

What's New in GWT 2.2

David Chandler
Google Web Toolkit Team
Atlanta, GA USA
drfibonacci@google.com



Agenda

Demos

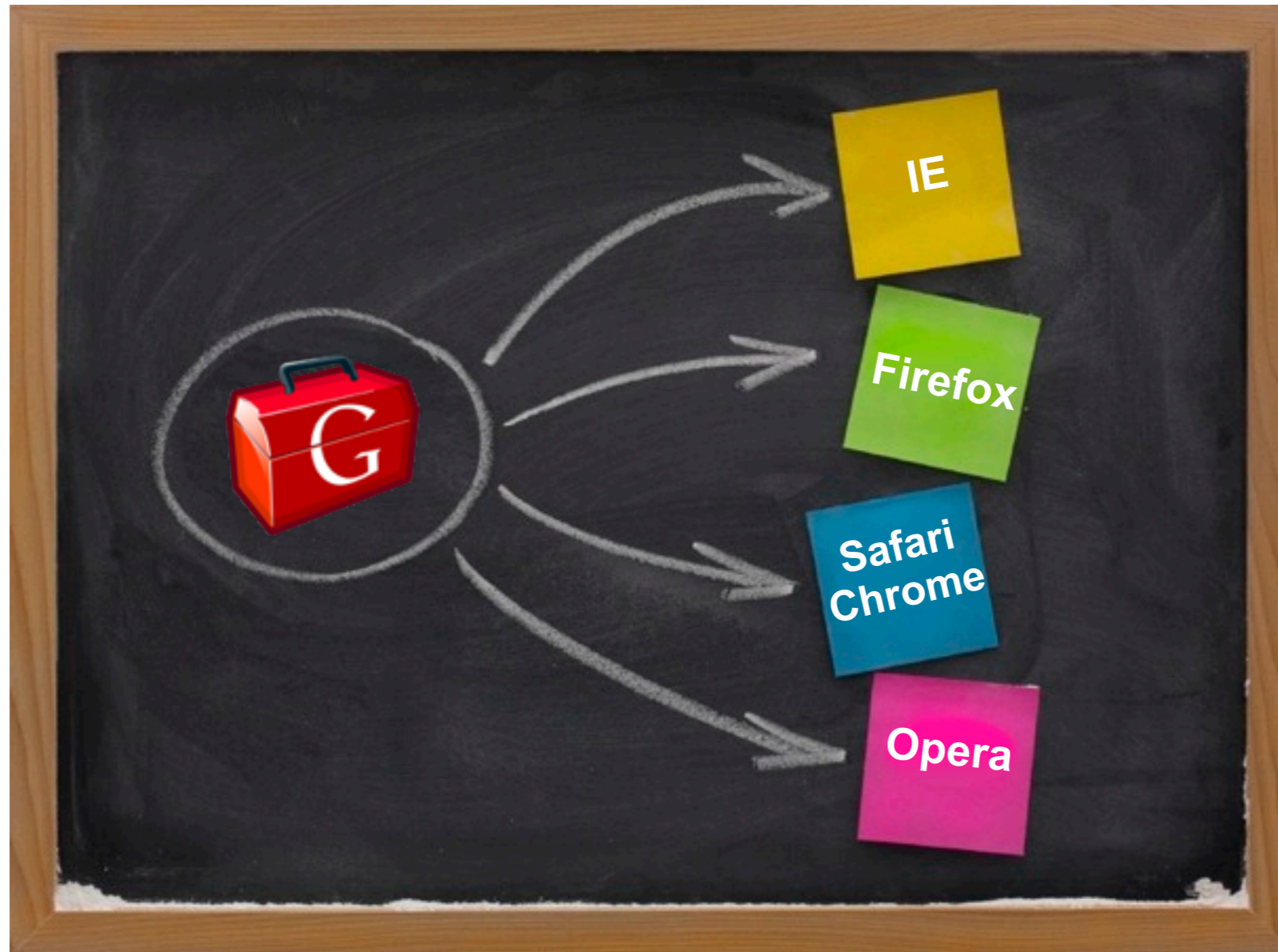
New tools (GWT Designer)

