agcy 2011

April 13-16, 2011. Oxford, UK

Writing web UI in Testfirst fashion with GWT

Uberto Barbini
uberto@ubiland.net
twitter: @ramtop

About me

Uberto Barbini

Software artisan

Agile enthusiast.

Hobby:
photography and the game of Go.

Teamleader and Architect for Vodafone editorial and backend products.

Editorial platform for big company:

4-5 people team

agile (scrum and kanban)

gwt, hibernate, smartgwt

started jan 2009

version 1.0 june 2009, now 3.2

total lines of code: ~2.000.000

test coverage: ~60%

http://netnumero.com

Net Numero - Business Bookkeeping

http://netnumero.appspot.com/company/mycompany#dashboard

Google

Apple Yahoo! Google Maps YouTube Wikipedia News (349) Popular

My Account Settings Help

April 10, 2011 9:55:40 AM | Logout

Beta

Dashboard | Money In | Money Out | Clients | Staff | Time | Accounts | Reports | ?

Dashboard

Welcome to the real time dashboard, here you will find a summary of your latest transactions and shortcuts to commonly used functions.

I would like to...

Create an Invoice > | Create an Estimate > | Track Time > | Generate a Report >

Money In

Key Figures

Manage your invoices, estimates, products or services here.

Latest Transaction

Invoice	PDF	Client	Total	Date Raised
InvoiceId16		wewew	0.00GB£	2/6/11
InvoiceId15		Uberto gama	120.00GB£	2/6/11
InvoiceId14		Uberto gama	120.00GB£	2/6/11
InvoiceId---		test	857.80GB£	1/29/11

Money Out

Key Figures

Manage your expenses here.

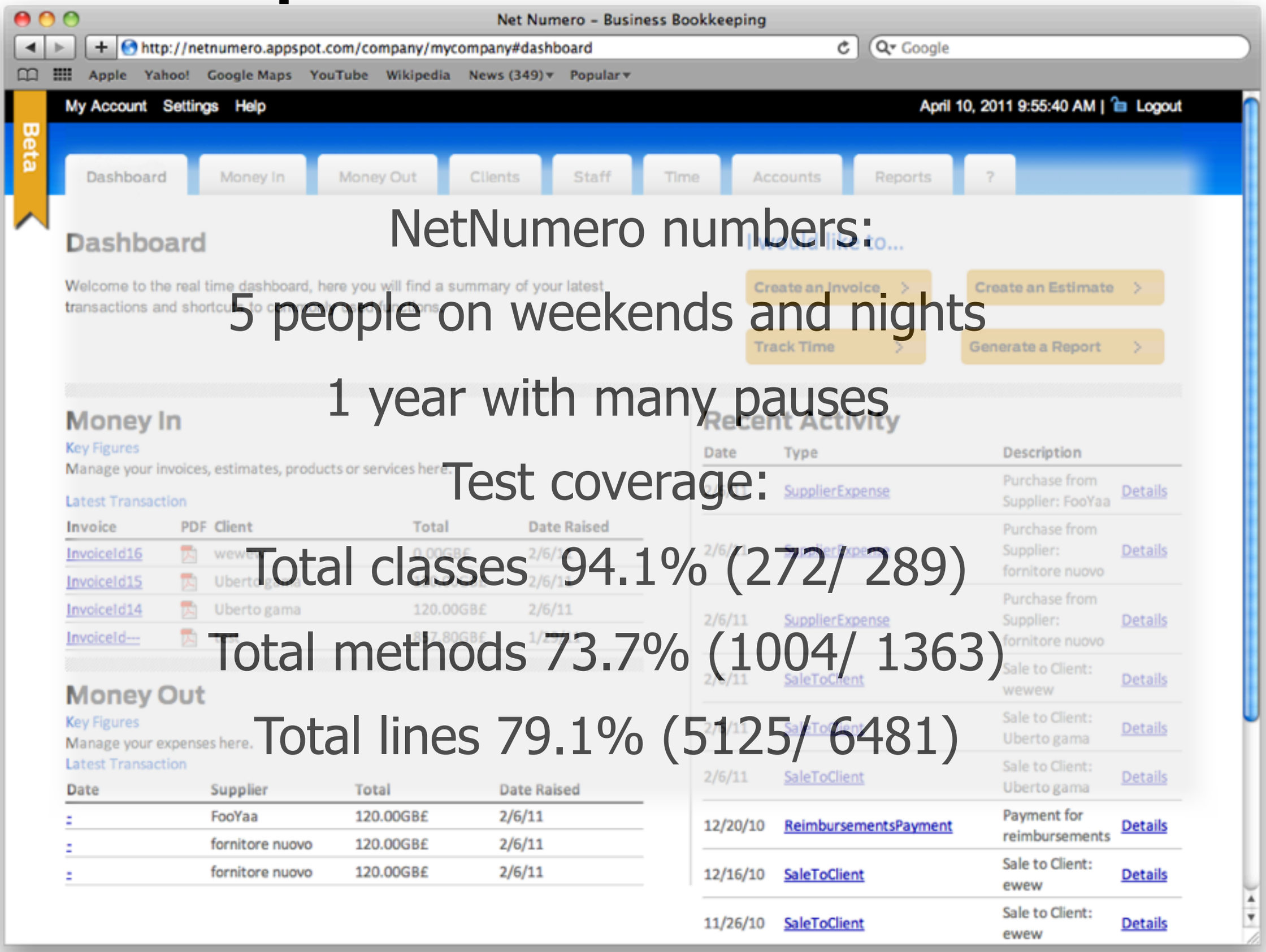
Latest Transaction

Date	Supplier	Total	Date Raised
-	FooYaa	120.00GB£	2/6/11
-	fornitore nuovo	120.00GB£	2/6/11
-	fornitore nuovo	120.00GB£	2/6/11

Recent Activity

Date	Type	Description	
2/6/11	SupplierExpense	Purchase from Supplier: FooYaa	Details
2/6/11	SupplierExpense	Purchase from Supplier: fornitore nuovo	Details
2/6/11	SupplierExpense	Purchase from Supplier: fornitore nuovo	Details
2/6/11	SaleToClient	Sale to Client: wewew	Details
2/6/11	SaleToClient	Sale to Client: Uberto gama	Details
2/6/11	SaleToClient	Sale to Client: Uberto gama	Details
12/20/10	ReimbursementsPayment	Payment for reimbursements	Details
12/16/10	SaleToClient	Sale to Client: ewew	Details
11/26/10	SaleToClient	Sale to Client: ewew	Details

http://netnumero.com



NetNumero numbers:

5 people on weekends and nights

1 year with many pauses

Test coverage:

Total classes 94.1% (272/ 289)

Total methods 73.7% (1004/ 1363)

Total lines 79.1% (5125/ 6481)

Four Noble Truths

of Buddhism

Sunday, April 17, 2011

Four Noble Truths

of Buddhism

- Suffering exists
- Suffering arises from attachment to desires
- Suffering ceases when attachment to desire ceases
- Freedom from suffering is possible by practising the Eightfold Path

Four Noble Truths

of complex* web applications development

Text

* *If your url can contain all the app state is not a complex application. Just use Rest*

Four Noble Truths

of complex* web applications development

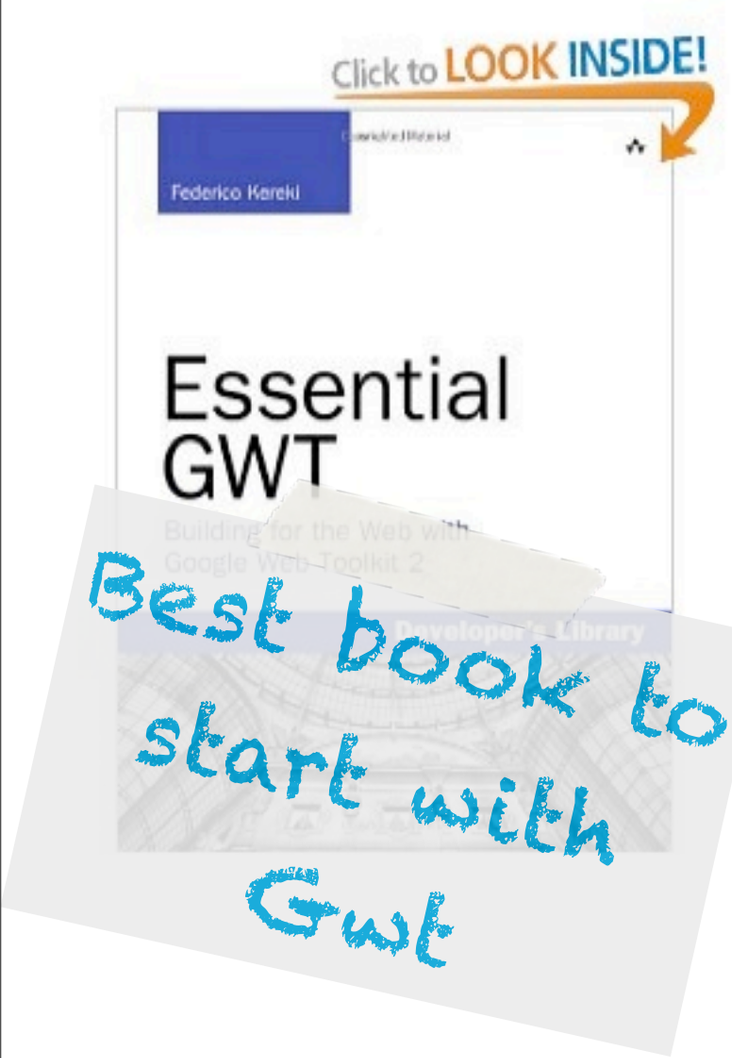
- Suffering exists when writing complex web application
- Suffering arises from user state attachment on the server
- Suffering ceases when attachment to user status ceases
- Freedom from suffering is possible by practicing the GWT Path

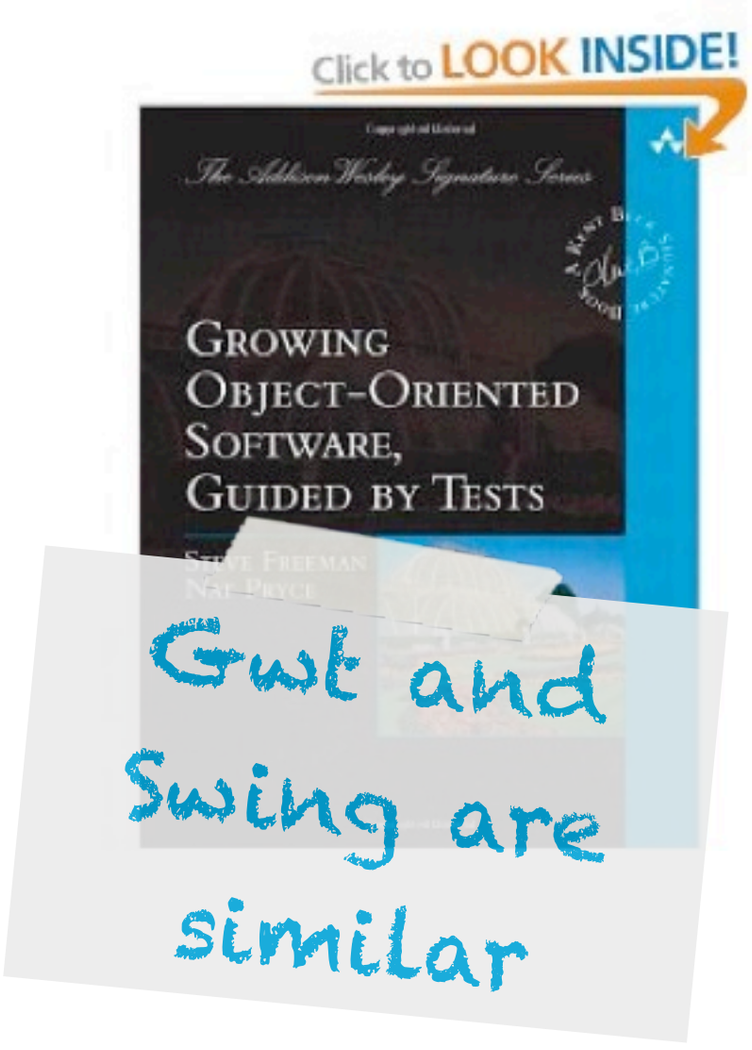
** If your url can contain all the app state is not a complex application. Just use Rest*

What's GWT anyway?


What's GWT anyway?
Short version:
Java to Javascript compiler

Where to start





Many video on Google like:

http://www.google.com/events/io/2009/sessions/GoogleWebToolkitBestPractices.html

GWT magic I
The compiler

- Directories will be created containing the JavaScript implementation of your project. One directory for each module.

```
C:\gwt-2.0.0\samples\Hello>ant
Buildfile: build.xml

libs:

javac:

gwtc:
     [java] Compiling module com.google.gwt.sample.hello.Hello
     [java]    Compiling 5 permutations
     [java]       Permutation compile succeeded
     [java]    Linking into war
     [java]       Link succeeded
     [java]    Compilation succeeded -- 20.313s

build:

BUILD SUCCESSFUL
Total time: 22 seconds
```

GWT magic I
The compiler

- Directories will be created containing the JavaScript implementation of your project. One directory for each module.

```
C:\gwt-2.0.0\samples\Hello>ant
Buildfile: build.xml

libs:

javac:

gwtc:
     [java] Compiling module com.google.gwt.sample.hello.Hello
     [java]    Compiling 5 permutations
     [java]       Permutation compile succeeded
     [java]    Linking into war
     [java]       Link succeeded
     [java]    Compilation succeeded -- 20.313s

build:

BUILD SUCCESSFUL
Total time: 22 seconds
```

5 x supported browser x 1 locale.
6 Browsers in 6 languages
= 36 permutations!

GWT magic 1
The compiler

- Directories will be created containing the JavaScript implementation of your project. One directory for each module.

```
C:\gwt-2.0.0\samples\Hello>ant
Buildfile: build.xml

libs:

javac:

gwtc:
     [java] Compiling module com.google.gwt.sample.hello.Hello
     [java]    Compiling 5 permutations
     [java]       Permutation compile succeeded
     [java]    Linking into war
     [java]       Link succeeded
     [java]    Compilation succeeded -- 20.313s

build:

BUILD SUCCESSFUL
Total time: 22 seconds
```

5 x supported browser x 1 locale.
6 Browsers in 6 languages
= 36 permutations!

22 seconds for an helloworld.
Several minutes for real projects.

NetNumero compilation:

```
Compiling 10 permutations
     Compiling permutation 0...
     Compiling permutation 1...
     Compiling permutation 2...
     Compiling permutation 3...
     Compiling permutation 4...
     Compiling permutation 5...
     Compiling permutation 6...
     Compiling permutation 7...
     Compiling permutation 8...
     Compiling permutation 9...
   Compile of permutations succeeded
Linking into /Users/ubertobarbini/svijava/netnumero/
war/application. Writing extras to /Users/
ubertobarbini/svijava/netnumero/extra/application
   Link succeeded
   Compilation succeeded -- 354.190s
```

NetNumero compilation:

5 Browsers x 2 Languages

```
Compiling 10 permutations
      Compiling permutation 0...
      Compiling permutation 1...
      Compiling permutation 2...
      Compiling permutation 3...
      Compiling permutation 4...
      Compiling permutation 5...
      Compiling permutation 6...
      Compiling permutation 7...
      Compiling permutation 8...
      Compiling permutation 9...
   Compile of permutations succeeded
Linking into /Users/ubertobarbini/svijava/netnumero/
war/application. Writing extras to /Users/
ubertobarbini/svijava/netnumero/extra/application
   Link succeeded
   Compilation succeeded -- 354.190s
```

NetNumero compilation:

```
Compiling 10 permutations
      Compiling permutation 0...
      Compiling permutation 1...
      Compiling permutation 2...
      Compiling permutation 3...
      Compiling permutation 4...
      Compiling permutation 5...
      Compiling permutation 6...
      Compiling permutation 7...
      Compiling permutation 8...
      Compiling permutation 9...
   Compile of permutations succeeded
Linking into /Users/ubertobarbini/svijava/netnumero/
war/application. Writing extras to /Users/
ubertobarbini/svijava/netnumero/extra/application
   Link succeeded
   Compilation succeeded -- 354.190s
```

5 Browsers x 2 Languages

about 6 minutes

After running the GWT compiler your war directory should look something like this:

```
C:\gwt-2.0.0\samples\Hello>\bin\find war
war
war\hello
war\hello\18EEC2DA45CB5F0C2050E2539AE61FCE.cache.html
war\hello\813B962DC4C22396EA14405DDEF020EE.cache.html
war\hello\86DA1DCEF4F40731BE71E7978CD4776A.cache.html
war\hello\A37FC20FF4D8F11605B2C4C53AF20B6F.cache.html
war\hello\E3C1ABB32E39A126A9194DB727F7742A.cache.html
war\hello\14A43CD7E24B0A0136C2B8B20D6DF3C0.cache.png
war\hello\548CDF11D6FE9011F3447CA200D7FB7F.cache.png
war\hello\9DA92932034707C17CFF15F95086D53F.cache.png
war\hello\A7CD51F9E5A7DED5F85AD1D82BA67A8A.cache.png
war\hello\B8517E9C2E38AA39AB7C0051564224D3.cache.png
war\hello\clear.cache.gif
war\hello\hello.nocache.js
war\hello\hosted.html
war\Hello.html
```

Your 5 permutations of compiled JS

After running the GWT compiler your war directory should look something like this:

```
C:\gwt-2.0.0\samples\Hello>\bin\find war
war
war\hello
war\hello\18EEC2DA45CB5F0C2050E2539AE61FCE.cache.html
war\hello\813B962DC4C22396EA14405DDEF020EE.cache.html
war\hello\86DA1DCEF4F40731BE71E7978CD4776A.cache.html
war\hello\A37FC20FF4D8F11605B2C4C53AF20B6F.cache.html
war\hello\E3C1ABB32E39A126A9194DB727F7742A.cache.html
war\hello\14A43CD7E24B0A0136C2B8B20D6DF3C0.cache.png
war\hello\548CDF11D6FE9011F3447CA200D7FB7F.cache.png
war\hello\9DA92932034707C17CFF15F95086D53F.cache.png
war\hello\A7CD51F9E5A7DED5F85AD1D82BA67A8A.cache.png
war\hello\B8517E9C2E38AA39AB7C0051564224D3.cache.png
war\hello\clear.cache.gif
war\hello\hello.nocache.js
war\hello\hosted.html
war\Hello.html
```

