scenario
   ```
   jmeter -n -t benchmark.jmx
       -Jhost=magento.example.com -Jbase_path=/
   ```
7. Compare results with previous

Customizing the Toolkit

Customizing Fixtures

Customizing Profiles

Customizing Scenarios

How to Create a Custom Fixture

1. Create fixture script file
   `fixtures/<fixture_name>.php`
2. Register fixture in
   `framework/fixtures.xml`
3. Describe fixture parameters in
   `framework/labels.xml`

Fixture Script Implementation

✓Use Magento API

Create from scratch

1. Log data when saving entity in the admin panel
2. Clean up logged data
3. Substitute variable data
4. Parameterize fixture

Reuse existing code

✓ Adopt fixtures from Magento 2

- ✓ Performance tests
- ✓ Integration tests

Fixture Example

fixtures/product_virtual.php

```php
$product = Mage::getModel('catalog/product');
$product->setData(array(
    'type_id'      => \Mage_Catalog_Model_Product_Type::TYPE_VIRTUAL,
    'name'         => 'Virtual Product 1',
    'sku'          => 'virtual_product_1',
    'price'        => 10,
    'visibility'   => \Mage_Catalog_Model_Product_Visibility::VISIBILITY_BOTH,
    'status'       => \Mage_Catalog_Model_Product_Status::STATUS_ENABLED,
    // ...
));
$product->save();
```

Parameterized Fixture

fixtures/products_virtual.php

```
$count = \Magento\ToolkitFramework\Config::getInstance()->getValue('virt_products', 1);
for ($i = 1; $i <= $count; $i++) {
    $product = Mage::getModel('catalog/product');
    $product->setData(array(
        'type_id'     => \Mage_Catalog_Model_Product_Type::TYPE_VIRTUAL,
        'name'        => 'Virtual Product ' . $i,
        'sku'         => 'virtual_product_' . $i,
        'price'       => 10,
        'visibility'  => \Mage_Catalog_Model_Product_Visibility::VISIBILITY_BOTH,
        'status'      => \Mage_Catalog_Model_Product_Status::STATUS_ENABLED,
        // ...
    ));
    $product->save();
}
```

Fixture Registration

framework/fixtures.xml

```
<fixtures>
    <products_virtual>
        <file>products_virtual.php</file>
        <action>Generating virtual products</action>
    </products_virtual>
</fixtures>
```

framework/labels.xml

```
<config>
    <labels>
        <virt_products>Number of virtual products</virt_products>
    </labels>
</config>
```

Customizing Profiles

How to Create Custom Profiles

How to Create a Custom Profile

profiles/<profile_name>.xml

```
<config>
    <profile>
        <websites>1</websites>
        <store_groups>1</store_groups>
        <store_views>1</store_views>
        <simple_products>800</simple_products>
        <configurable_products>50</configurable_products>
        <virt_products>100</virt_products>
        <categories>30</categories>
        <categories_nesting_level>3</categories_nesting_level>
        <catalog_price_rules>10</catalog_price_rules>
        <catalog_target_rules>2</catalog_target_rules>
        <cart_price_rules>10</cart_price_rules>
        <cart_price_rules_floor>2</cart_price_rules_floor>
        <customers>20</customers>
    </profile>
</config>
```