New features

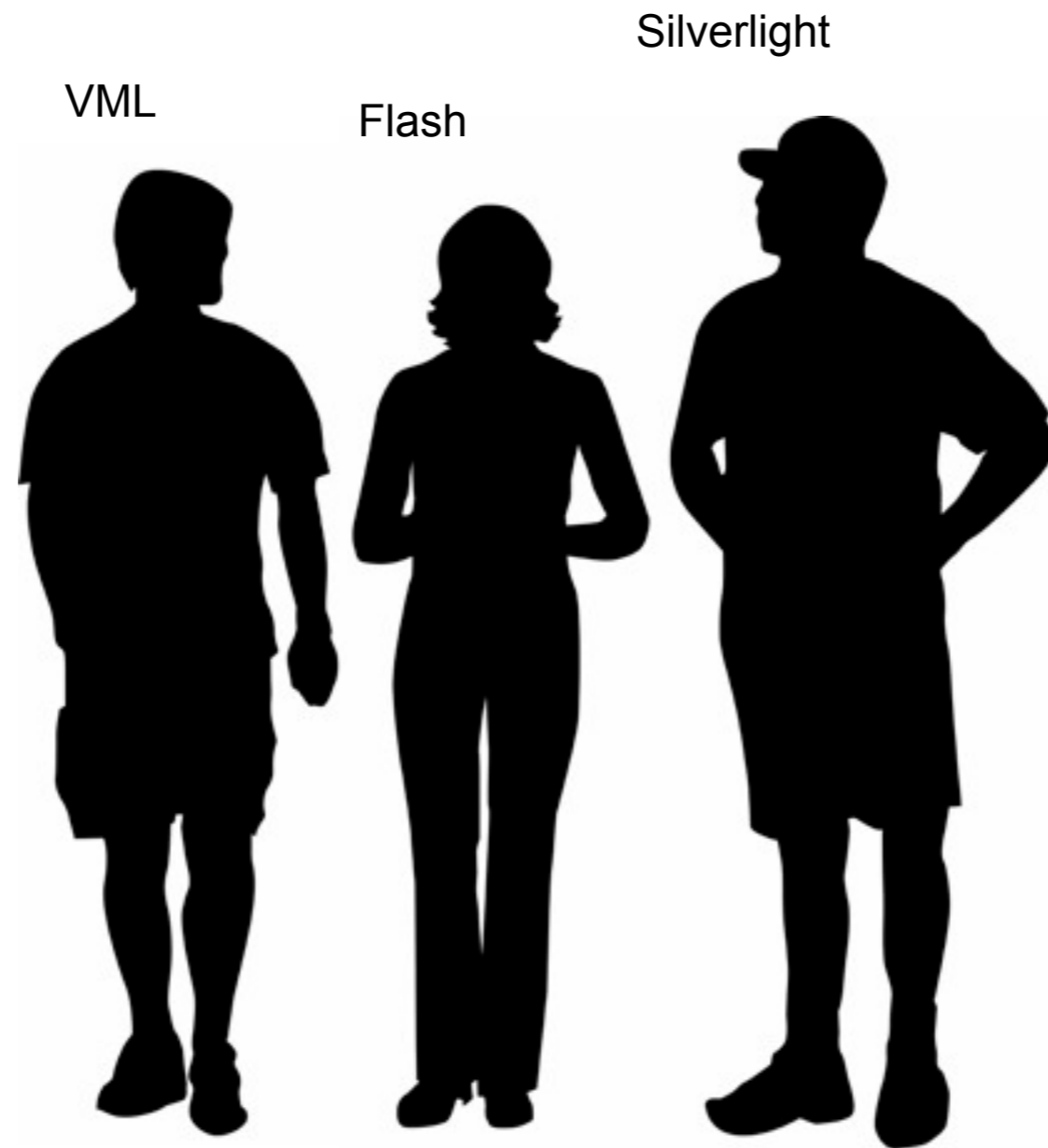
GWT in 10 sec

Aynchronous
Java~~Script~~
And
XML++

Browser-Proof Your JS Code



No plugins required



Eating our own dogfood

Google AdWords

Customer ID: [redacted] | [New](#) | [Announcements \(1\)](#)

Home Campaigns Opportunities Reporting and Tools Billing My account

None of your ads are running because all of your active campaigns have ended. [View](#)

All online campaigns Last 7 days
Nov 7, 2010 - Nov 13, 2010

Campaigns Ad groups Settings Ads Keywords Networks Audiences

All but deleted campaigns Segment Filter Columns Search

View Chart

[+ New campaign](#) [Change status...](#) [Alerts](#)

<input type="checkbox"/>	<input checked="" type="checkbox"/>	Campaign	Budget	Status	Clicks	Impr.	CTR	Avg. CPC	Cost
<input type="checkbox"/>	<input checked="" type="checkbox"/>	US	\$5.00/day	Ended	0	0	0.00%	\$0.00	\$0.00
Total - all but deleted campaigns					0	0	0.00%	\$0.00	\$0.00