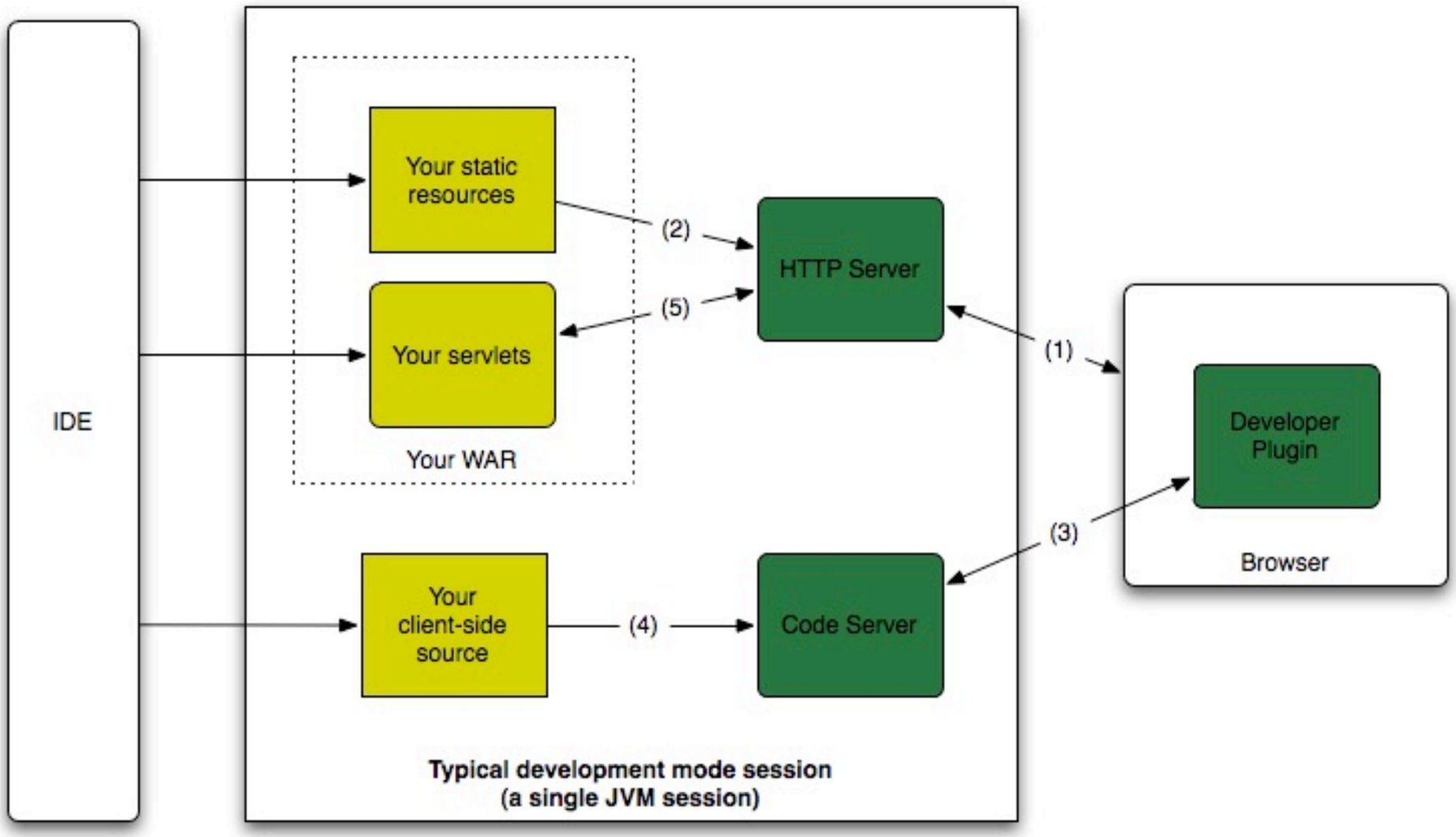
Your 5 permutations of compiled JS

After running the GWT compiler your war directory should look something like this:

```
C:\gwt-2.0.0\samples\Hello>\bin\find war
war
war\hello
war\hello\18EEC2DA45CB5F0C2050E2539AE61FCE.cache.html
war\hello\813B962DC4C22396EA14405DDEF020EE.cache.html
war\hello\86DA1DCEF4F40731BE71E7978CD4776A.cache.html
war\hello\A37FC20FF4D8F11605B2C4C53AF20B6F.cache.html
war\hello\E3C1ABB32E39A126A9194DB727F7742A.cache.html
war\hello\14A43CD7E24B0A0136C2B8B20D6DF3C0.cache.png
war\hello\548CDF11D6FE9011F3447CA200D7FB7F.cache.png
war\hello\9DA92932034707C17CFF15F95086D53F.cache.png
war\hello\A7CD51F9E5A7DED5F85AD1D82BA67A8A.cache.png
war\hello\B8517E9C2E38AA39AB7C0051564224D3.cache.png
war\hello\clear.cache.gif
war\hello\hello.nocache.js
war\hello\hosted.html
war\Hello.html
```

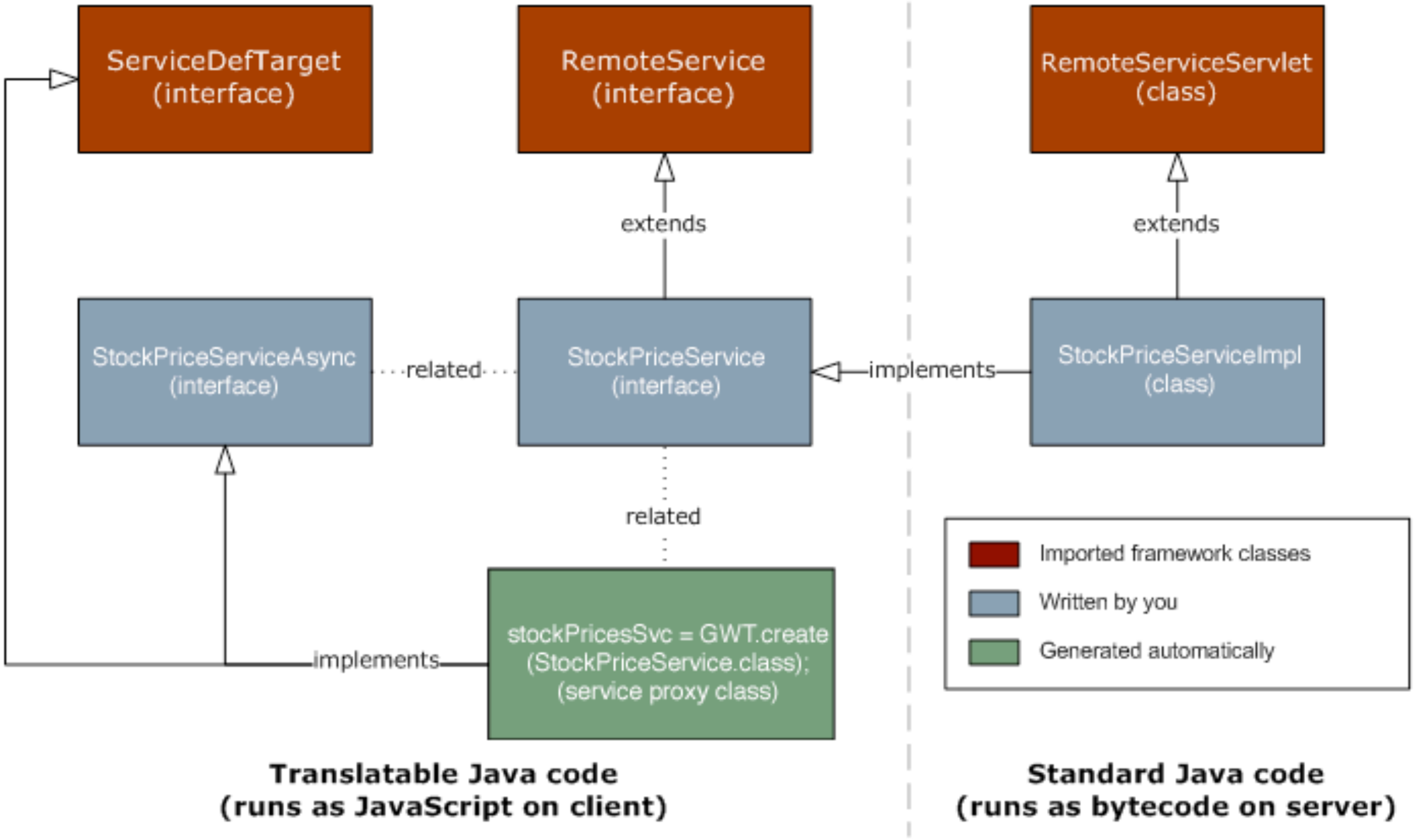
This is the non-cachable JS launcher (select the permutation).

GWT magic 2
Development mode

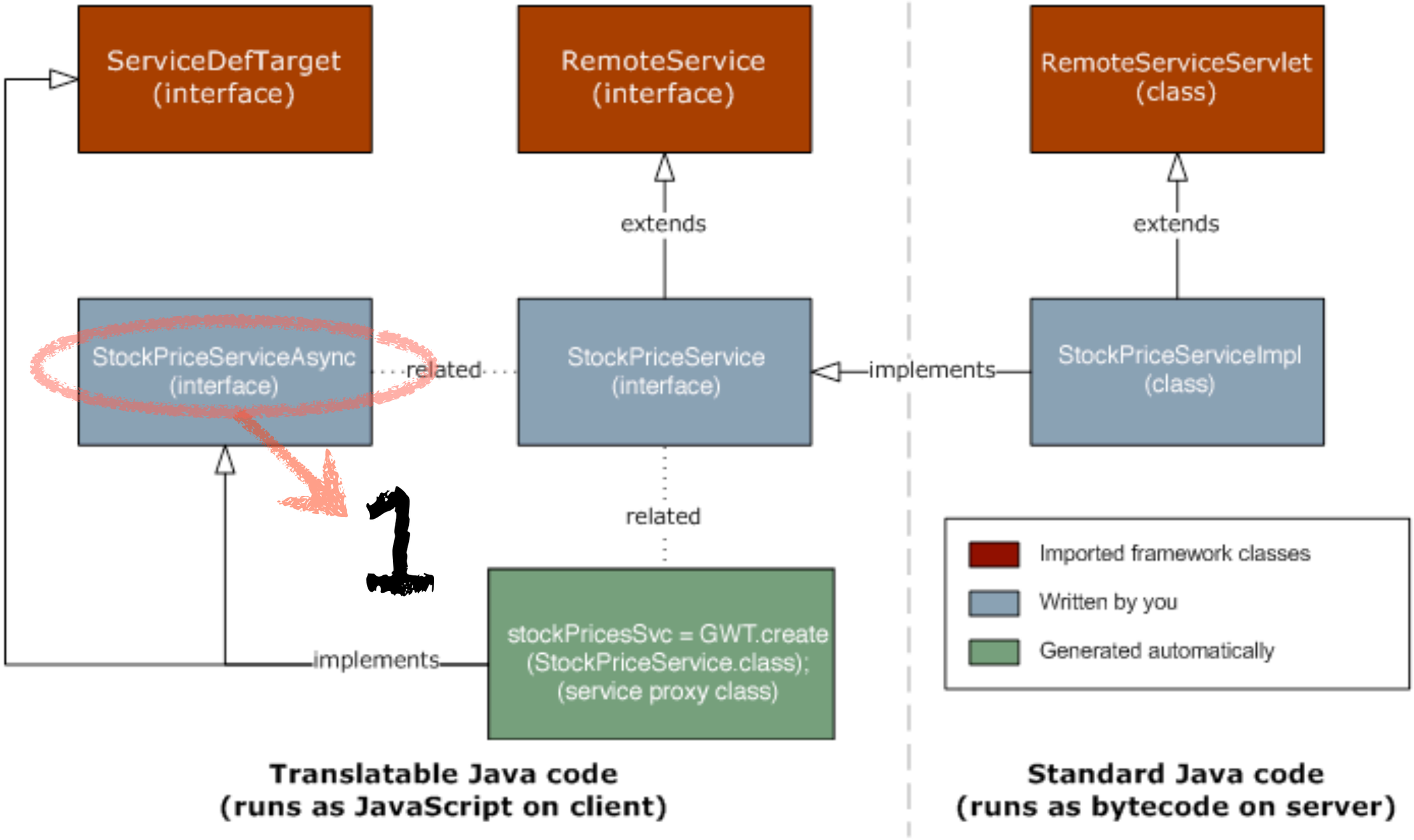


Devmode main advantage is client code hot-replace.
You need just a refresh on the browser.