Customizing Scenarios

How to Create Custom Scenarios

JMeter Scenario Implementation

✓Use JMeter GUI

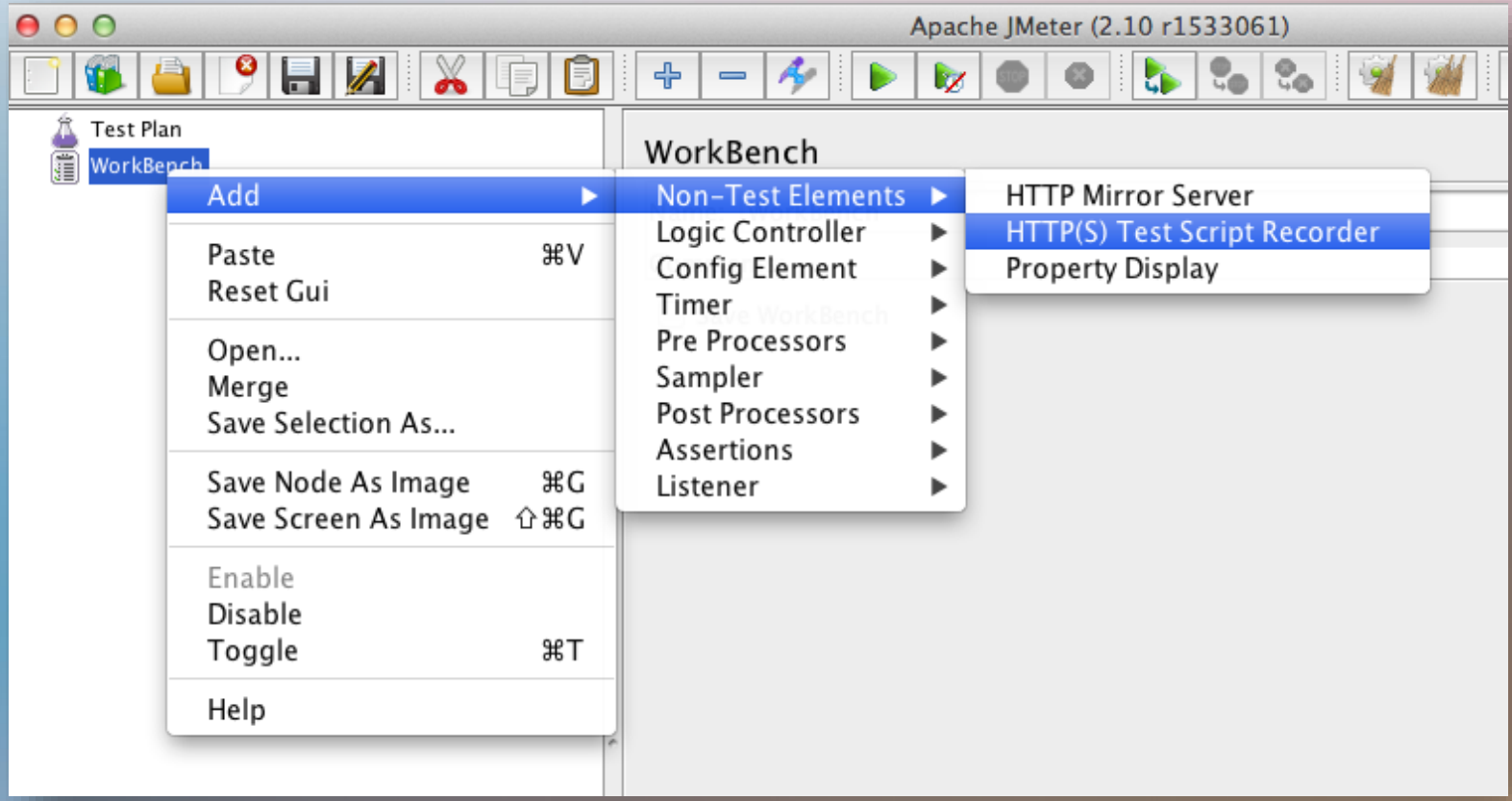
Create from scratch

1. Set up JMeter to act as HTTP proxy server
2. Configure browser to use proxy
3. Record your manual actions
4. Clean up recorded requests
5. Substitute variable data
6. Parameterize scenario

Reuse existing code

- ✓ Extract scenarios from `benchmark.jmx`
- ✓ Adopt scenarios from Magento 2 performance tests

JMeter GUI


Apache JMeter (2.10 r1533061)
Test Plan
WorkBench
WorkBench
Add
Paste ⌘V
Reset Gui
Open...
Merge
Save Selection As...
Save Node As Image ⌘G
Save Screen As Image ⇧⌘G
Enable
Disable
Toggle ⌘T
Help
Non-Test Elements
Logic Controller
Config Element
Timer
Pre Processors
Sampler
Post Processors
Assertions
Listener
HTTP Mirror Server
HTTP(S) Test Script Recorder
Property Display

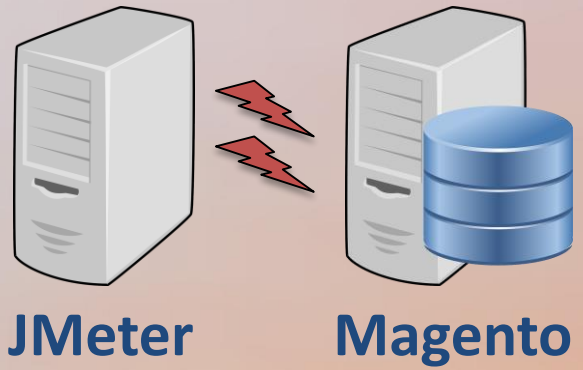
Parameterized Scenario

- Accept input parameters
  `${__P(<param_name>, <default_value>)}`
- Reuse parameters of `benchmark.jmx`
  `host`, `base_path`, etc.
- Run with custom parameters

```
jmeter -n -t <scenario_name>.jmx
     -J<param_name>=<param_value>
```

Summary & Tips

- Fixture creation
  - Use Magento models
  - Use import/export API
  - Adopt fixtures from Magento 2
- Scenario creation
  - Use JMeter script recorder
  - Disable secret keys in URLs
  - Adopt scenarios from Magento 2
- Infrastructure
  - Run JMeter on a separate server

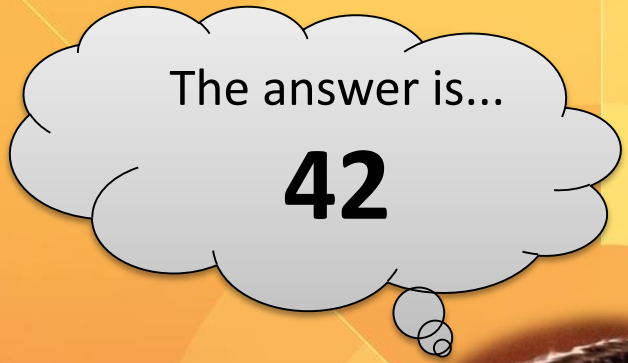


Thank You!

Q & A

@SergiiShymko

sshymko@ebay.com



MagentoLive

UK | 2015

Resources

- JMeter and plugins
  jmeter.apache.org
  jmeter-plugins.org
- Magento Performance Toolkit for Magento 1.x
  github.com/magento/magento-performance-toolkit
- Automated Tests for Magento 2
  github.com/magento/magento2/tree/develop/dev/tests/