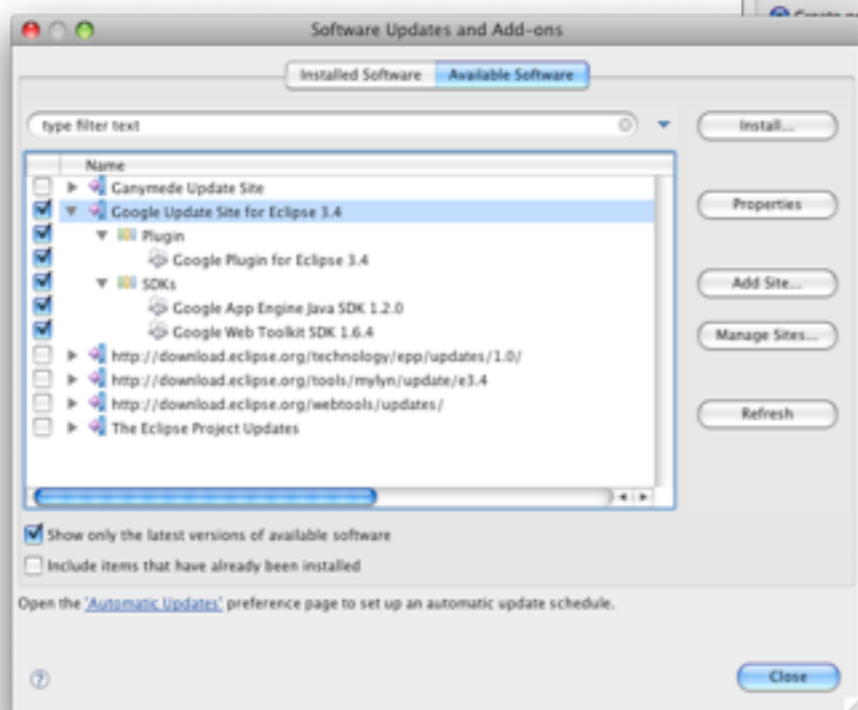
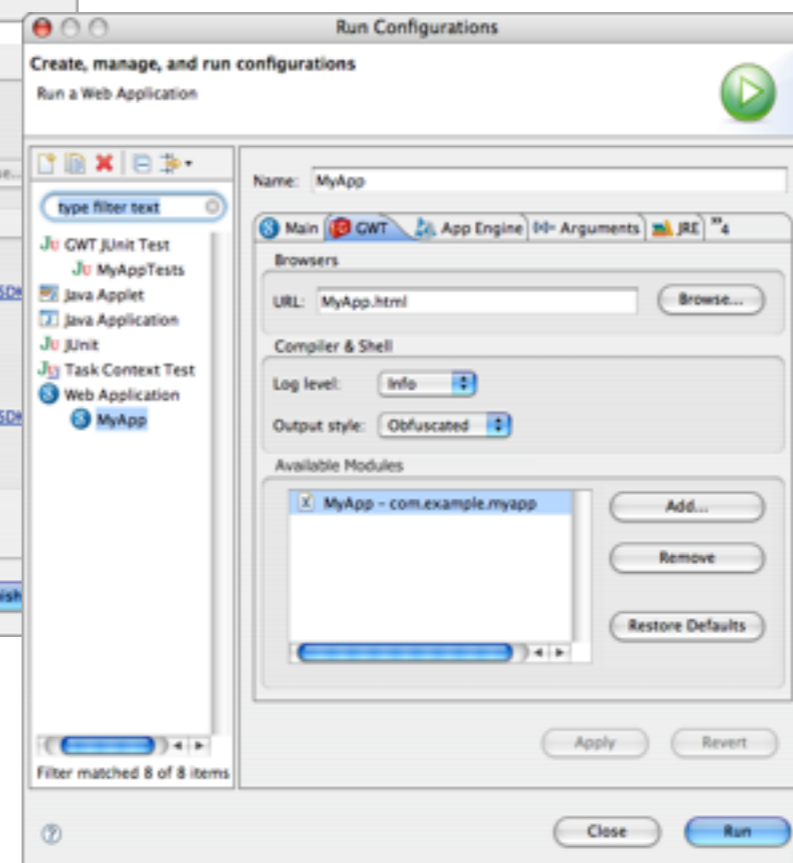
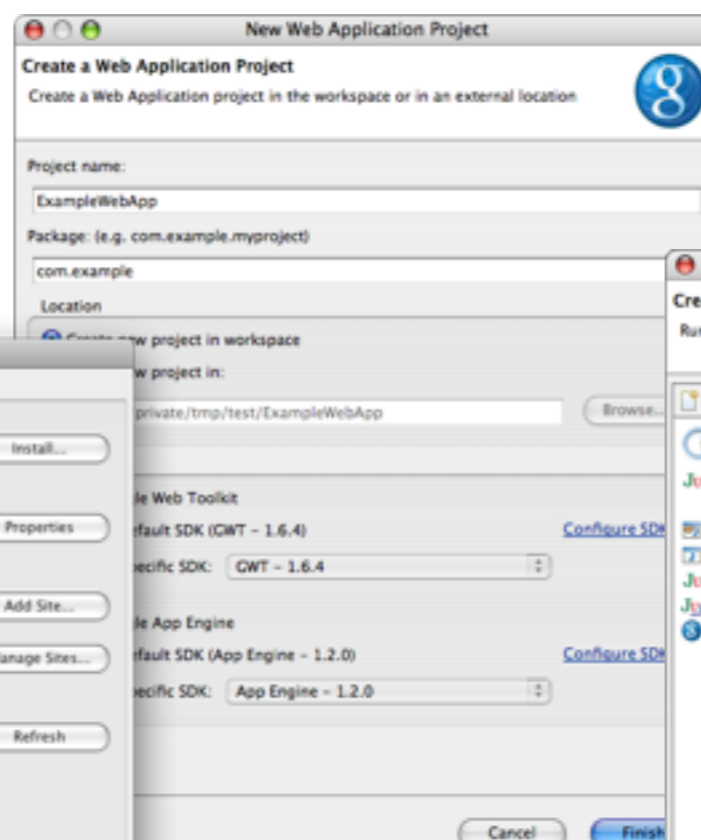
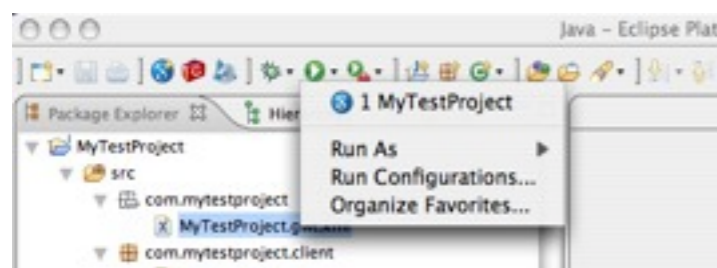
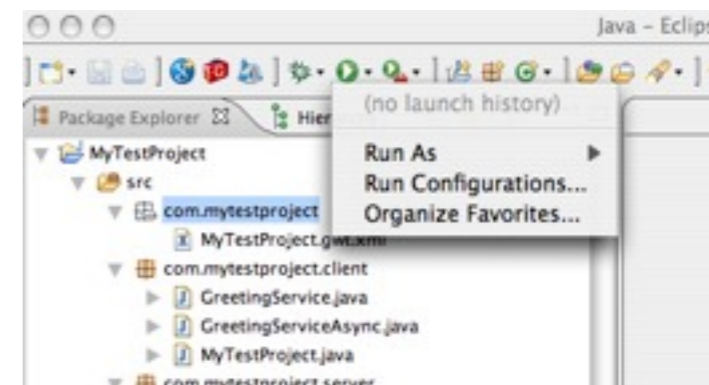
+ = AdSense, Maps, Docs, Groups, Blogger...

New tools

Google Plugin for Eclipse



```
private static native void jsniMethod(boolean sayHi)/*- {  
    // Display a pop-up  
    if (sayHi) {  
        var name = this.@com.example.myapp.client.MyApp::name;  
        window.alert('Hello, ' + name);  
    }  
}/*-*/;
```



GWT Quickstart

```
> svn checkout https://google-web-toolkit.googlecode.com/svn/trunk/samples/expenses expenses
```

```
> cd expenses/src/main/webapp
```

```
> mvn -f ../../../../pom.xml gwt:run (dev mode)
```

(note: most projects can use mvn gwt:run in the dir containing pom.xml--this one is special because it loads test data from a .txt file in src/main/webapp)