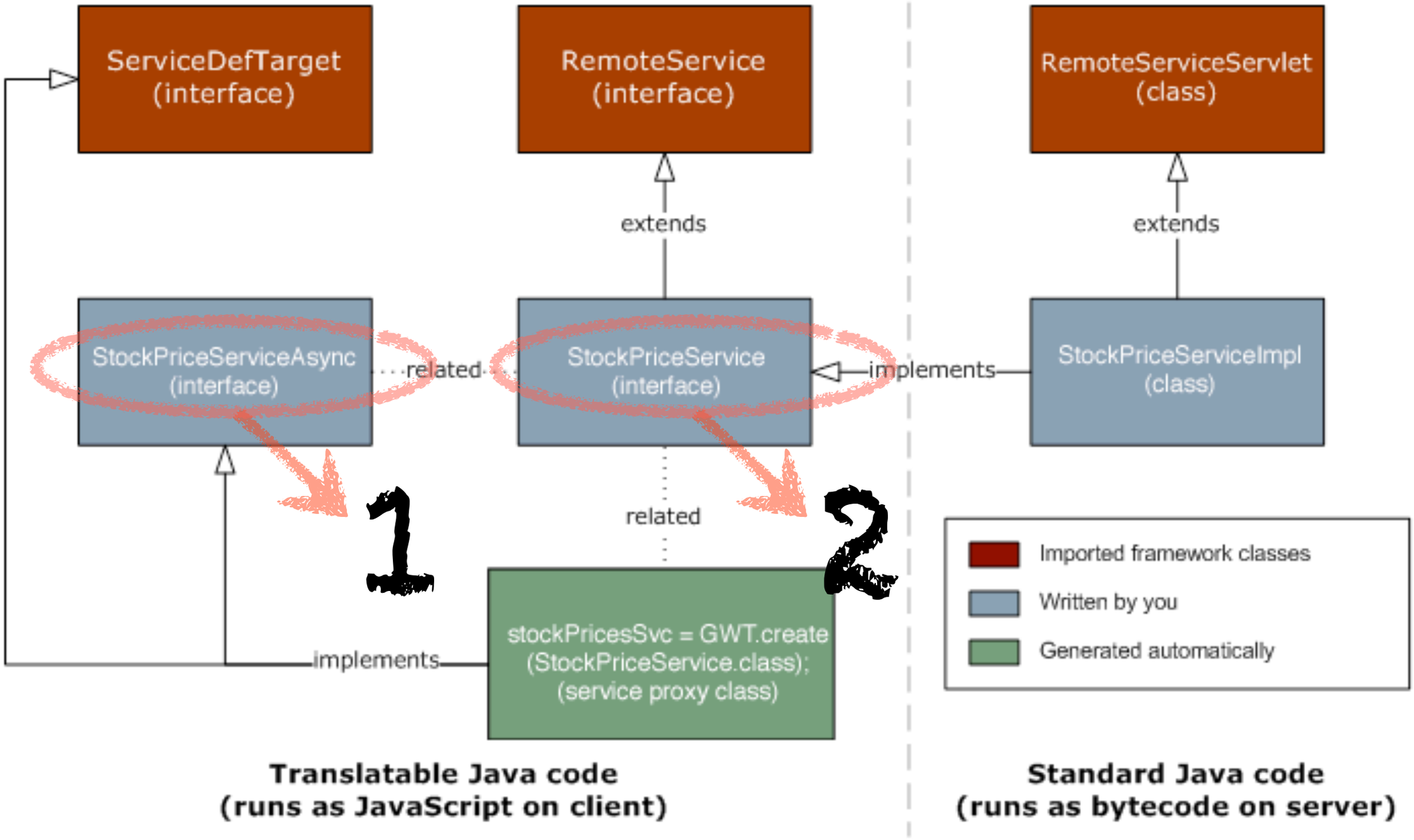
GWT magic 3
RPC services



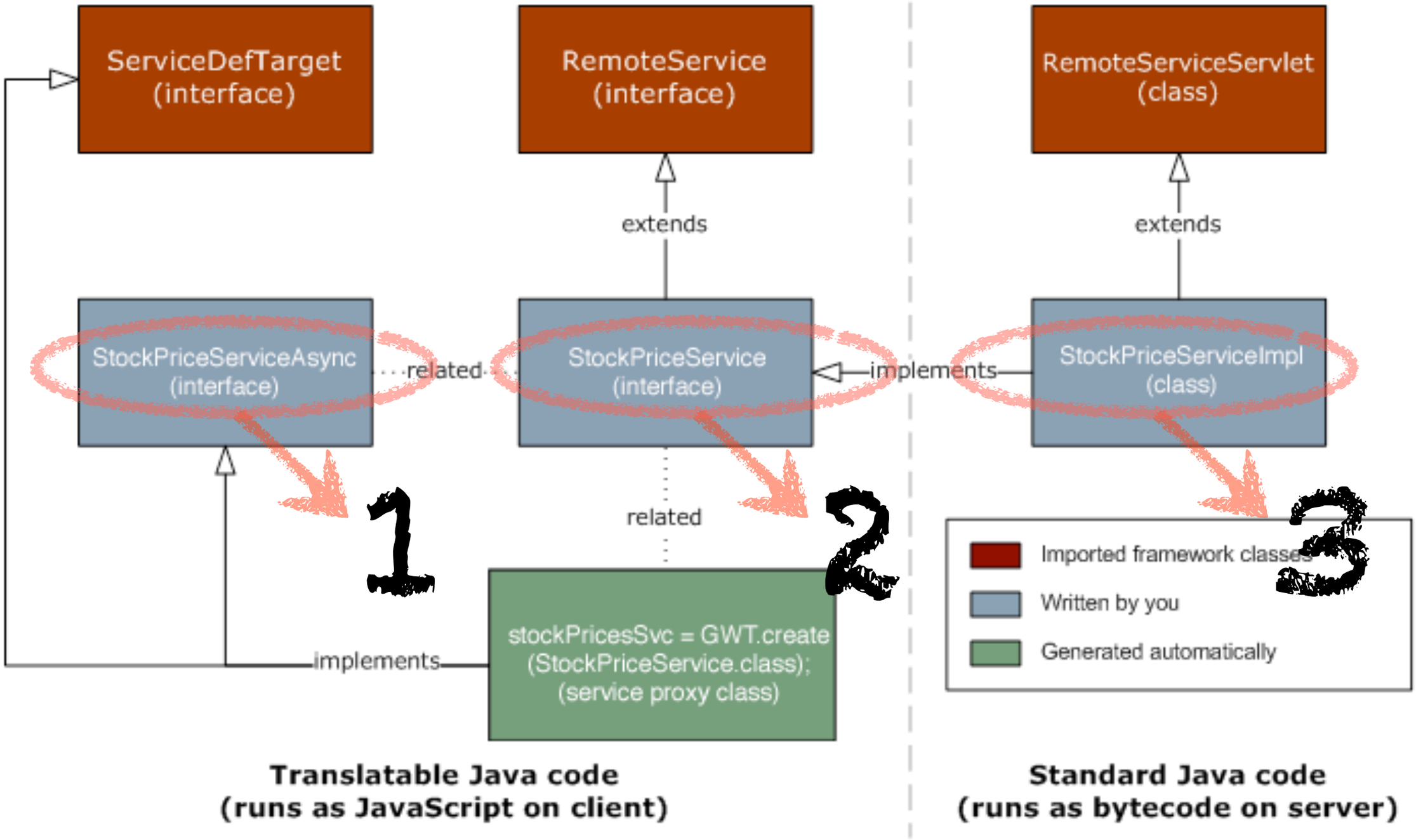
GWT magic 3
RPC services



GWT magic 3
RPC services



GWT magic 3
RPC services

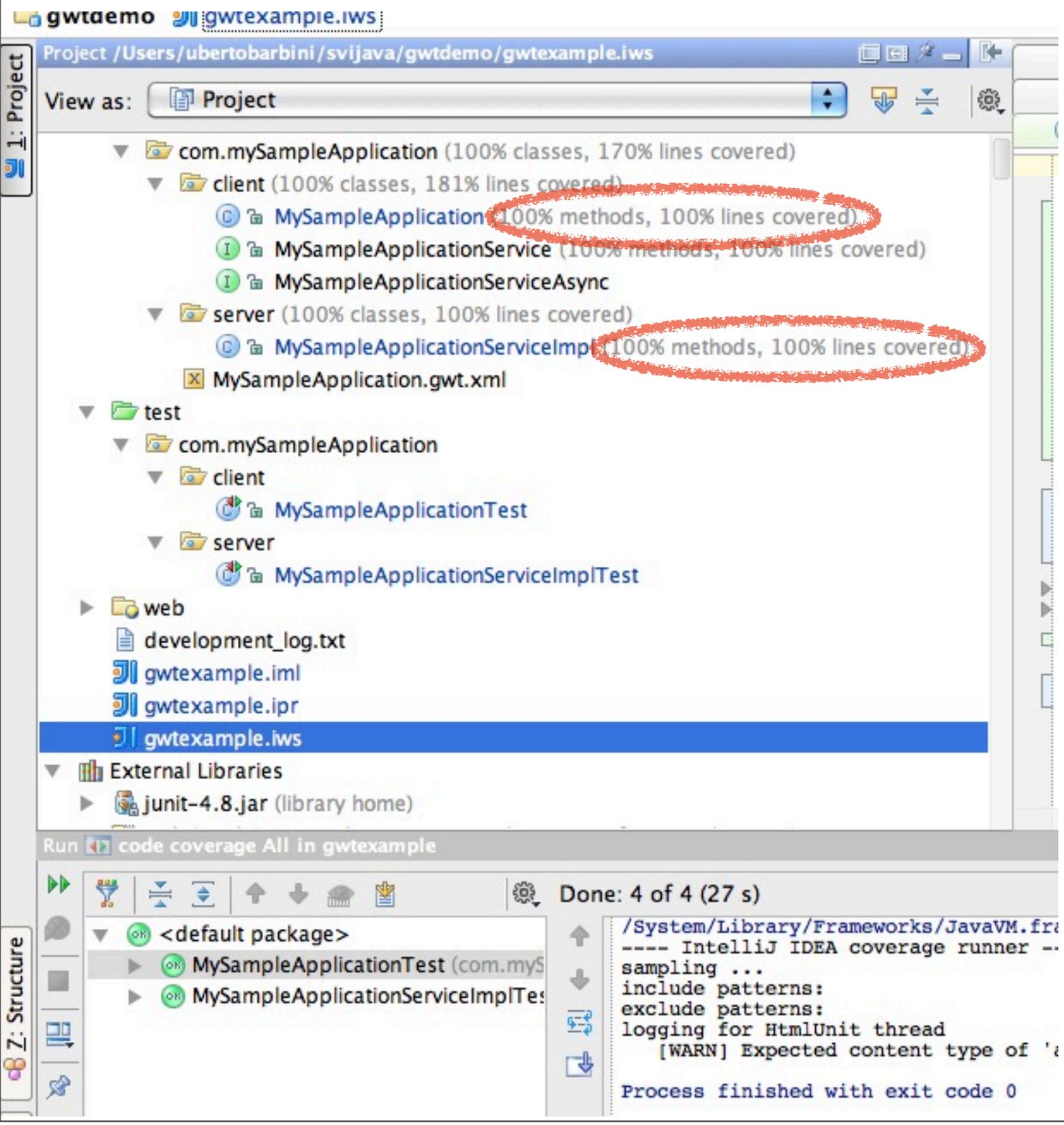


Enough talk, let's code!

- example of Hello in devmode

- example in IntelliJ of BugList Hello

Example HelloWorld



Starting with the gwt helloworld example we can write test to cover everything.

Let's see how...

```
package com.mySampleApplication.client;

import <...>

public class MySampleApplication implements
EntryPoint {

    public Button clickMeButton;
    public Label messageLabel;
    public String message = "Hello, World!";

    public void onModuleLoad() {
        clickMeButton = new Button("Click me");
        messageLabel = new Label();

        clickMeButton.addClickHandler
(createClickHandler(messageLabel, message));


        RootPanel.get("slot1").add(clickMeButton);
        RootPanel.get("slot2").add(messageLabel);
    }
```

```
    private ClickHandler createClickHandler(final
Label label, final String msg) {
        return new ClickHandler() {

            public void onClick(ClickEvent event) {
                if (label.getText().equals("")) {

MySampleApplicationService.App.getInstance
().getMessage(msg, new MyAsyncCallback(label));
                }
                else {
                  label.setText("");
                }
            }
        };
    }
```

```
    private static class MyAsyncCallback implements
AsyncCallback<String> {
        private Label label;
        public MyAsyncCallback(Label label) {
          this.label = label;
        }

        public void onSuccess(String result) {
          label.getElement().setInnerHTML(result);
        }

        public void onFailure(Throwable throwable) {
          label.setText("Failed to receive answer
from server!");
        }
    }
}
```

```
package com.mySampleApplication.client;

import ...

public class MySampleApplicationTest extends GWTTestCase
{
    @Override
    protected void gwtSetUp() throws Exception {
    }    ...//setup stuff

    @Override
    public String getModuleName() {
... } // return .gwt.xml name

    private void asyncTestValidation(Timer timer) {
        delayTestFinish(500);
        timer.schedule(100);
    } // to validate async test
```

```
    public void testUpdateLabelFromServerAtClick()
throws Exception {
        assertEquals(1, RootPanel.get
("slot1").getWidgetCount());

        app.clickMeButton.click();

        asyncTestValidation(new Timer() {
            public void run() {
                assertEquals("Client said: \"Hello,
World!\"Server answered: \"Hi!\"",
app.messageLabel.getText());
                finishTest();
            }
        });
    }
```

```
    public void testNotifyServerError() throws Exception
{
        final MySampleApplication wrongApp = new
MySampleApplication();
        wrongApp.message = null; //to simulate an error
from server
        wrongApp.onModuleLoad();

        wrongApp.clickMeButton.click();

        asyncTestValidation(new Timer() {
            public void run() {
                assertEquals("Failed to receive answer
from server!", wrongApp.messageLabel.getText());
                finishTest();
            }
        });
    }
```

```
    public void testResetLabelOnSecondClick() throws
Exception {

        app.clickMeButton.click();

        asyncTestValidation(new Timer() {
            public void run() {
                app.clickMeButton.click();

                asyncTestValidation(new Timer() {
                    public void run() {

                        assertEquals("",
app.messageLabel.getText());
                        finishTest();
                    }
                });
            }
        });
    }
}
```

```
package com.mySampleApplication.client;

import ...

public class MySampleApplicationTest extends GWTTestCase
{
    @Override
    protected void gwtSetUp() throws Exception {
    }    ...//setup stuff

    @Override
    public String getModuleName() {
... } // return .gwt.xml name

    private void asyncTestValidation(Timer timer) {
        delayTestFinish(500);
        timer.schedule(100);
    } // to validate async test
```

```
    public void testUpdateLabelFromServerAtClick()
throws Exception {
        assertEquals(1, RootPanel.get
("slot1").getWidgetCount());

        app.clickMeButton.click();

        asyncTestValidation(new Timer() {
            public void run() {
                assertEquals("Client said: \"Hello,
World!\"Server answered: \"Hi!\"",
app.messageLabel.getText());
                finishTest();
            }
        });
    }
```

So what's wrong with this?

```
    public void testNotifyServerError() throws Exception
{
        final MySampleApplication wrongApp = new
MySampleApplication();
        wrongApp.message = null; //to simulate an error
from server
        wrongApp.onModuleLoad();

        wrongApp.clickMeButton.click();

        asyncTestValidation(new Timer() {
            public void run() {
                assertEquals("Failed to receive answer
from server!", wrongApp.messageLabel.getText());
                finishTest();
            }
        });
    }
```

```
    public void testResetLabelOnSecondClick() throws
Exception {

        app.clickMeButton.click();

        asyncTestValidation(new Timer() {
            public void run() {
                app.clickMeButton.click();

                asyncTestValidation(new Timer() {
                    public void run() {

                        assertEquals("",
app.messageLabel.getText());
                        finishTest();
                    }
                });
            }
        });
    }
}
```

```
package com.mySampleApplication.client;

import ...

public class MySampleApplicationTest extends GWTTestCase
{
    @Override
    protected void gwtSetUp() throws Exception {
    }    ...//setup stuff

    @Override
    public String getModuleName() {
... } // return .gwt.xml name

    private void asyncTestValidation(Timer timer) {
        delayTestFinish(500);
        timer.schedule(100);
    } // to validate async test
```

Slow! it recompiles in Javascript

```
    public void testUpdateLabelFromServerAtClick()
throws Exception {
        assertEquals(1, RootPanel.get
("slot1").getWidgetCount());

        app.clickMeButton.click();

        asyncTestValidation(new Timer() {
            public void run() {
                assertEquals("Client said: \"Hello,
World!\"Server answered: \"Hi!\"",
app.messageLabel.getText());
                finishTest();
            }
        });
    }
```

```
    public void testNotifyServerError() throws Exception
{
        final MySampleApplication wrongApp = new
MySampleApplication();
        wrongApp.message = null; //to simulate an error
from server
        wrongApp.onModuleLoad();
        wrongApp.clickMeButton.click();

        asyncTestValidation(new Timer() {
            public void run() {
                assertEquals("Failed to receive answer
from server!", wrongApp.messageLabel.getText());
                finishTest();
            }
        });
    }
```

```
    public void testResetLabelOnSecondClick() throws
Exception {

        app.clickMeButton.click();

        asyncTestValidation(new Timer() {
            public void run() {
                app.clickMeButton.click();

                asyncTestValidation(new Timer() {
                    public void run() {

                        assertEquals("",
app.messageLabel.getText());
                        finishTest();
                    }
                });
            }
        });
    }
}
```

So what's wrong with this?

```
package com.mySampleApplication.client;

import ...

public class MySampleApplicationTest extends GWTTestCase
{
    @Override
    protected void gwtSetUp() throws Exception {
    }    ...//setup stuff

    @Override
    public String getModuleName() {
... } // return .gwt.xml name

    private void asyncTestValidation(Timer timer) {
        delayTestFinish(500);
        timer.schedule(100);
    } // to validate async test
```

Slow! it recompiles in Javascript

```
    public void testUpdateLabelFromServerAtClick()
throws Exception {
        assertEquals(1, RootPanel.get
("slot1").getWidgetCount());
        app.clickMeButton.click();

        asyncTestValidation(new Timer() {
            public void run() {
                assertEquals("Client said: \"Hello,
World!\"Server answered: \"Hi!\"",
app.messageLabel.getText());
                finishTest();
            }
        });
    }
```

Fragile, it use whole app

```
    public void testNotifyServerError() throws Exception
{
        final MySampleApplication wrongApp = new
MySampleApplication();
        wrongApp.message = null; //to simulate an error
from server
        wrongApp.onModuleLoad();
        wrongApp.clickMeButton.click();

        asyncTestValidation(new Timer() {
            public void run() {
                assertEquals("Failed to receive answer
from server!", wrongApp.messageLabel.getText());
                finishTest();
            }
        });
    }
```

```
    public void testResetLabelOnSecondClick() throws
Exception {
        app.clickMeButton.click();

        asyncTestValidation(new Timer() {
            public void run() {
                app.clickMeButton.click();

                asyncTestValidation(new Timer() {
                    public void run() {

                        assertEquals("",
app.messageLabel.getText());
                        finishTest();
                    }
                });
            }
        });
    }
}
```

So what's wrong with this?

```
package com.mySampleApplication.client;

import ...

public class MySampleApplicationTest extends GWTTestCase
{
    @Override
    protected void gwtSetUp() throws Exception {
    }    ...//setup stuff

    @Override
    public String getModuleName() {
... } // return .gwt.xml name

    private void asyncTestValidation(Timer timer) {
        delayTestFinish(500);
        timer.schedule(100);
    } // to validate async test
```

Slow! it recompiles in Javascript

```
    public void testUpdateLabelFromServerAtClick()
throws Exception {
        assertEquals(1, RootPanel.get
("slot1").getWidgetCount());
        app.clickMeButton.click();

        asyncTestValidation(new Timer() {
            public void run() {
                assertEquals("Client said: \"Hello,
World!\"Server answered: \"Hi!\"",
app.messageLabel.getText());
                finishTest();
            }
        });
    }
```

Fragile, it use whole app

Slow! Async tests take time

```
    public void testNotifyServerError() throws Exception
{
        final MySampleApplication wrongApp = new
MySampleApplication();
        wrongApp.message = null; //to simulate an error
from server
        wrongApp.onModuleLoad();
        wrongApp.clickMeButton.click();

        asyncTestValidation(new Timer() {
            public void run() {
                assertEquals("Failed to receive answer
from server!", wrongApp.messageLabel.getText());
                finishTest();
            }
        });
    }
```

```
    public void testResetLabelOnSecondClick() throws
Exception {
        app.clickMeButton.click();

        asyncTestValidation(new Timer() {
            public void run() {
                app.clickMeButton.click();

                asyncTestValidation(new Timer() {
                    public void run() {
                        assertEquals("",
app.messageLabel.getText());
                        finishTest();
                    }
                });
            }
        });
    }
}
```

So what's wrong with this?

```
package com.mySampleApplication.client;

import ...

public class MySampleApplicationTest extends GWTTestCase
{
    @Override
    protected void gwtSetUp() throws Exception {
    }    ...//setup stuff

    @Override
    public String getModuleName() {
... } // return .gwt.xml name

    private void asyncTestValidation(Timer timer) {
        delayTestFinish(500);
        timer.schedule(100);
    } // to validate async test
```

Slow! it recompiles in Javascript

It tests state not interactions

```
    public void testNotifyServerError() throws Exception
{
        final MySampleApplication wrongApp = new
MySampleApplication();
        wrongApp.message = null; //to simulate an error
from server
        wrongApp.onModuleLoad();
        wrongApp.clickMeButton.click();

        asyncTestValidation(new Timer() {
            public void run() {
                assertEquals("Failed to receive answer
from server!", wrongApp.messageLabel.getText());
                finishTest();
            }
        });
    }
```

Fragile!

```
    public void testUpdateLabelFromServerAtClick()
throws Exception {
        assertEquals(1, RootPanel.get
("slot1").getWidgetCount());
        app.clickMeButton.click();

        asyncTestValidation(new Timer() {
            public void run() {
                assertEquals("Client said: \"Hello,
World!\"Server answered: \"Hi!\"",
app.messageLabel.getText());
                finishTest();
            }
        });
    }
```

Fragile, it use whole app

Slow! Async tests take time

```
    public void testResetLabelOnSecondClick() throws
Exception {
        app.clickMeButton.click();

        asyncTestValidation(new Timer() {
            public void run() {
                app.clickMeButton.click();

                asyncTestValidation(new Timer() {
                    public void run() {
                        assertEquals("",
app.messageLabel.getText());
                        finishTest();
                    }
                });
            }
        });
    }
}
```

So what's wrong with this?

HelloWorld with Tests

What we learned?

HelloWorld with Tests

What we learned?

Gwt HelloWorld can easily be tested with 100% coverage using GWTTestCase.

HelloWorld with Tests

What we learned?

Gwt HelloWorld can easily be tested with 100% coverage using GWTTestCase.

But we need to do better than that!

HelloWorld with Tests

What we learned?

Gwt HelloWorld can easily be tested with 100% coverage using GWTTestCase.

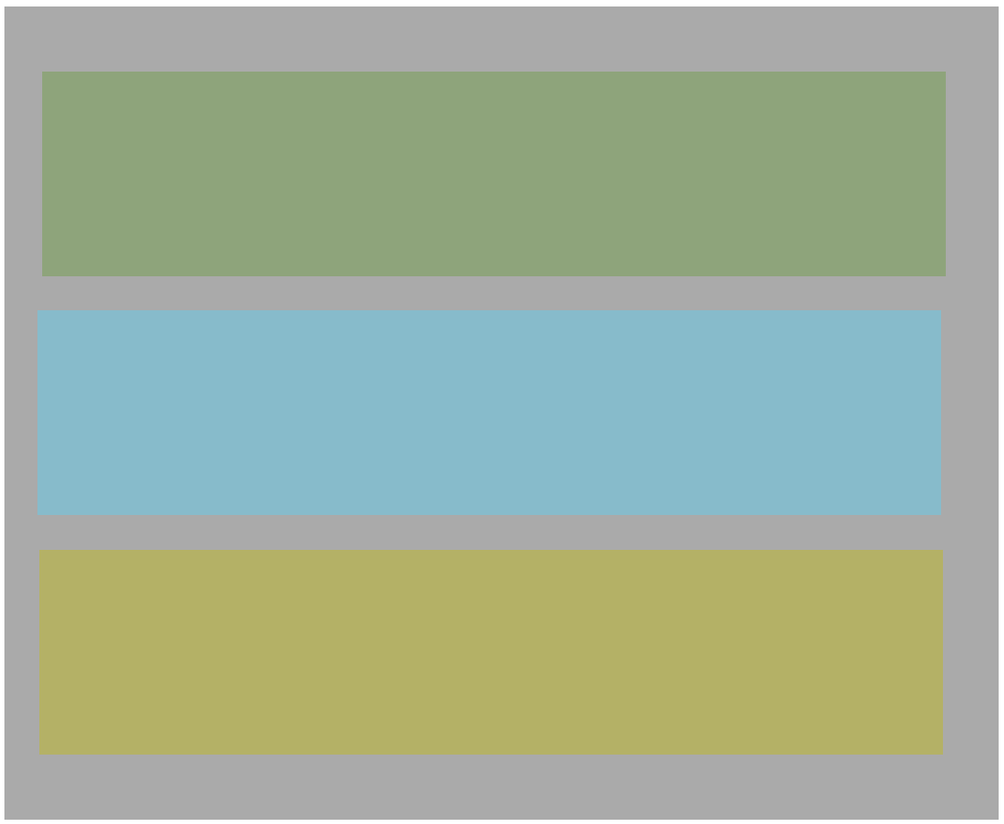
But we need to do better than that!

We need to keep GWTTestCase use at minimum and test objects separately.


Ok, now I know
I can test all my
Gwt classes...
But last time I
checked TDD meant
Test Driven **Design**

Introducing MVP

Remove Presenter and View from Application



Introducing MVP

Remove Presenter and View from Application

Introducing MVP

Remove Presenter and View from Application

Introducing MVP

Remove Presenter and View from Application

Introducing MVP

Remove Presenter and View from Application

Introducing MVP

Remove Presenter and View from Application

Application

Presenter

Introducing MVP

Remove Presenter and View from Application

Introducing MVP

Remove Presenter and View from Application

Application

Services

Presenter

Introducing MVP

Remove Presenter and View from Application

Application

Services

Presenter

View

```
public class MySampleApplication implements
EntryPoint {
    public void onModuleLoad() {
        Presenter presenter = getPresenterFromUrl();
        presenter.activate();
    }

    private Presenter getPresenterFromUrl()
    ... //this will read from history
   }
}
```

```
public class BugListPresenter extends Presenter {
...
    public BugListPresenter(BugListView view,
MySampleApplicationServiceAsync messageService) {
        super(view);
        this.view = view;
        this.messageService = messageService;
        asyncCallback = new MyAsyncCallback(view);
        view.addClickMeHandler(createClickHandler
());
    }

    @Override
    protected void onShow() {
        view.setTitle("Active bugs");
    }
```

```
    private ClickHandler createClickHandler() {
        return new ClickHandler() {

            public void onClick(ClickEvent event) {
                if (view.getMessageText().isEmpty())
{
                    messageService.getMessage
(message, asyncCallback);
                } else {
                    view.setMessageText("");
```

```
                }
            }
        };
    }

    private static class MyAsyncCallback implements
AsyncCallback<String> {

        private NotificationView view;

        public MyAsyncCallback(NotificationView
view) {
            this.view = view;
        }

        public void onSuccess(String result) {
            view.setMessageText(result);
        }

        public void onFailure(Throwable throwable) {
            view.setMessageText("Failed to receive
answer from server!");
        }
    }

}
```

Result of ajax call on the label

Application start

Ajax call on click

```
public class BugListViewFlexTable extends ViewAbstractRoot implements BugListView {

    public Button clickMeButton = new Button("Click me");;

    public Label messageLabel = new Label();

    public void setTitle(String title) {

        RootPanel.get("slot1").add(clickMeButton);
        RootPanel.get("slot2").add(messageLabel);

        RootPanel.get("title-label").getElement().setInnerText(title);
    }


    @Override
    public void addClickMeHandler(ClickHandler clickHandler) {
        clickMeButton.addClickHandler(clickHandler);
    }

    @Override
    public void setMessageText(String message) {
        messageLabel.getElement().setInnerHTML(message);  //for html
    }

    @Override
    public String getMessageText() {
        return messageLabel.getText();
    }
}
```

View very simple
with clear interface

```
ublic class BugListViewTest extends GwtTestWithApp {


    private BugListView bugListView;


    @Override
    protected void gwtSetUp() throws Exception {
        super.gwtSetUp();

        bugListView = new BugListViewFlexTable();
    }

    public void testCreateAnEmptyGridOnActivation() throws Exception {
        bugListView.show();
        assertEquals(1, RootPanel.get("wrapper").getWidgetCount());
        assertEquals(0, bugListView.getRowCount());
        assertEquals("Active bugs", RootPanel.get("title-label").getElement().getInnerText());


    }

    public void testDisplayAndReadTheMessage() throws Exception {
        bugListView.show();
        assertEquals("<div class=\"gwt-Label\"></div>", RootPanel.get("slot2").getElement().getInnerHTML());

        bugListView.setMessageText("My message");
        assertEquals("<div class=\"gwt-Label\"></div>", RootPanel.get("slot2").getElement().getInnerHTML());

        assertEquals("My message", bugListView.getMessageText());
    }

}
```

View stories

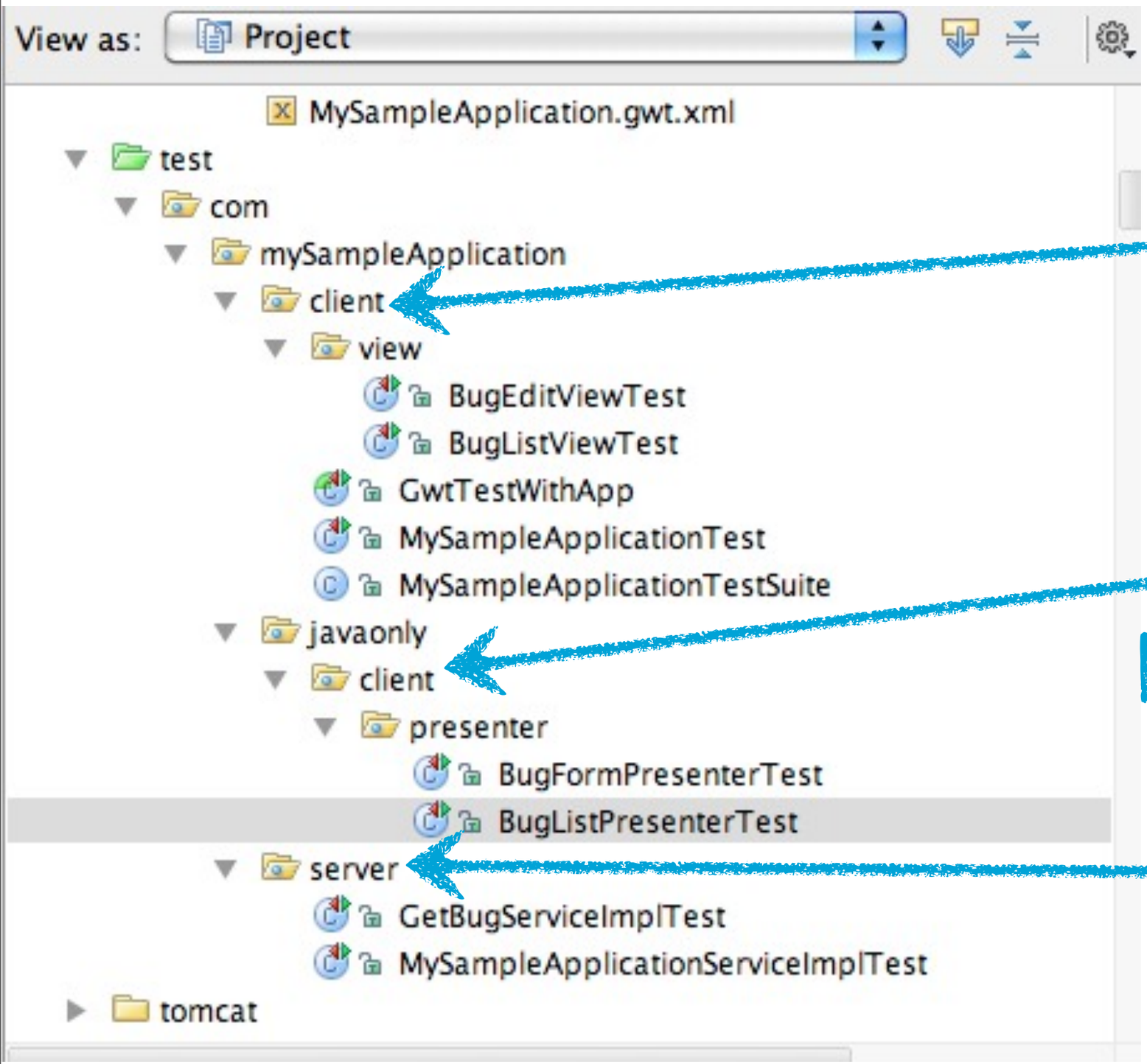
```
public class BugListPresenterTest {

    @Before
    public void setUp() throws Exception {
        clickAnswer = new ClickAnswer();
        doAnswer(clickAnswer).when
(view).addClickMeHandler(any(ClickHandler.class));

        doAnswer(new MessageAnswer()).when
(getMessageService).getMessage(anyString(), any
(AsyncCallback.class));
        when(view.getMessageText()).thenReturn("");

        pres = new BugListPresenter(view,
getMessageService);
    }

    @After
    public void dropDown() throws Exception {
        Mockito.verifyNoMoreInteractions(view);
        Mockito.verifyNoMoreInteractions
(getMessageService);

    }

    @Test
    public void shouldShowTheMessageOnClick() throws
Exception {
        pres.activate();

        verifyViewInvocations();
        clickAnswer.clickHandler.onClick(null);

        verify(getMessageService).getMessage(anyString
(), any(AsyncCallback.class));
        verify(view).getMessageText();

        verify(view).setMessageText("MessageAnswer
invoked with: \"Hello, World!\"");
    }

    @Test
    public void shouldShowTheErrorWhenServerFails()
throws Exception {
        doAnswer(new MessageErrorAnswer()).when
(getMessageService).getMessage(anyString(), any
(AsyncCallback.class));

        pres.activate();

        verifyViewInvocations();
        clickAnswer.clickHandler.onClick(null);

        verify(getMessageService).getMessage(anyString
(), any(AsyncCallback.class));
        verify(view).getMessageText();

        verify(view).setMessageText("Failed to receive
answer from server!");
    }

    @Test
    public void shouldResetTheMessageTheSecondClick()
throws Exception {
        when(view.getMessageText()).thenReturn("not
void");

        pres.activate();

        verifyViewInvocations();

        clickAnswer.clickHandler.onClick(null);

        verify(view).getMessageText();
        verify(view).setMessageText("");
    }
```

```
public class BugListPresenterTest {

    @Before
    public void setUp() throws Exception {
        clickAnswer = new ClickAnswer();
        doAnswer(clickAnswer).when
(view).addClickMeHandler(any(ClickHandler.class));

        doAnswer(new MessageAnswer()).when
(getMessageService).getMessage(anyString(), any
(AsyncCallback.class));
        when(view.getMessageText()).thenReturn("");

        pres = new BugListPresenter(view,
getMessageService);
    }

    @After
    public void dropDown() throws Exception {
        Mockito.verifyNoMoreInteractions(view);
        Mockito.verifyNoMoreInteractions
(getMessageService);

    }

    @Test
    public void shouldShowTheMessageOnClick() throws
Exception {
        pres.activate();

        verifyViewInvocations();
        clickAnswer.clickHandler.onClick(null);

        verify(getMessageService).getMessage(anyString
(), any(AsyncCallback.class));
        verify(view).getMessageText();

        verify(view).setMessageText("MessageAnswer
invoked with: \"Hello, World!\"");
    }
```

Presenter with logic using mocks for view and services

```
    @Test
    public void shouldShowTheErrorWhenServerFails()
throws Exception {
        doAnswer(new MessageErrorAnswer()).when
(getMessageService).getMessage(anyString(), any
(AsyncCallback.class));
        pres.activate();

        verifyViewInvocations();
        clickAnswer.clickHandler.onClick(null);

        verify(getMessageService).getMessage(anyString
(), any(AsyncCallback.class));
        verify(view).getMessageText();

        verify(view).setMessageText("Failed to receive
answer from server!");
    }


    @Test
    public void shouldResetTheMessageTheSecondClick()
throws Exception {
        when(view.getMessageText()).thenReturn("not
void");

        pres.activate();

        verifyViewInvocations();

        clickAnswer.clickHandler.onClick(null);

        verify(view).getMessageText();
        verify(view).setMessageText("");
    }
```

MVP Unit Tests



Client GWT Tests

Pure Java pseudo-client Tests

Pure Java standard Tests

Model View Presenter

Model View Presenter

- 100% test coverage.
  To let them drive your design.

Model View Presenter

- 100% test coverage.
  To let them drive your design.

- pseudo-client tests split.
  Test Presenters and Model in plain Java.

Model View Presenter

- 100% test coverage.
  To let them drive your design.
- pseudo-client tests split.
  Test Presenters and Model in plain Java.
- GwtTests only for View/Services.
  *Services have no status. Views just a very shallow one (MVP not MVC). You can also test the DOM**

Model View Presenter

- 100% test coverage.
  To let them drive your design.
- pseudo-client tests split.
  Test Presenters and Model in plain Java.
- GwtTests only for View/Services.
  *Services have no status. Views just a very shallow one (MVP not MVC). You can also test the DOM**
- asyncTests only for testing the Services.
  That's why Services are not in the View.

Model View Presenter

- 100% test coverage.
  To let them drive your design.

- pseudo-client tests split.
  Test Presenters and Model in plain Java.

- GwtTests only for View/Services.
  *Services have no status. Views just a very shallow one (MVP not MVC). You can also test the DOM**

- asyncTests only for testing the Services.
  That's why Services are not in the View.

**in lack of something better*

If you have to remember just a slide please remember this one!

- 100% test coverage.
  To let them drive your design.

- pseudo-client tests split.
  Test Presenters and Model in plain Java.

- GwtTests only for View/Services.
  *Services have no status. Views just a very shallow one (MVP not MVC). You can also test the DOM**

- asyncTests only for testing the Services.
  That's why Services are not in the View.

**in lack of something better*

Example BugTracker

Example BugTracker

- First the Model
  serverside TDD

Example BugTracker

- First the Model
  serverside TDD
- Then the View
  developer mode design + client TDD

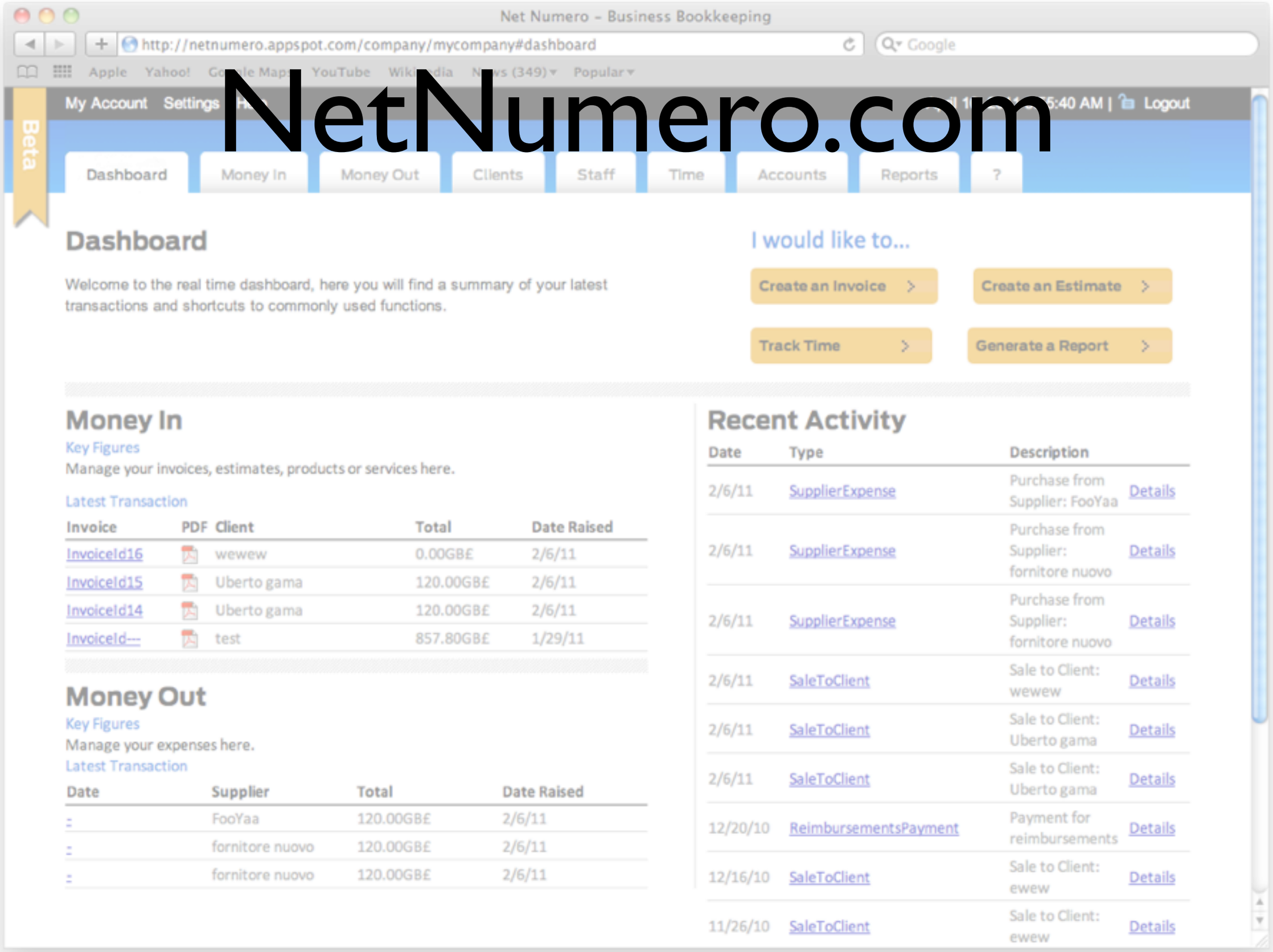
Example BugTracker

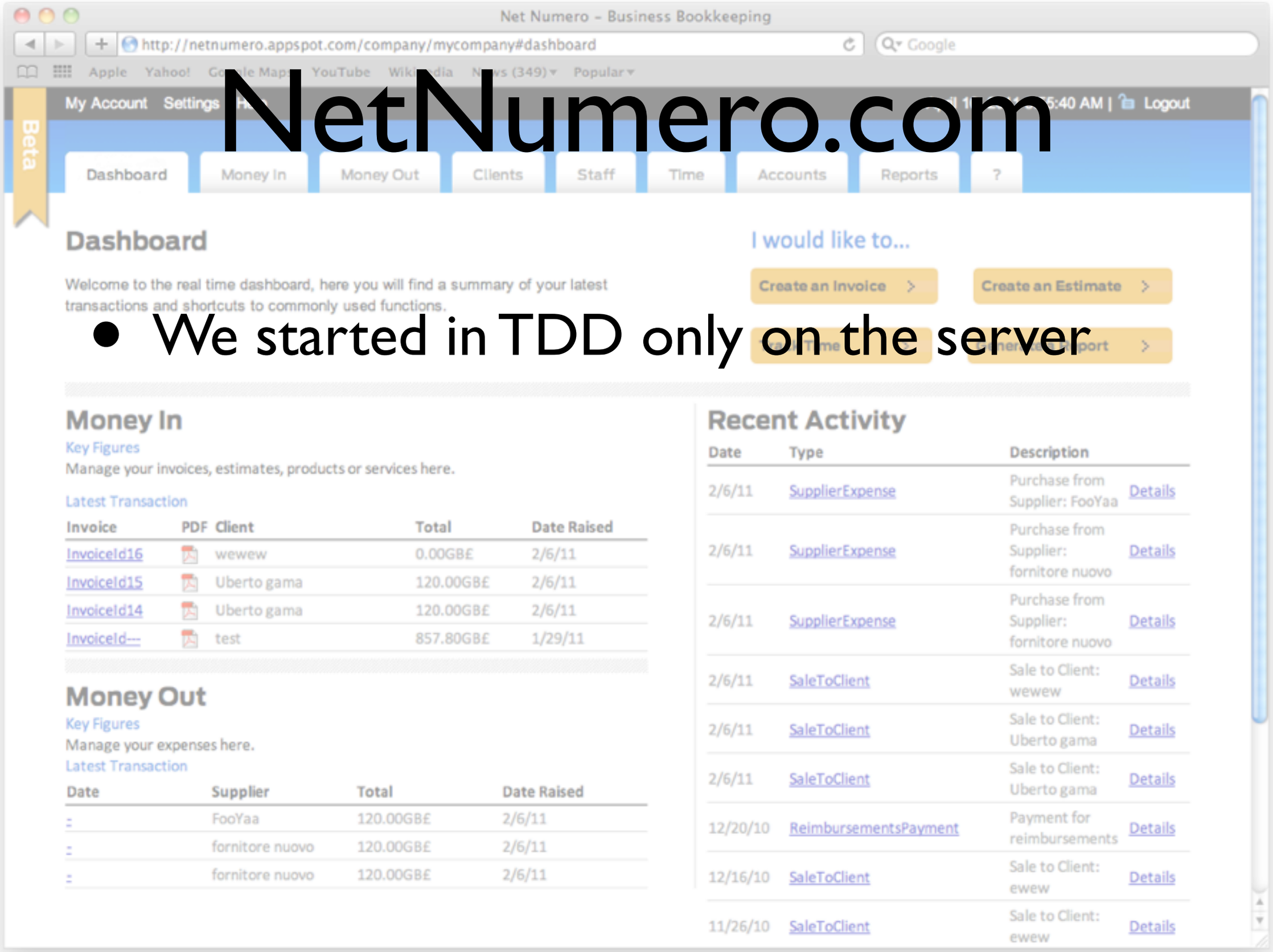
- First the Model
  serverside TDD
- Then the View
  developer mode design + client TDD
- Continue with Presenter
  pseudo-client TDD