1. Browse to <http://127.0.0.1:8888/LoadExpensesDB.html?gwt.codesvr=127.0.0.1:9997>

2. Click Generate Data button

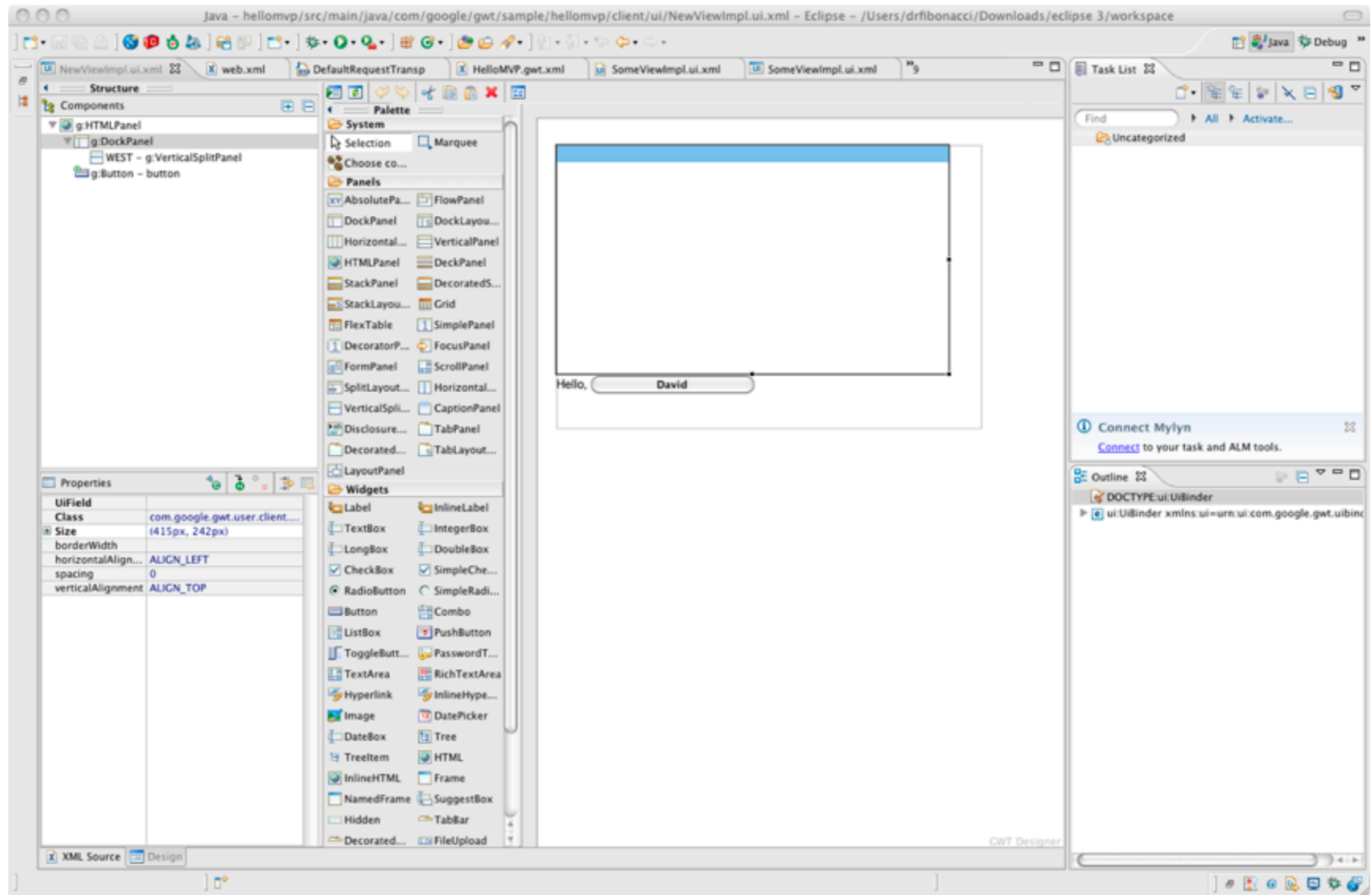
3. Browse to Expenses.html to run the app

To create an Eclipse project:

1. Install Sonatype m2eclipse and m2extras

2. File | Import | Existing maven project, select expenses/pom.xml

GWT Designer Java, ui.xml

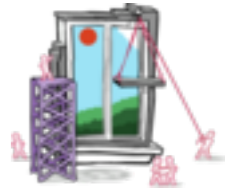


GWT Designer & UiBinder



```
<!DOCTYPE ui:UiBinder SYSTEM "http://dl.google.com/gwt/DTD/xhtml.ent">
<ui:UiBinder xmlns:ui="urn:ui:com.google.gwt.uibinder"
            xmlns:g="urn:import:com.google.gwt.user.client.ui">
  <ui:style>
    .important {
      font-weight: bold;
    }
  </ui:style>
  <g:HTMLPanel>
    <span class="{style.important}" ui:field="nameSpan" />
    <g:VerticalPanel height="203px">
      <g:HTML wordWrap="true">
        <h1>Hello</h1>
      </g:HTML>
      <g:Anchor ui:field="goodbyeLink" text="Say good-bye"></g:Anchor>
    </g:VerticalPanel>
  </g:HTMLPanel>
</ui:UiBinder>
```

GWT Designer CSS Editor



The screenshot displays the GWT Designer CSS Editor interface. On the left, a list titled "CSS rules in file" contains several selectors, with ".sendButton" selected. A "Selector filter" input field is positioned above the list. Below the list are buttons for "Add...", "Rename...", "Edit...", "Remove...", and "Sort".

The main area is divided into two panes. The left pane, titled "CSS Rule Editor: .sendButton", has tabs for "Font", "Background", "Box", "Border", "Text", and "Other". The "Font" tab is active, showing settings for Family (sans-serif), Size (16), Style (normal), Variant, Weight (bold), Stretch, and Color (Blue). A "Decoration" section includes checkboxes for none, underline, overline, line-through, and blink. A "Clear" button is at the bottom of this pane.

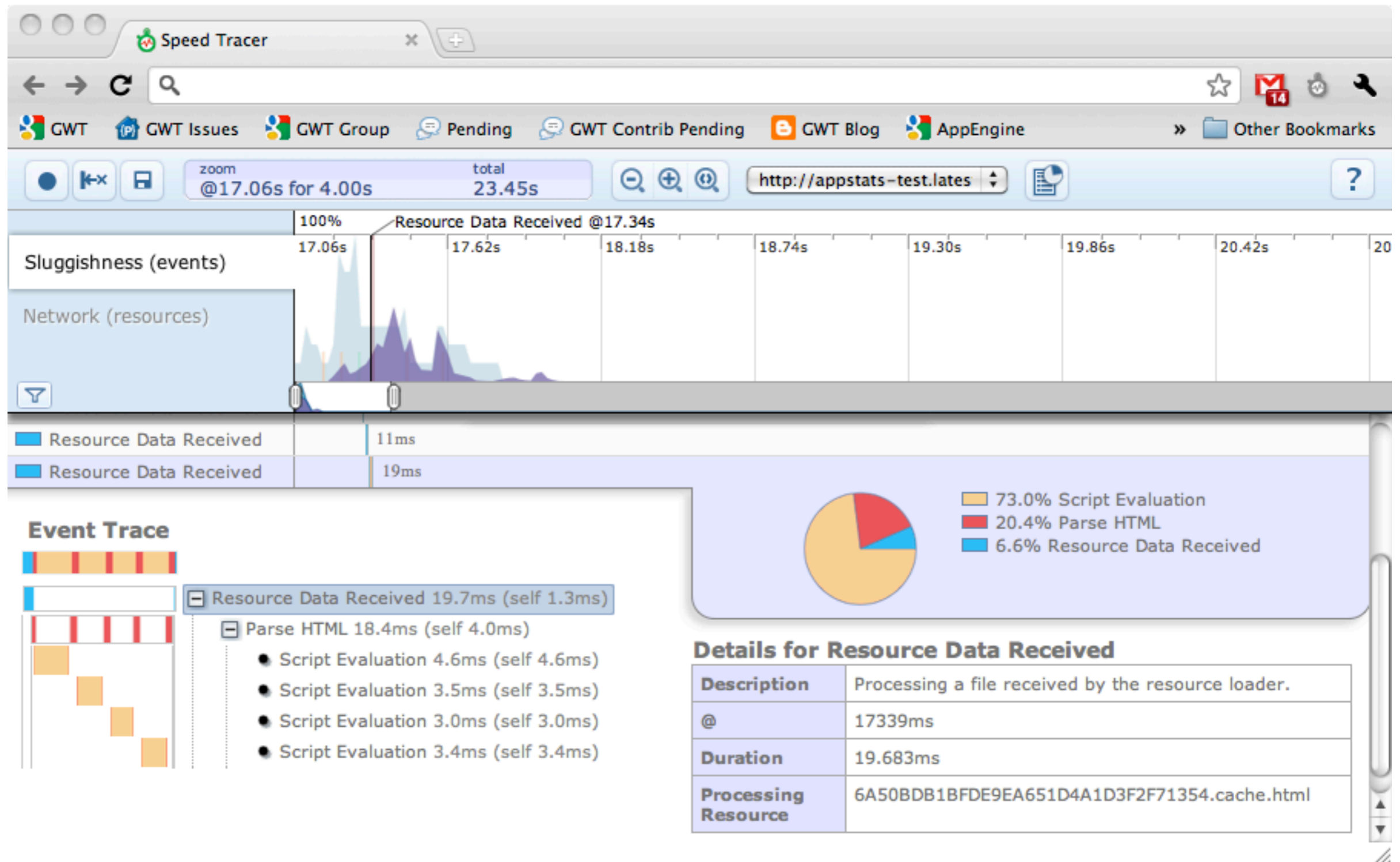
The right pane, titled "CSS rule", shows the CSS code for the selected rule:

```
.sendButton {  
  display: block;  
  font-size: 16pt;  
}
```

Overlaid on the right side is a preview window showing the rendered text: "Nolite mittere margaeritas ante porcas! *Latin proverb*". Below this, a portion of another preview is visible, showing "However you should" and "cas! *Latin*".

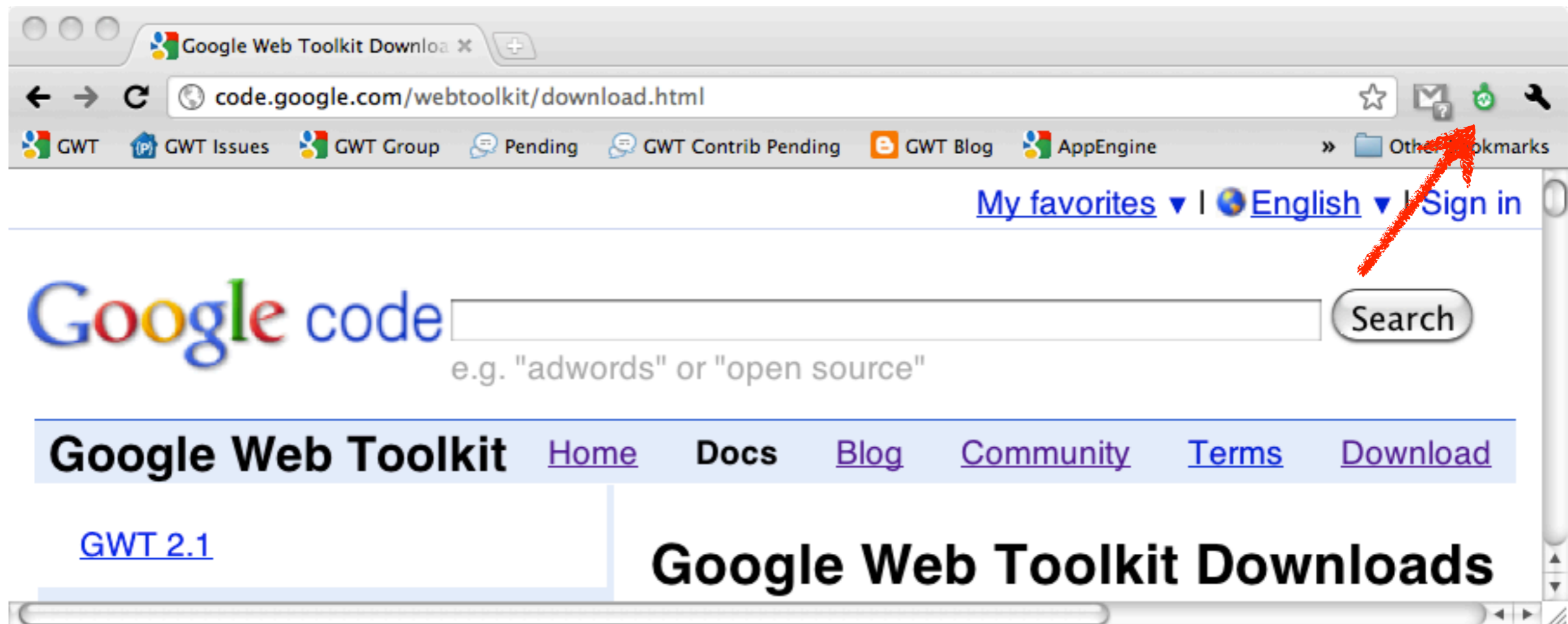
At the bottom of the main editor window, there are "OK" and "Cancel" buttons.