Example BugTracker

- First the Model
  serverside TDD
- Then the View
  developer mode design + client TDD
- Continue with Presenter
  pseudo-client TDD
- Finish with the Plumbing
  Integration Tests

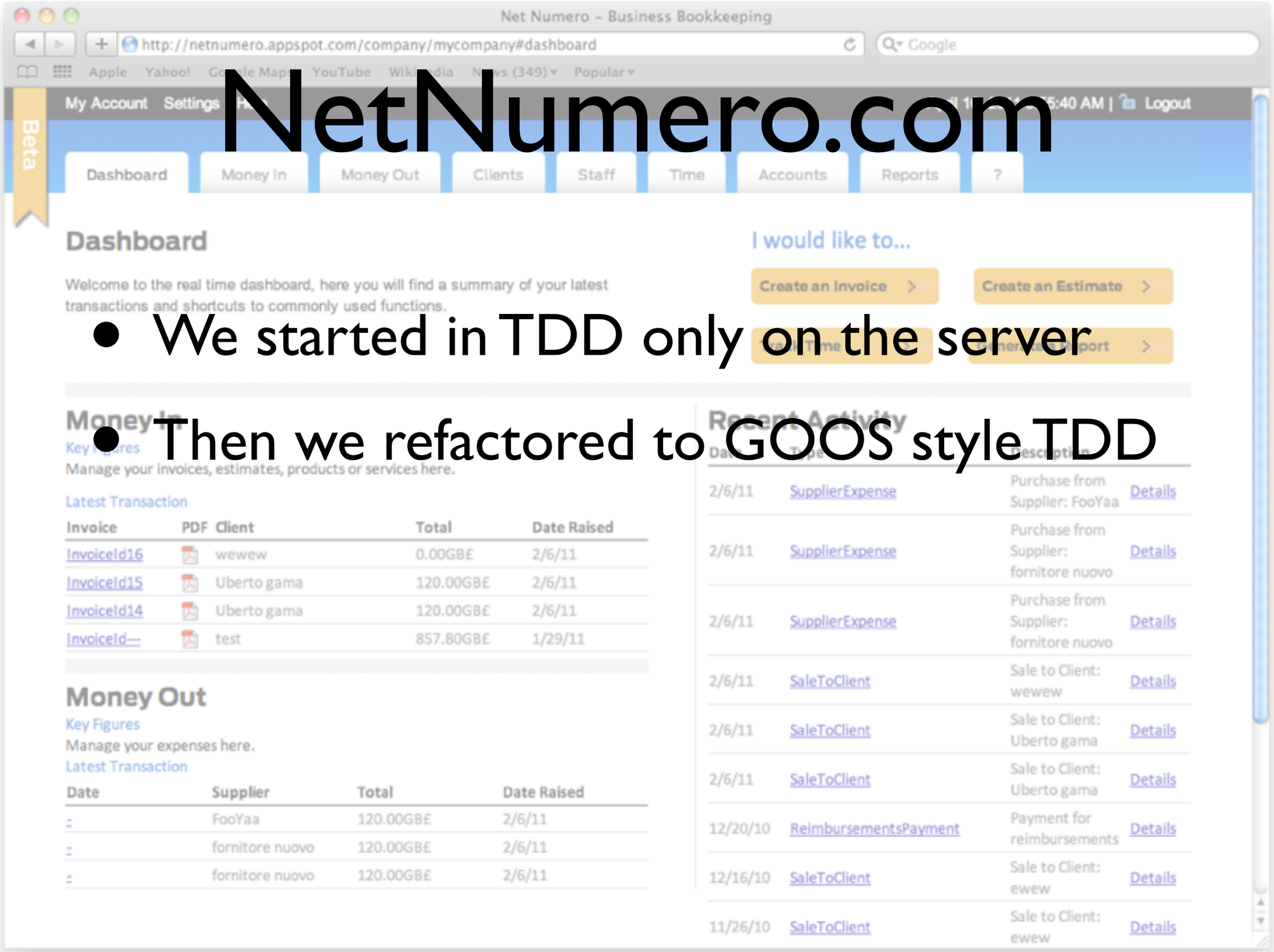
NetNumero.com





NetNumero.com

- We started in TDD only on the server

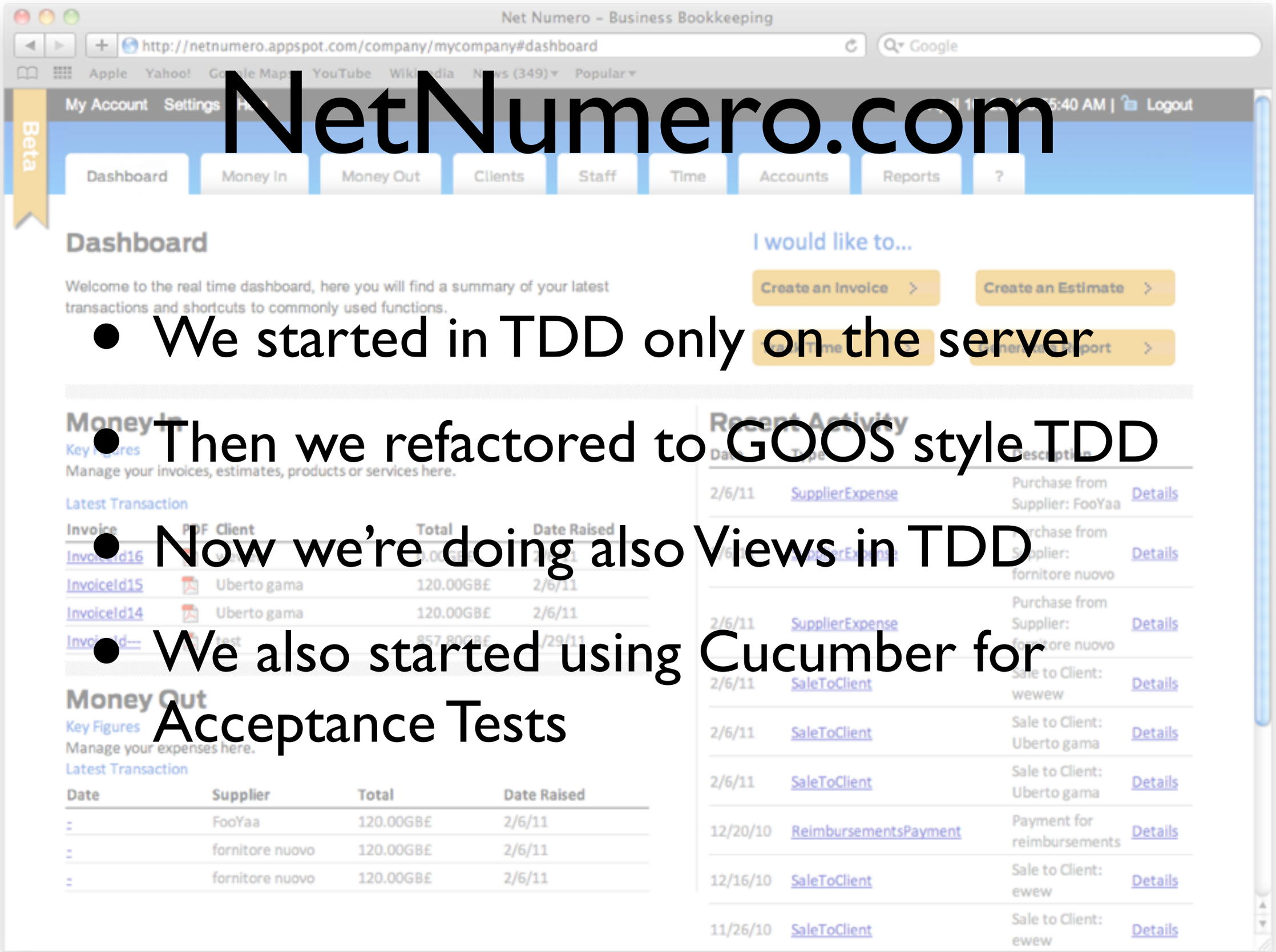


NetNumero.com

- We started in TDD only on the server
- Then we refactored to GOOS style TDD

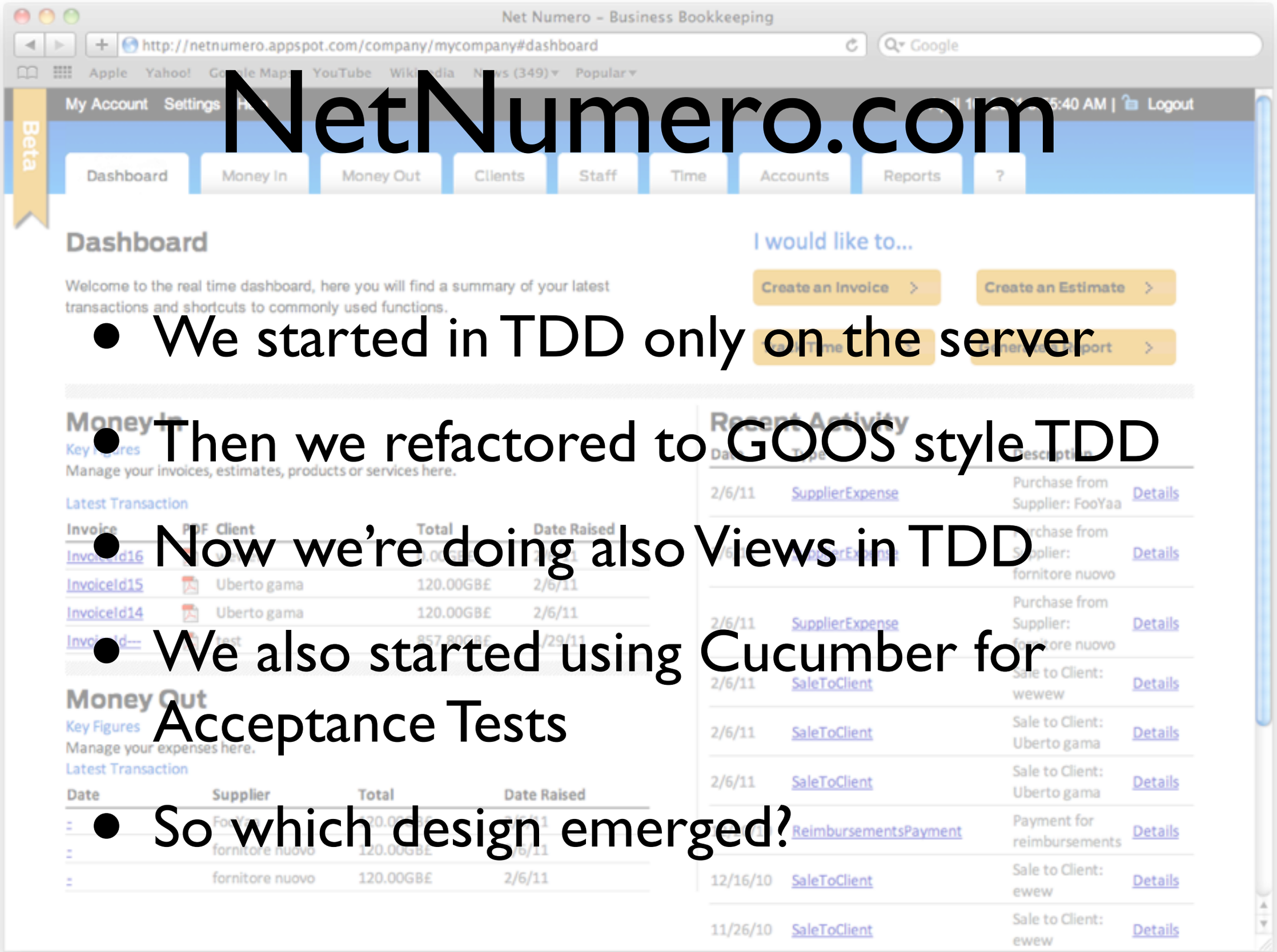
NetNumero.com

- We started in TDD only on the server
- Then we refactored to GOOS style TDD
- Now we're doing also Views in TDD



NetNumero.com

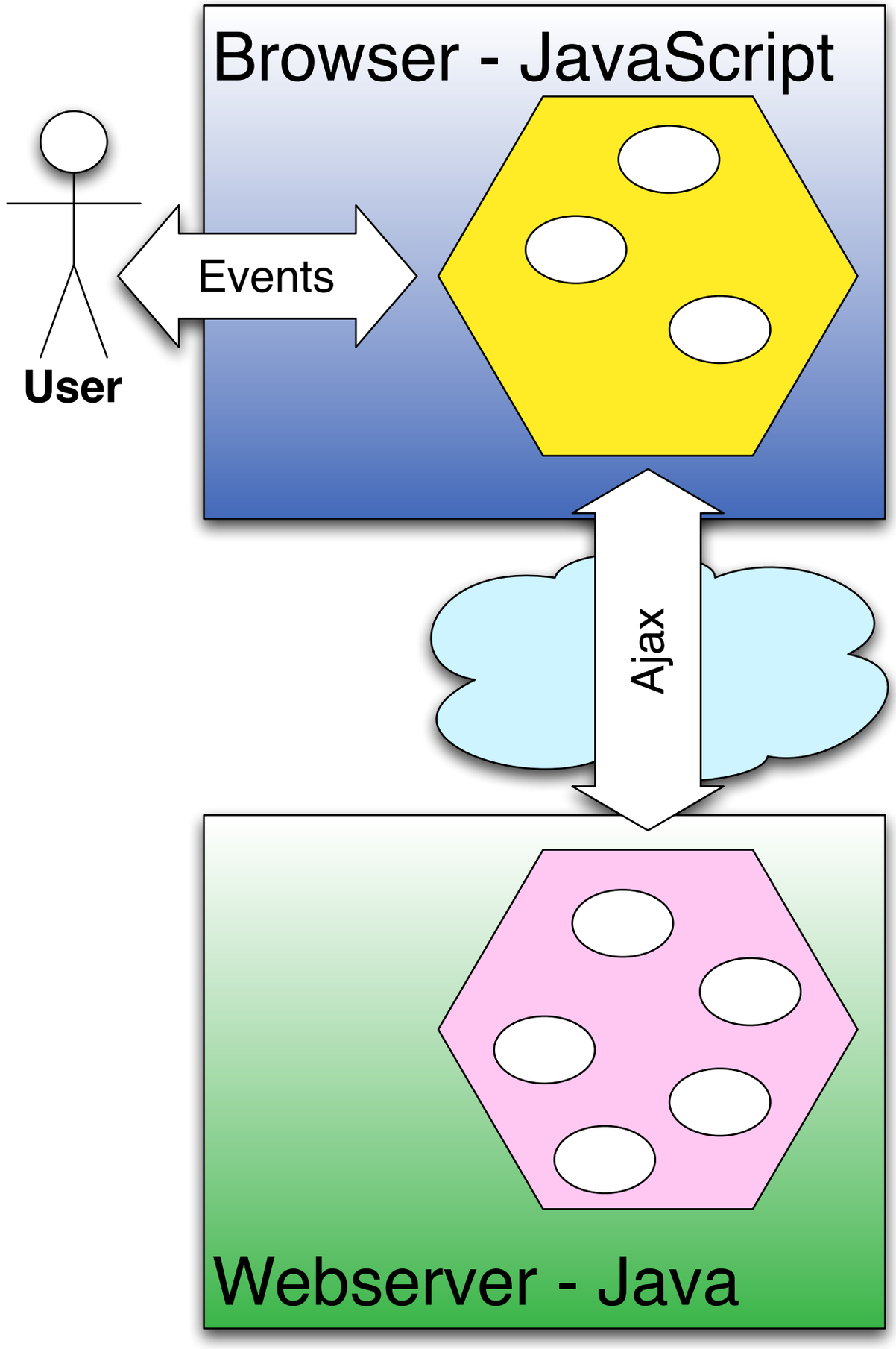
- We started in TDD only on the server
- Then we refactored to GOOS style TDD
- Now we're doing also Views in TDD
- We also started using Cucumber for Acceptance Tests