Optimize with Speed Tracer



Launch SpeedTracer

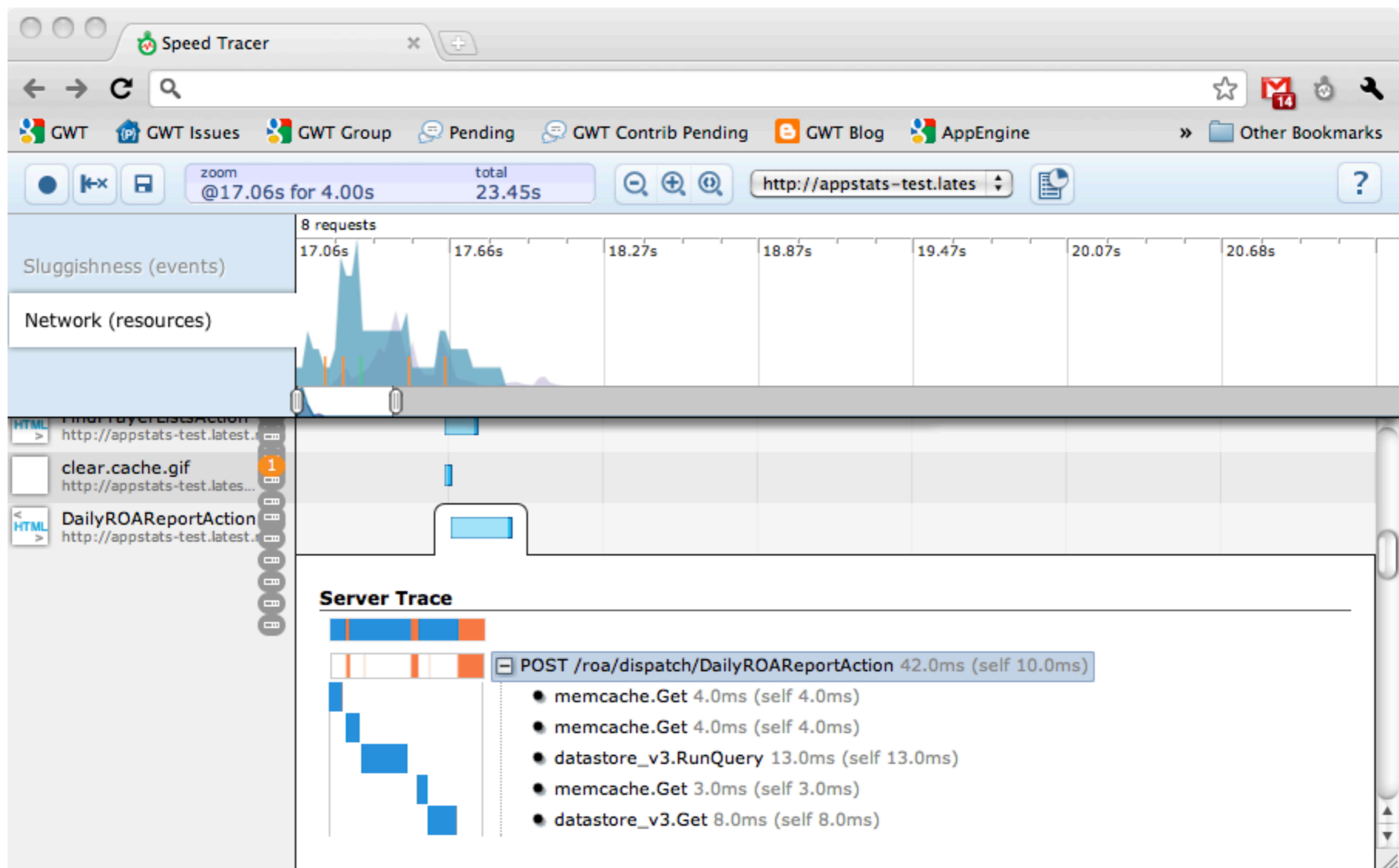
From Chrome...