NetNumero.com

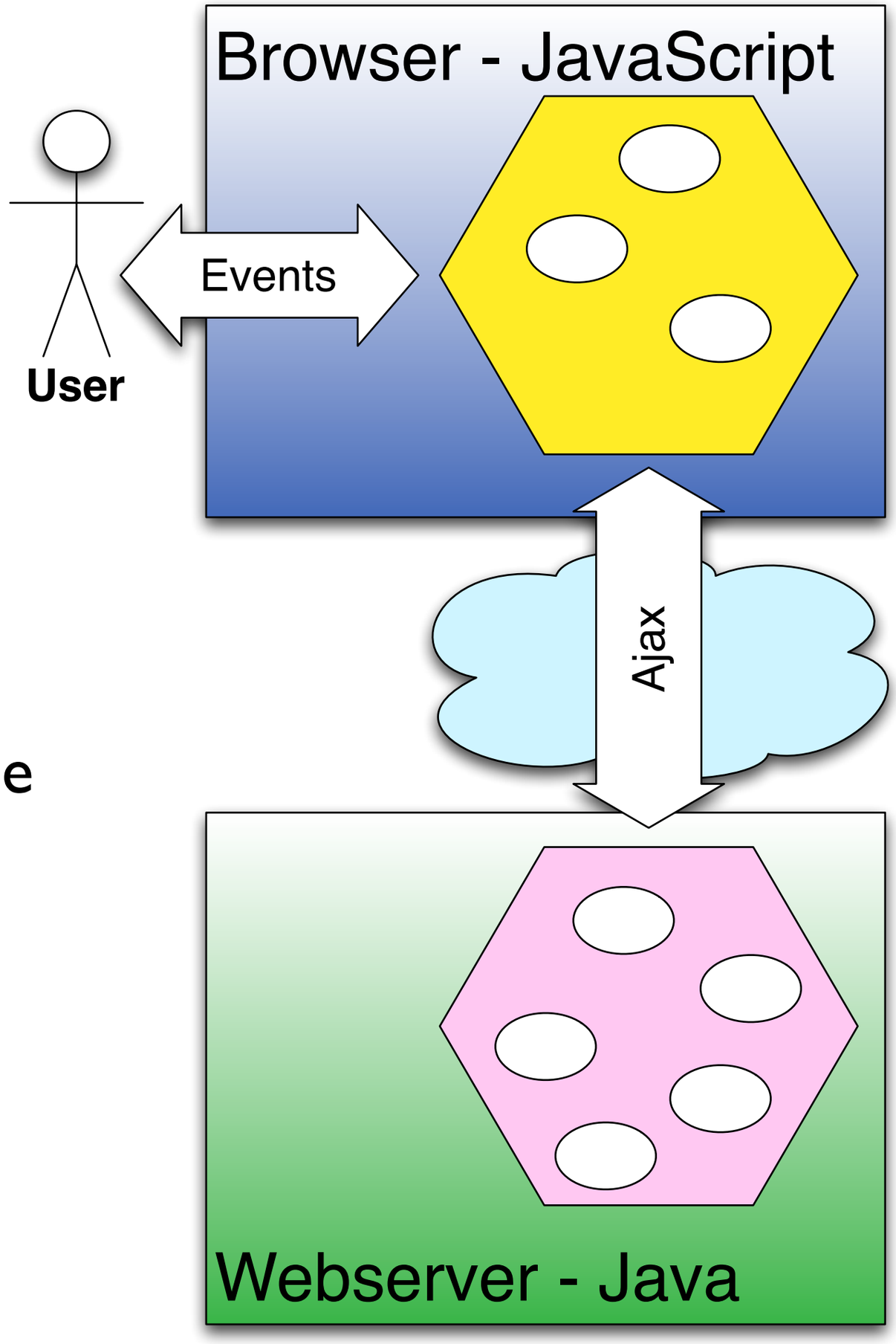
- We started in TDD only on the server
- Then we refactored to GOOS style TDD
- Now we're doing also Views in TDD
- We also started using Cucumber for Acceptance Tests
- So which design emerged?

Dual-Exagon Architecture

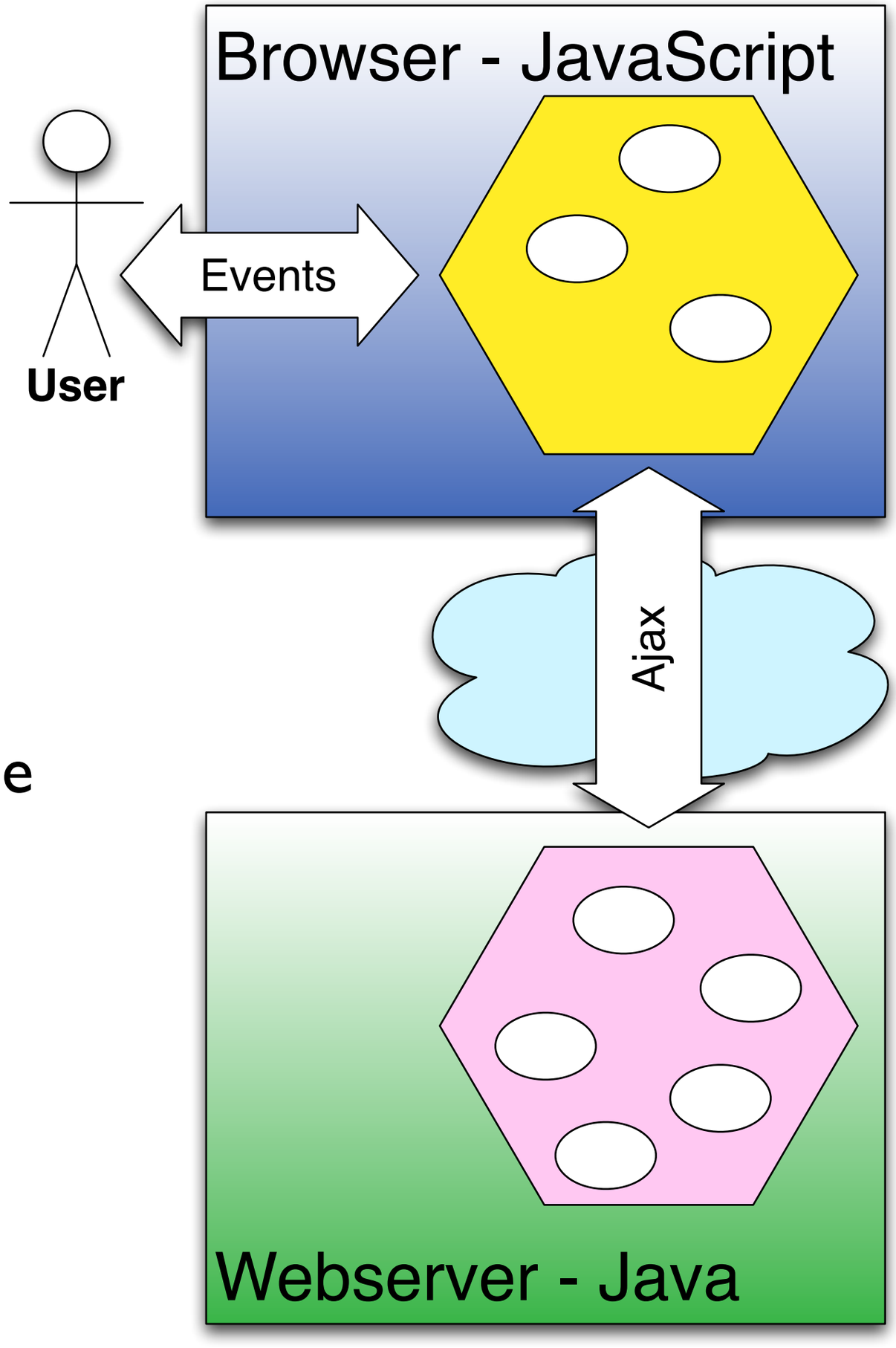


Dual-Exagon Architecture

- All client code is on a single html page.

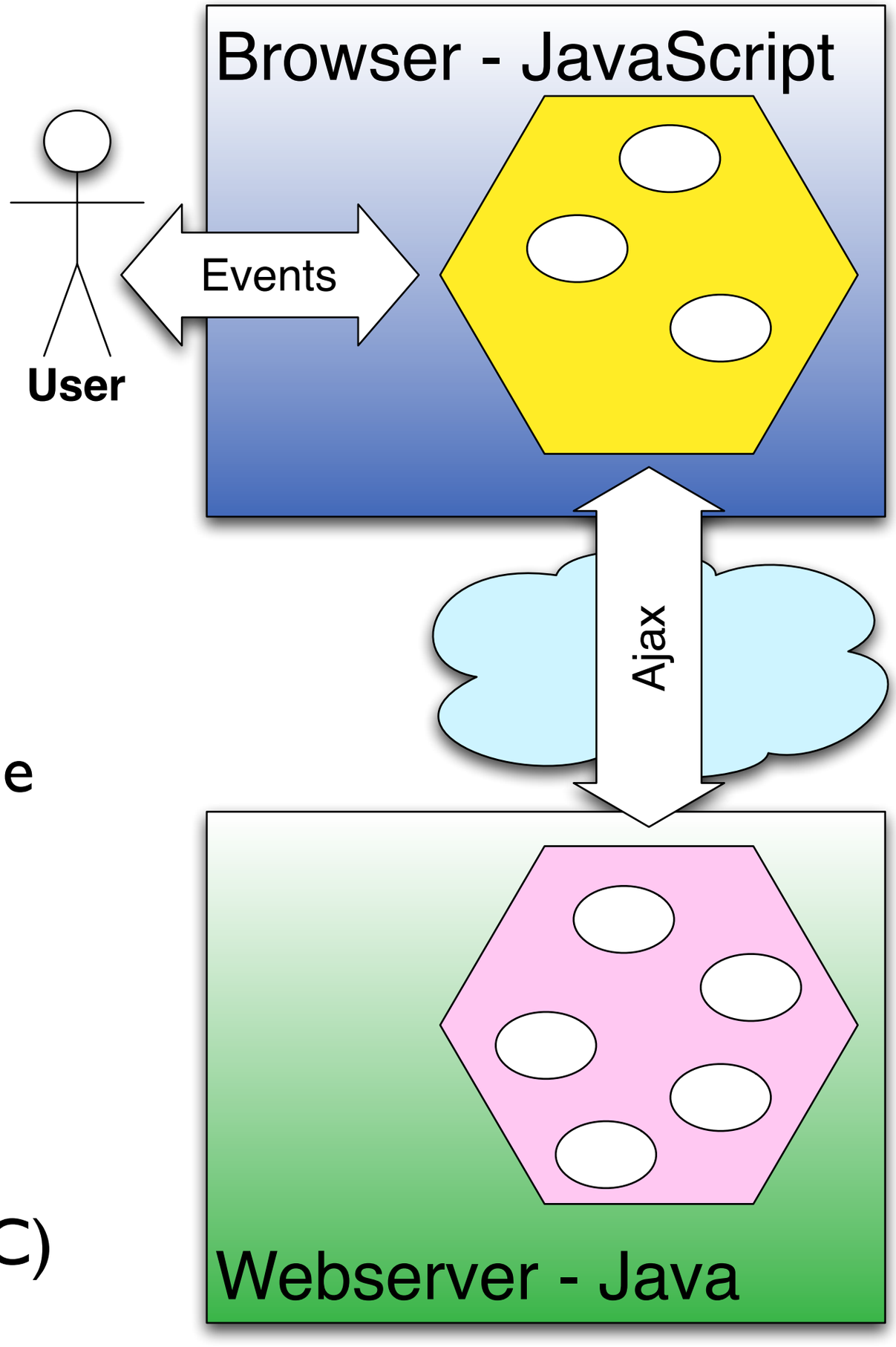


Dual-Exagon Architecture



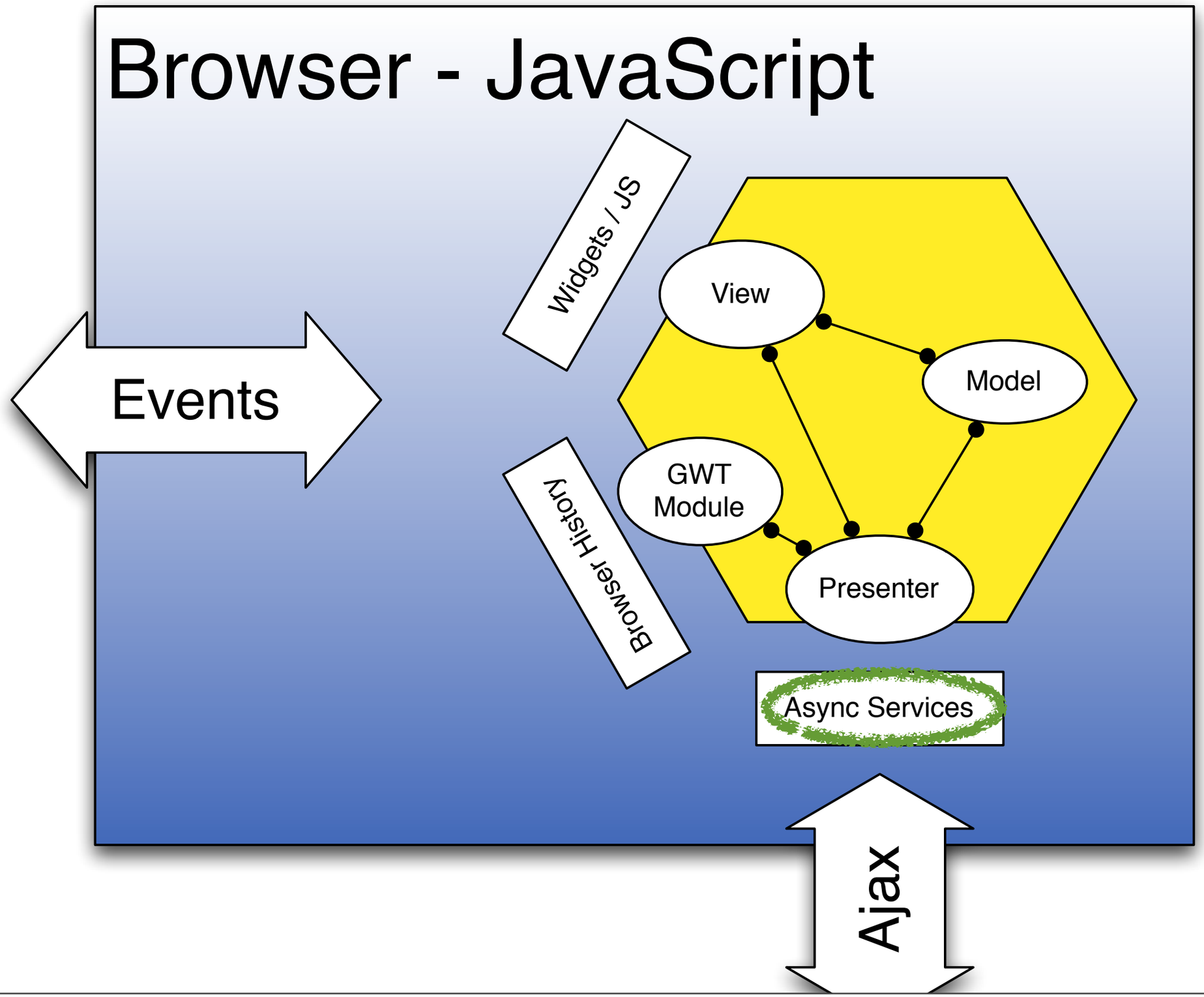
- All client code is on a single html page.
- Internal navigation with dynamic bookmark

Dual-Exagon Architecture



- All client code is on a single html page.
- Internal navigation with dynamic bookmark
- Completely sessionless server (REST or GWT-RPC)

Client

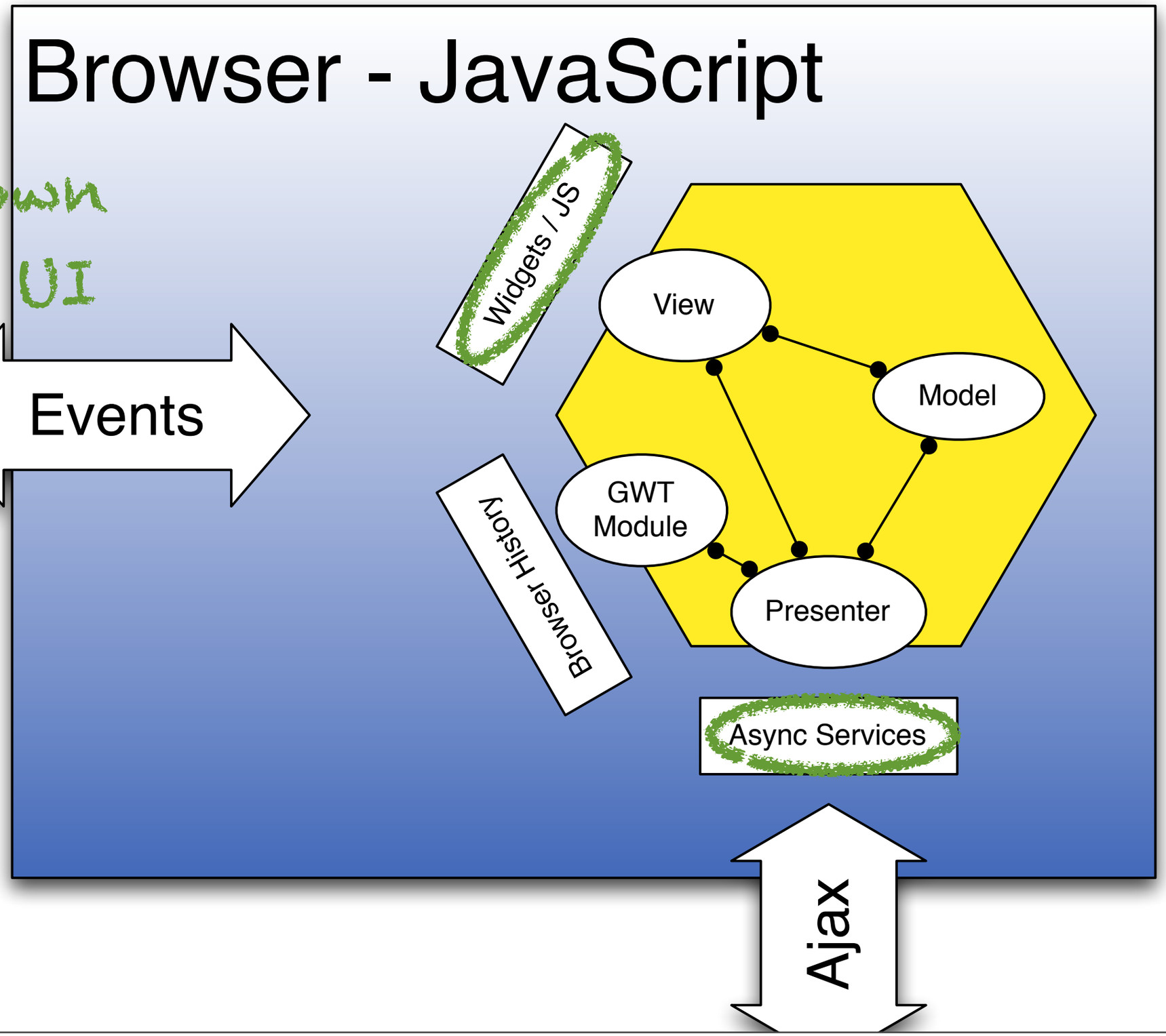


Services based on generics

Client

Write your own widgets for UI

Services based on generics

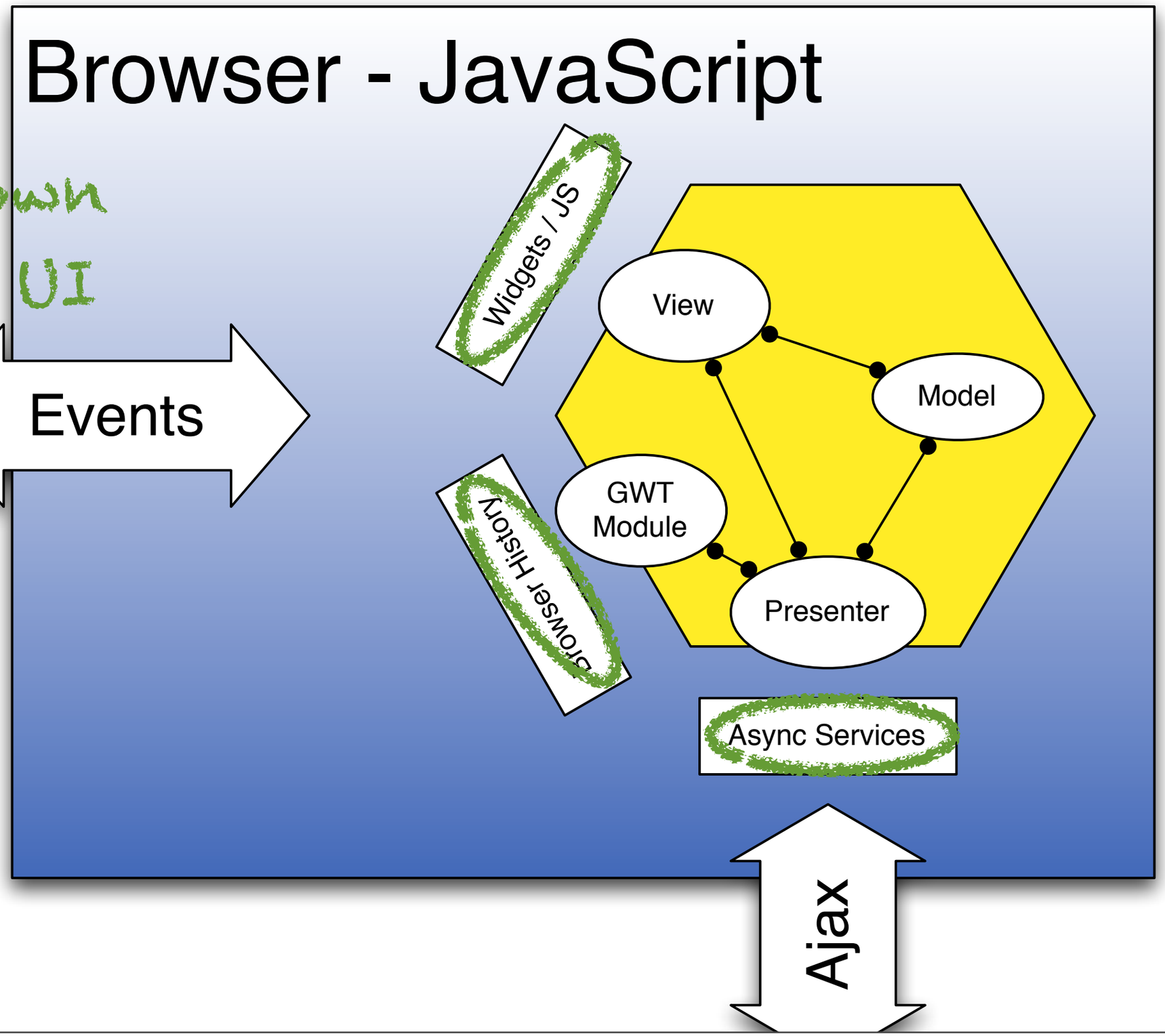


Client

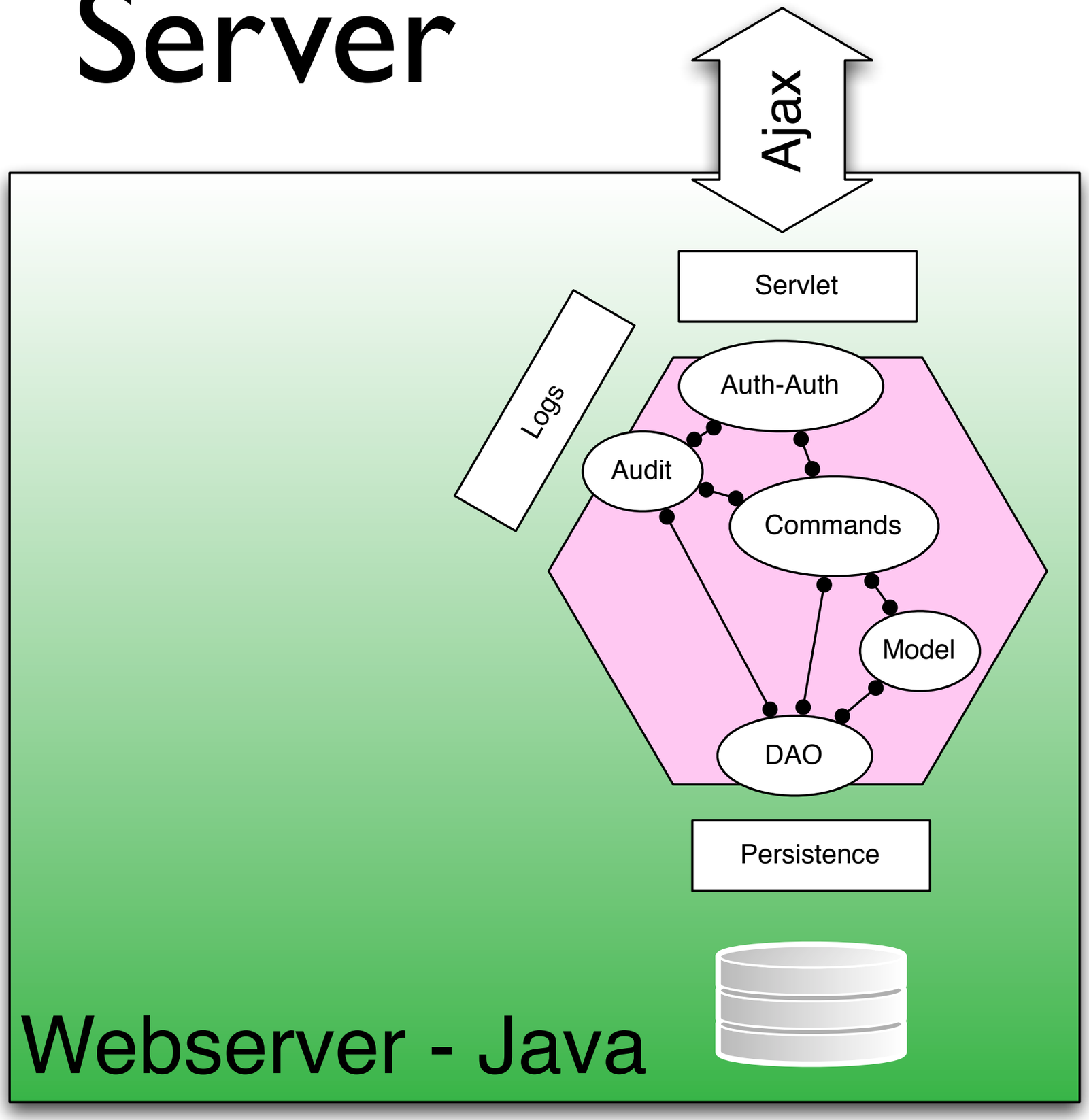
Write your own widgets for UI

Browser events based

Services based on generics

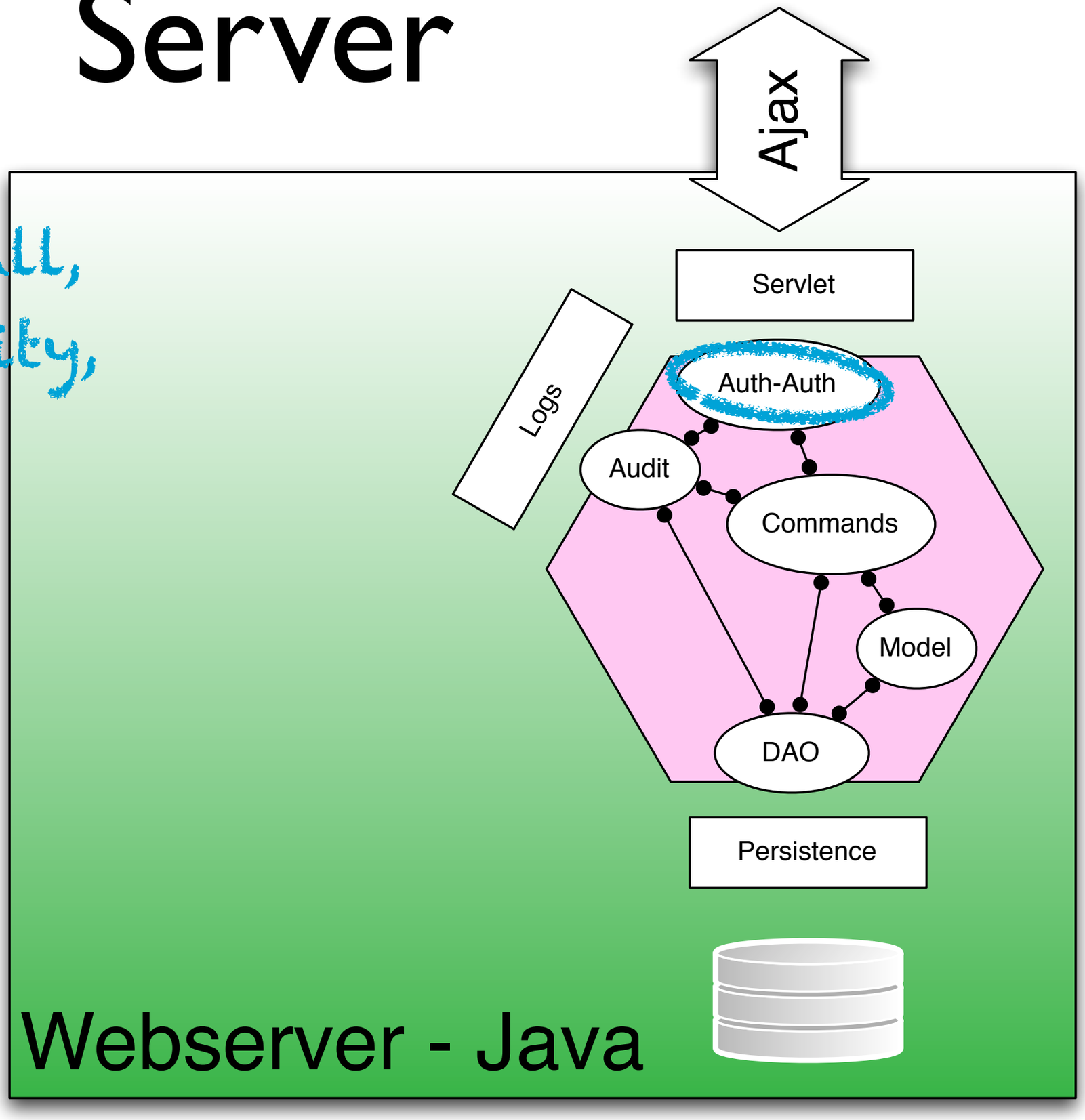


Server


Ajax
Servlet
Logs
Auth-Auth
Audit
Commands
Model
DAO
Persistence
Webserver - Java

Server

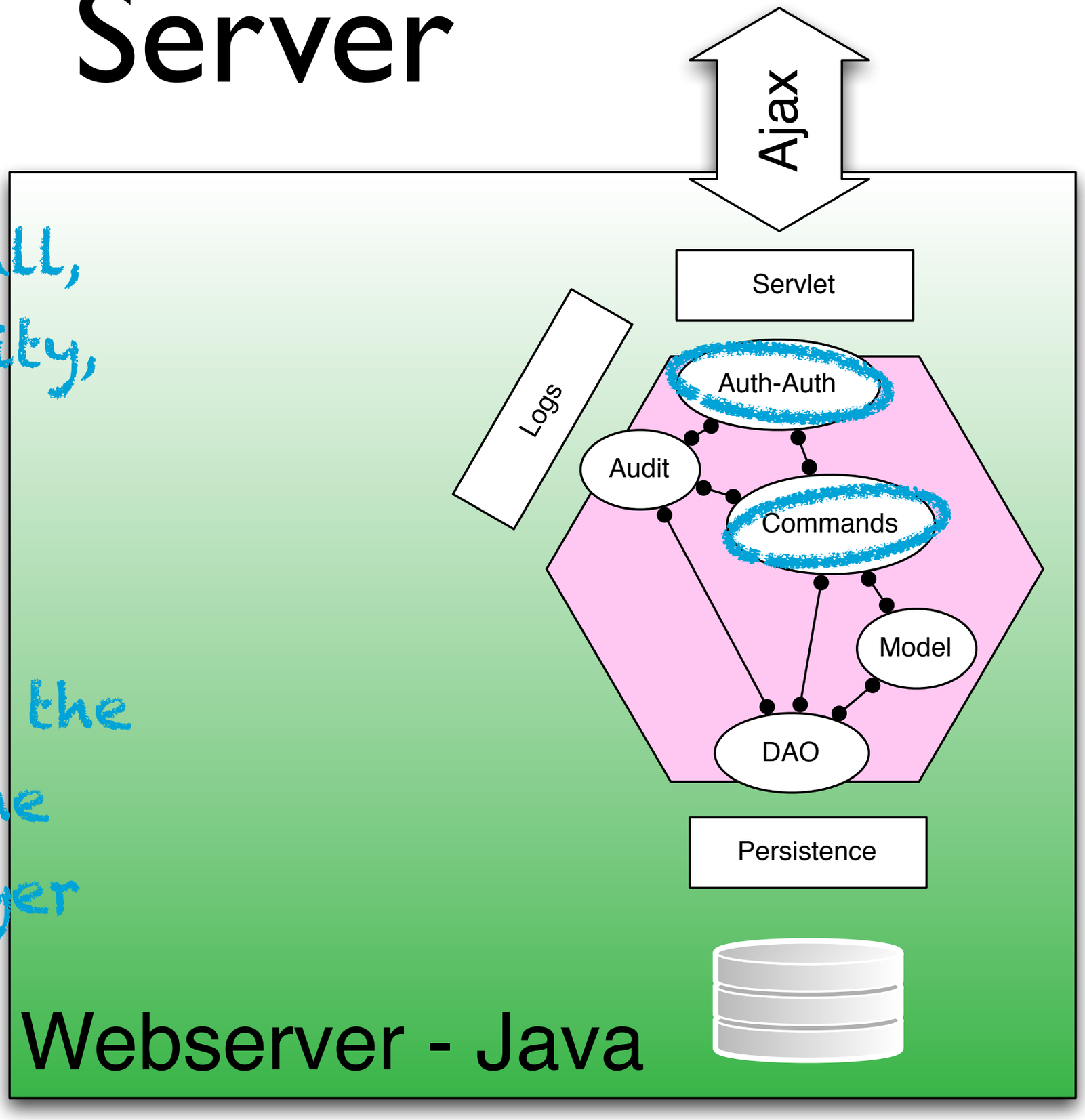
Get the ajax call, check the security, process the command