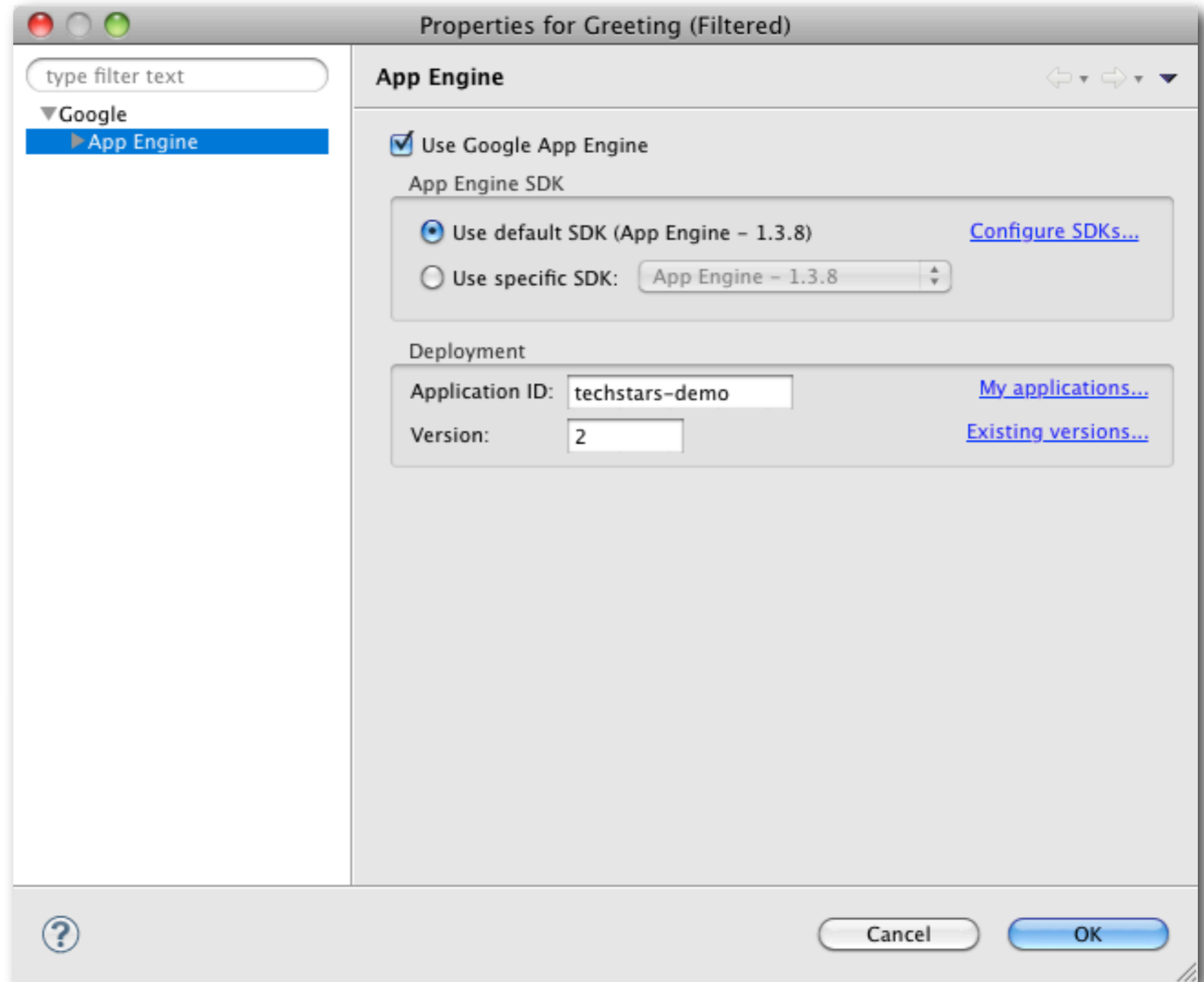
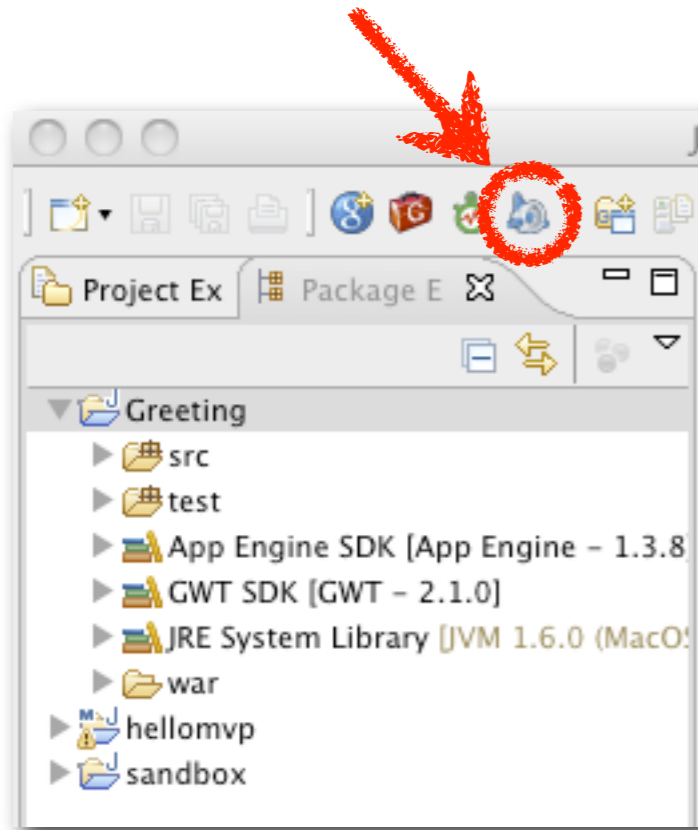
Or from Eclipse...



Server traces GAE, tcServer



One click deploy to GAE



New features

What's new in 2.2?

SafeHTML wrappers

Application framework

- RequestFactory
- Activities and Places

Cell Widgets and Editor framework

GWT Canvas

Built-in logging

Four flavors of RPC

1. RequestBuilder + JSONParser (see RESTY-GWT)
2. RequestBuilder + XMLParser
3. GWT-RPC (easiest)
4. RequestFactory (new in 2.1)

GWT-RPC

```
@RemoteServiceRelativePath("greet")  
public interface GreetingService extends RemoteService {  
    String greetServer(String name) throws IllegalArgumentException;  
}
```

```
public interface GreetingServiceAsync {  
    void greetServer(String input, AsyncCallback<String> callback)  
        throws IllegalArgumentException;  
}
```

```
public class GreetingServiceImpl extends RemoteServiceServlet implements  
    GreetingService  
{  
    public String greetServer(String input) throws IllegalArgumentException {  
        ...  
    }  
}
```

Simple, powerful GWT-RPC

Send / receive Plain Old Java Objects (POJO)

- Easy: interface, async, & implementation
- Versioned to help keep client & server in sync
- Even smaller than JSON
- Supports polymorphism
- No JavaScript hijacking risk (JSON attack)
- Easy to find all calls to given service in IDE

GWT 2.1 RequestFactory

Newer alternative to GWT-RPC

Designed for data-oriented services

- Higher level of abstraction than GWT-RPC
- Foundation for future caching / batching

Even faster than GWT-RPC

- JSON-based == very fast (no serialization / deserialization required)
- Tracks changes on the client and sends **only diffs**