Server

Get the ajax call, check the security, process the command

Commands call the model and the persistence layer



When things grow up

Sunday, April 17, 2011

When things grow up

- Code split

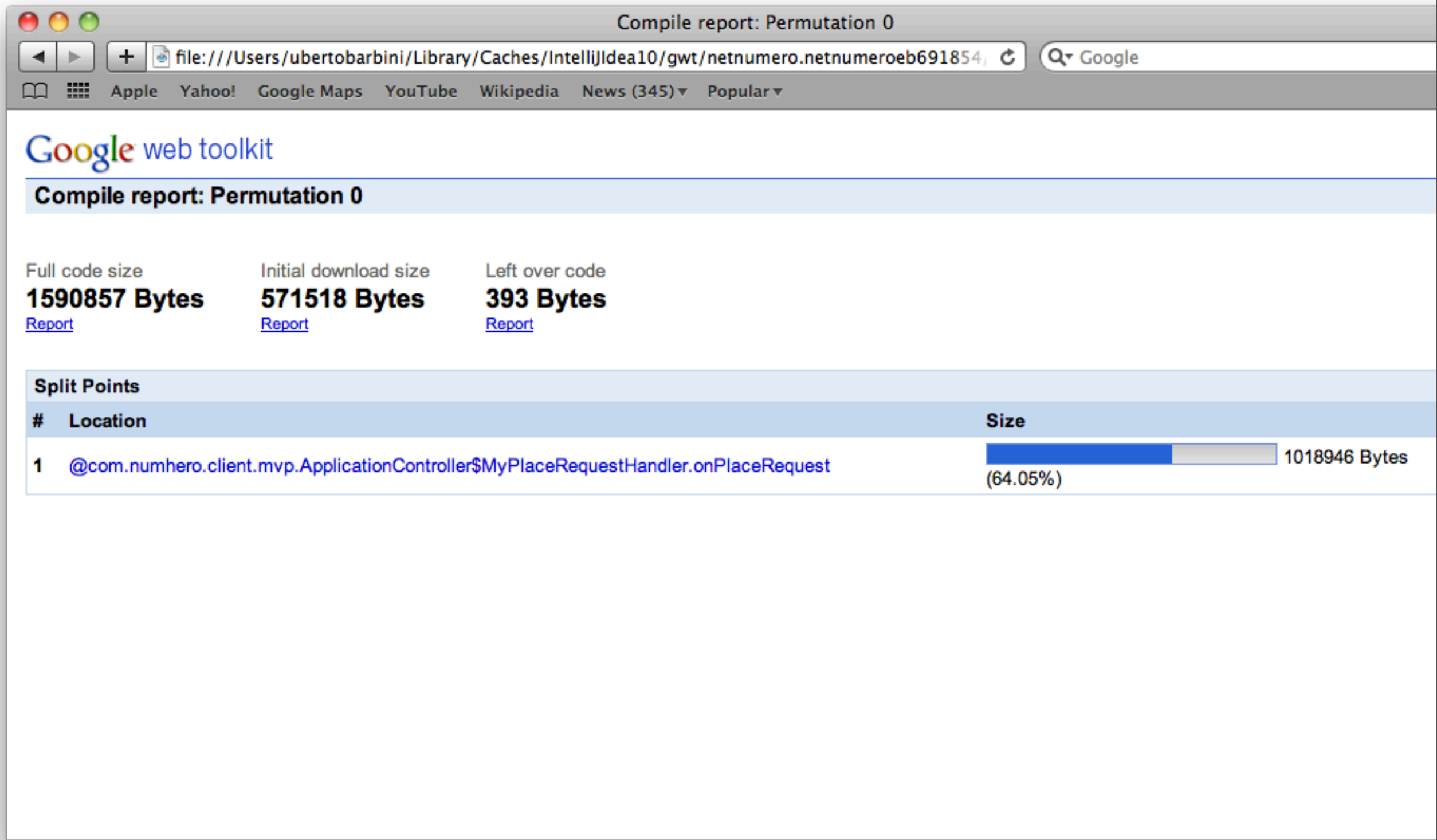
When things grow up

- Code split
- Presenter proxy pattern

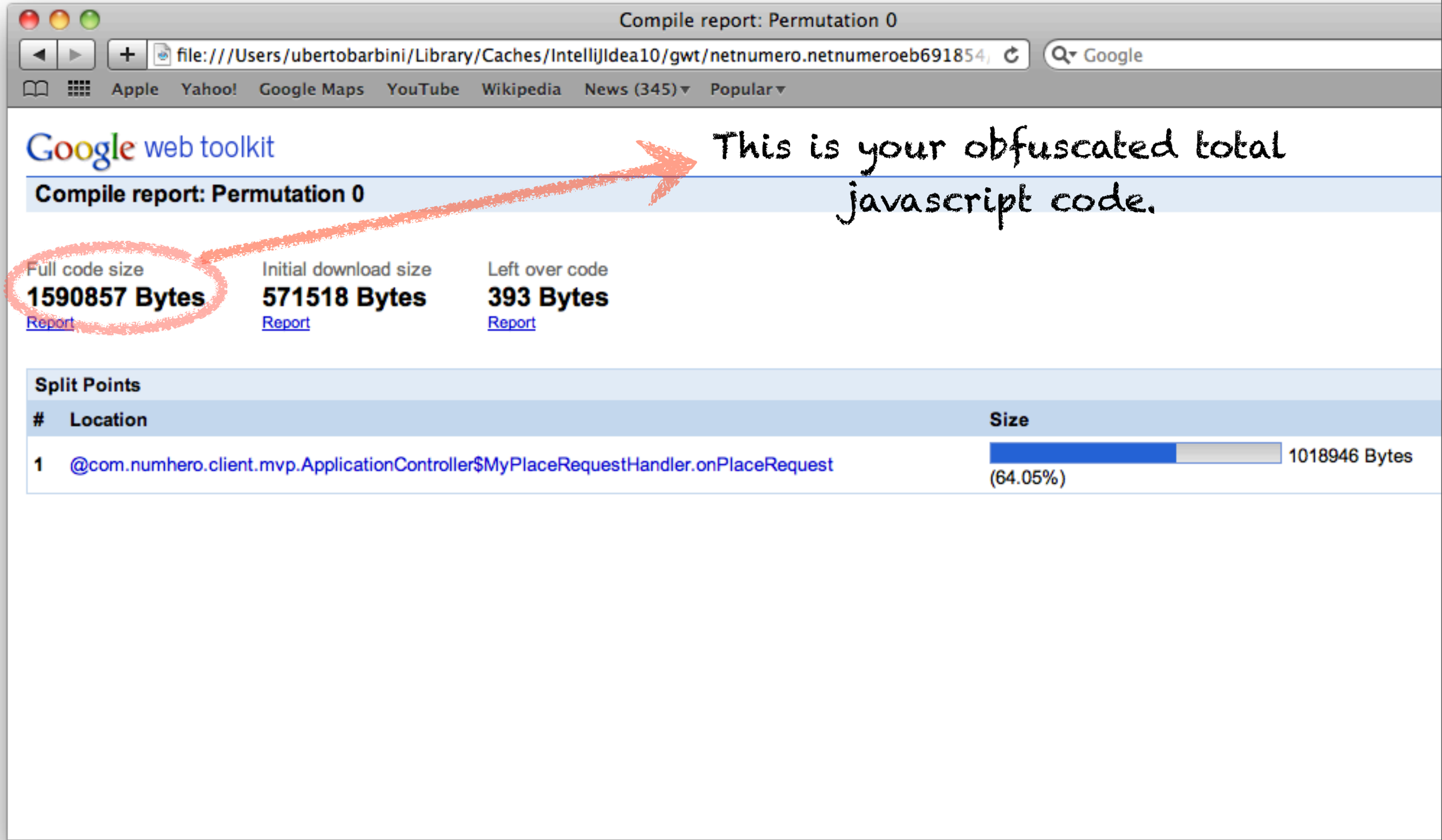
When things grow up

- Code split
- Presenter proxy pattern
- Modular Views and Presenters

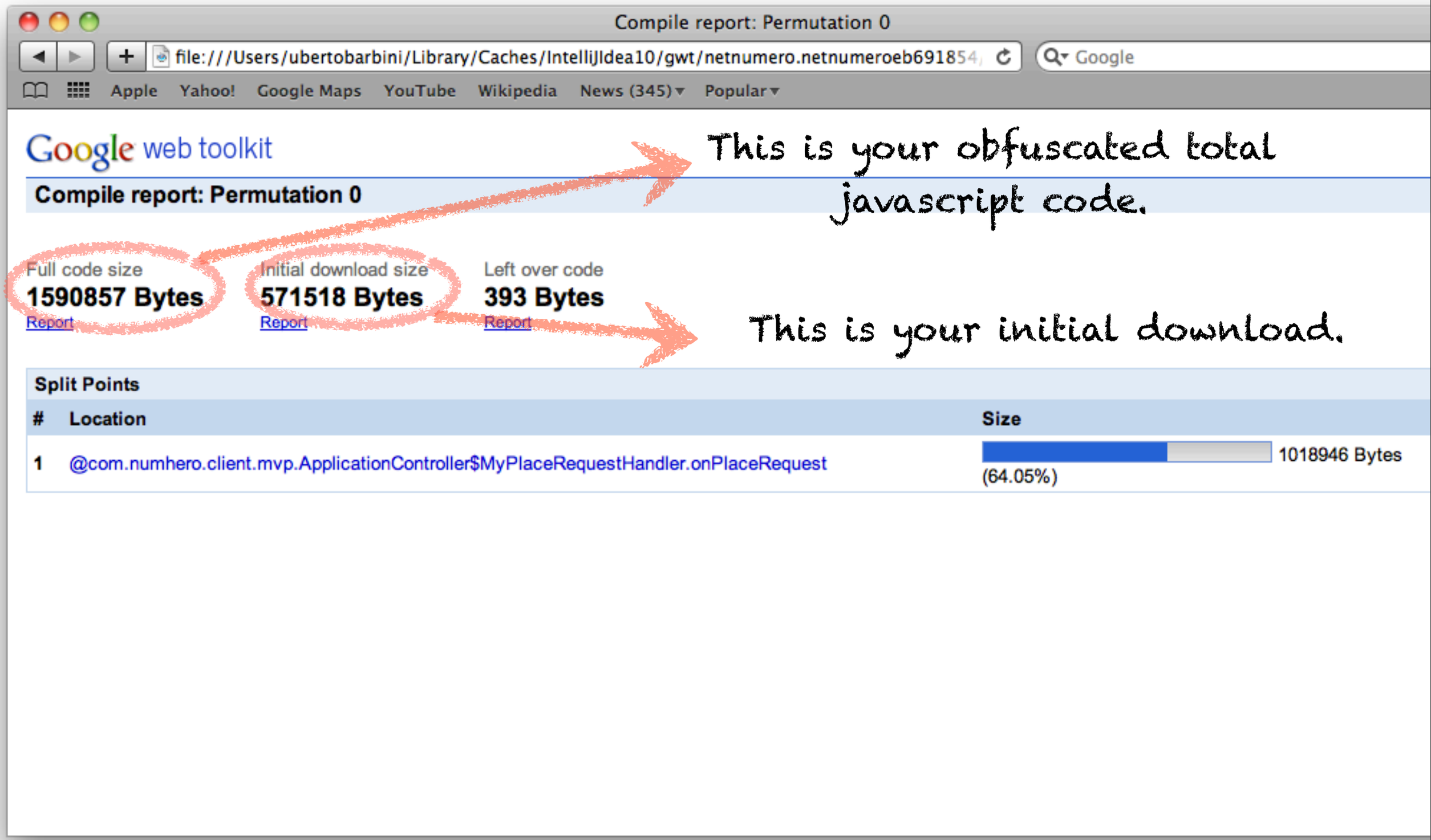
Code split report



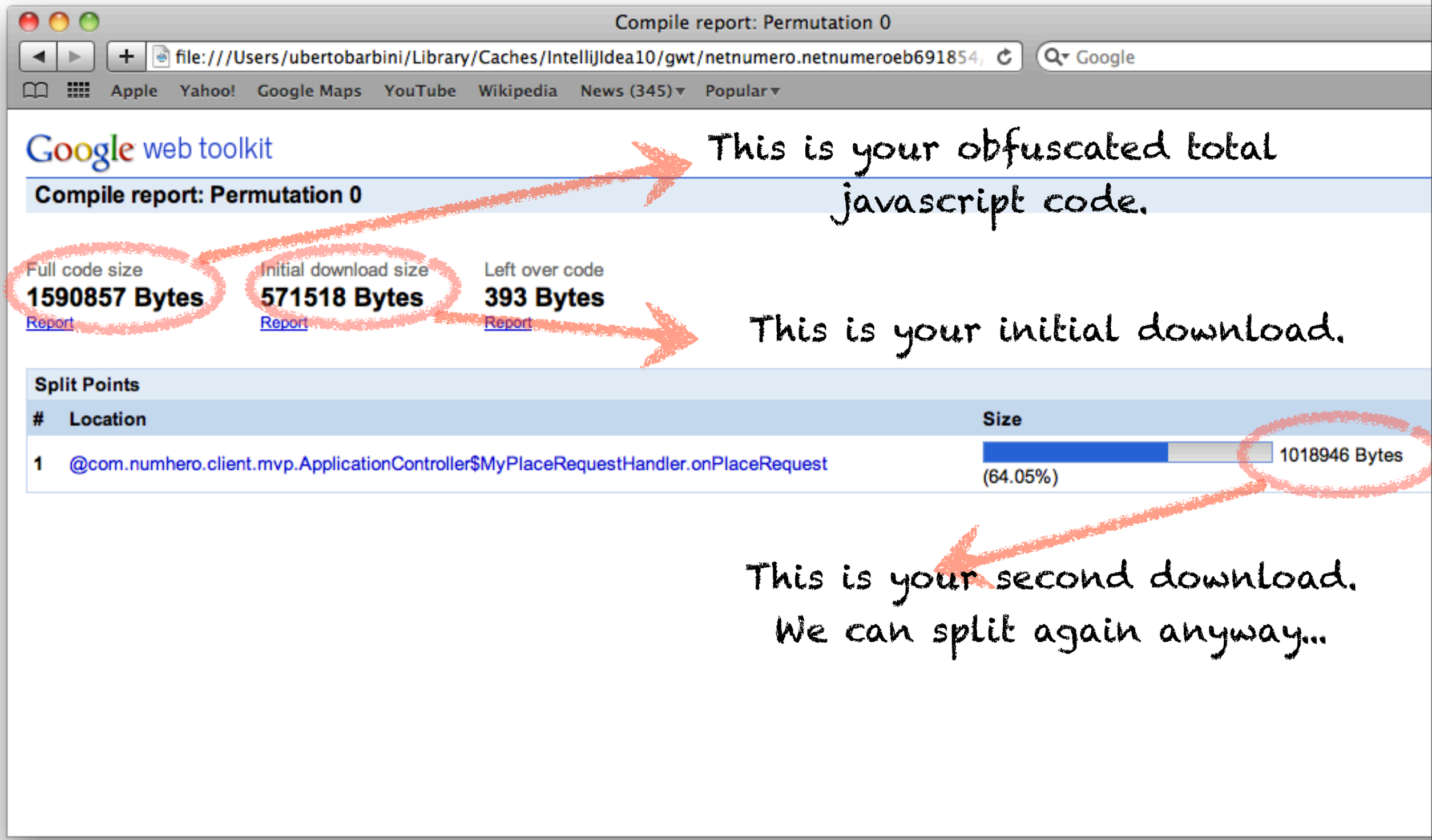
Code split report



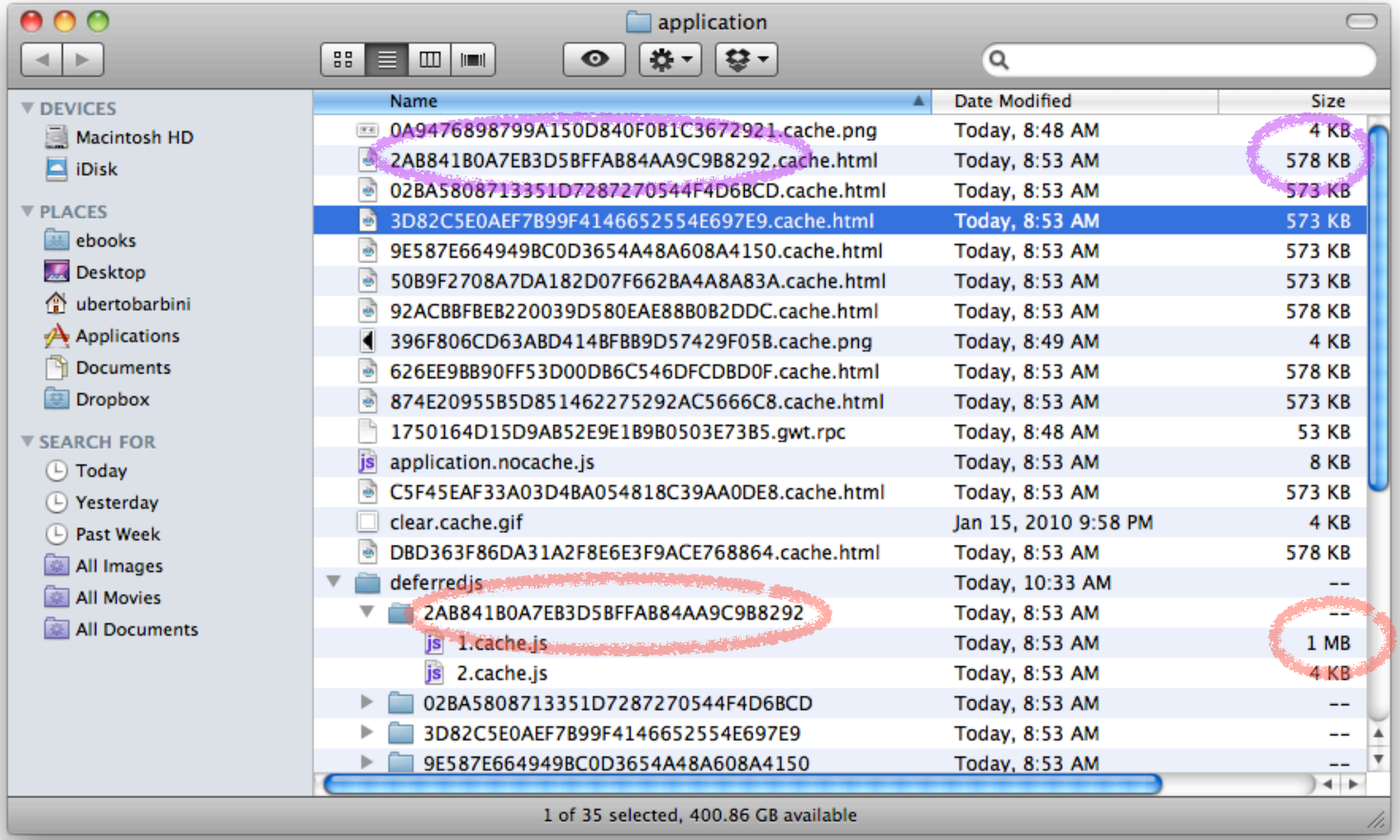
Code split report



Code split report



Split code


application
DEVICES
Macintosh HD
iDisk
PLACES
ebooks
Desktop
ubertobarbini
Applications
Documents
Dropbox
SEARCH FOR
Today
Yesterday
Past Week
All Images
All Movies
All Documents
Name
Date Modified
Size
0A9476898799A150D840F0B1C3672921.cache.png Today, 8:48 AM 4 KB
2AB841B0A7EB3D5BFFAB84AA9C9B8292.cache.html Today, 8:53 AM 578 KB
02BA5808713351D7287270544F4D6BCD.cache.html Today, 8:53 AM 573 KB
3D82C5E0AEF7B99F4146652554E697E9.cache.html Today, 8:53 AM 573 KB
9E587E664949BC0D3654A48A608A4150.cache.html Today, 8:53 AM 573 KB
50B9F2708A7DA182D07F662BA4A8A83A.cache.html Today, 8:53 AM 573 KB
92ACBBFBEB220039D580EAE88B0B2DDC.cache.html Today, 8:53 AM 578 KB
396F806CD63ABD414BFBB9D57429F05B.cache.png Today, 8:49 AM 4 KB
626EE9BB90FF53D00DB6C546DFCDBD0F.cache.html Today, 8:53 AM 578 KB
874E20955B5D851462275292AC5666C8.cache.html Today, 8:53 AM 573 KB
1750164D15D9AB52E9E1B9B0503E73B5.gwt.rpc Today, 8:48 AM 53 KB
application.nocache.js Today, 8:53 AM 8 KB
C5F45EAF33A03D4BA054818C39AA0DE8.cache.html Today, 8:53 AM 573 KB
clear.cache.gif Jan 15, 2010 9:58 PM 4 KB
DBD363F86DA31A2F8E6E3F9ACE768864.cache.html Today, 8:53 AM 578 KB
deferredjs Today, 10:33 AM --
2AB841B0A7EB3D5BFFAB84AA9C9B8292 Today, 8:53 AM --
1.cache.js Today, 8:53 AM 1 MB
2.cache.js Today, 8:53 AM 4 KB
02BA5808713351D7287270544F4D6BCD Today, 8:53 AM --
3D82C5E0AEF7B99F4146652554E697E9 Today, 8:53 AM --
9E587E664949BC0D3654A48A608A4150 Today, 8:53 AM --
1 of 35 selected, 400.86 GB available

- examples:

Sparse thoughts

- It's not that I don't like JS. It's all about tools and having same language everywhere
- The issue with 100% coverage it's not really about tests. If there is a 1% that I cannot test there must be a design problem there.
- Javascript threads and other differences between Java and GWT compilable Java
- View Dom Manipulation and xpath tests
- View test first with UIBindings and widgets


The future...


The future...

The future...

my wish list:

The future...

my wish list:

- Incremental compile

The future...

my wish list:

- Incremental compile
- Multi browsers javascript

The future...

my wish list:

- Incremental compile
- Multi browsers javascript
- Other languages (Python, Scala)

The future...

my wish list:

- Incremental compile
- Multi browsers javascript
- Other languages (Python, Scala)
- GWT toolkit for JS server side?