RequestFactory

The entity / DTO problem

EntityProxy / ValueProxy

Service stub interfaces extend RequestContext

AppRequestFactory extends RequestFactory

GWT.create(MyAppRequestFactory.class)

EntityProxy

```
@Entity
public class ItemList extends DatastoreObject
{
    private String name;
    private Key<AppUser> owner;
    private ListType listType;
    @Embedded
    private List<ListItem> items; // value type
    ...
}
```

```
@ProxyFor(ItemList.class)
public interface ItemListProxy extends DatastoreObjectProxy
{
    // TODO enums work!
    public enum ListType {NOTES, TODO}
    String getName();
    void setName(String name);
    List<ListItemProxy> getItems();
    AppUserProxy getOwner(); // NOT Key
    ...
}
```

DatastoreObject

```
package com.listwidget.domain;
public class DatastoreObject
{
    @Id
    private Long id;
    private Integer version = 0;

    @PrePersist
    void onPersist()
    {
        this.version++;
    }
    ...
}
```

```
package com.listwidget.shared.proxy;
public interface DatastoreObjectProxy extends EntityProxy
{
    Long getId();
    Integer getVersion();
}
```

ValueProxy

```
public class ListItem // POJO
{
    private String itemText;
    private Date dateCreated;

    public Date getDateCreated()
    {
        return dateCreated;
    }
}
```

```
@ProxyFor(value = ListItem.class)
public interface ListItemProxy extends ValueProxy
{
    String getItemText();
    void setItemText(String itemText);
    Date getDateCreated();
}
```

Making a RequestFactory

```
public interface ListwidgetRequestFactory extends RequestFactory
{
    @Service(value = ItemListDao.class, locator = DaoServiceLocator.class)
    interface ItemListRequestContext extends RequestContext
    {
        Request<List<ItemListProxy>> listAll();
        Request<Void> save(ItemListProxy list);
        Request<ItemListProxy> saveAndReturn(ItemListProxy newList);
    }
    ItemListRequestContext itemListRequest();
}
```

```
private final ListwidgetRequestFactory rf =
    GWT.create(ListwidgetRequestFactory.class);
```

Using RequestFactory

```
@Override
public void persistList(String listName)
{
    final ListwidgetRequestFactory rf = this.clientFactory
        .getRequestFactory();
    ItemListRequestContext reqCtx = rf.itemListRequest();
    final ItemListProxy newList = reqCtx.create(ItemListProxy.class);
    newList.setName(listName);
    newList.setListType(ListType.TODO);
    reqCtx.saveAndReturn(newList).fire(new Receiver<ItemListProxy>()
    {
        @Override
        public void onSuccess(final ItemListProxy savedList)
        {
            // Refresh table
            listDataProvider.getData();
        }
    });
}
```

Using RequestFactory

```
@Override
public void update(int index, ItemListProxy obj, final String newName)
{
    ItemListRequestContext reqCtx = clientFactory.getRequestFactory()
        .itemListRequest();
    ItemListProxy editable = reqCtx.edit(obj);
    editable.setName(newName);
    reqCtx.save(editable).fire(new Receiver<Void>()
    {
        @Override
        public void onSuccess(Void response)
        {
            EventBus.fireEvent(new MessageEvent(newName + " updated",
                MessageType.INFO));
        }
    });
}
```

Using RequestFactory

```
private void getData()
{
    // To retrieve relations and value types, use .with()
    Request<List<ItemListProxy>> findAllReq = rf.itemListRequest()
        .listAll().with("owner");
    // Receiver specifies return type
    findAllReq.fire(new Receiver<List<ItemListProxy>>()
    {
        @Override
        public void onSuccess(List<ItemListProxy> response)
        {
            updateRowData(0, response);
        }
    });
}
```

Using RequestFactory

```
editList = reqCtx.edit(editList);
List<ListItemProxy> items = editList.getItems();
// must initialize collections
if (items == null)
{
    editList.setItems(new ArrayList<ListItemProxy>());
}
editList.getItems().add(newItem);
reqCtx.save(editList).with("items").to(new Receiver<Void>())
{
    @Override
    public void onSuccess(Void response)
    {
        itemsProvider.setList(editList.getItems());
        itemsProvider.refresh();
    }
}).fire();
```

GWT MVP - Concepts

View

- Interface + implementation
- Interface enables testing without GWTTestCase
- Typically expensive to construct so make reusable

Presenter

- No Widgets, just business logic
- Middle man between service layer and views

GWT MVP - Concepts

Place

- Place represents bookmarkable state of an activity
- PlaceController makes back button / bookmarks work like users expect
- PlaceTokenizers map to / from String tokens on URL

Demo

third_party/java_src/gwt/svn/trunk/samples/
expenses

Place

```
public class EditListPlace extends Place
{
    private String token;
    public EditListPlace(String token)
    {
        this.token = token;
    }
    public String getToken()
    {
        return token;
    }
    public static class Tokenizer implements PlaceTokenizer<EditListPlace>
    {
        public EditListPlace getPlace(String token)
        {
            return new EditListPlace(token);
        }
        public String getToken(EditListPlace place)
        {
            return place.getToken();
        }
    }
}
```

PlaceHistoryMapper

```
/**
 * PlaceHistoryMapper interface is used to attach all places which the
 * PlaceHistoryHandler should be aware of. This is done via the @WithTokenizers
 * annotation or by extending PlaceHistoryMapperWithFactory and creating a
 * separate TokenizerFactory.
 */
@WithTokenizers({ ListsPlace.Tokenizer.class, EditListPlace.Tokenizer.class })
public interface AppPlaceHistoryMapper extends PlaceHistoryMapper
{
}
```

Hello Places

```
// Start PlaceHistoryHandler with our PlaceHistoryMapper
PlaceHistoryMapper historyMapper = clientFactory.getHistoryMapper();
PlaceHistoryHandler historyHandler = new PlaceHistoryHandler(historyMapper);
historyHandler.register(placeController, eventBus, defaultPlace);
// Goes to place represented on URL or default place
historyHandler.handleCurrentHistory();
```

Places: Moving parts

EventBus

**Place
Controller**

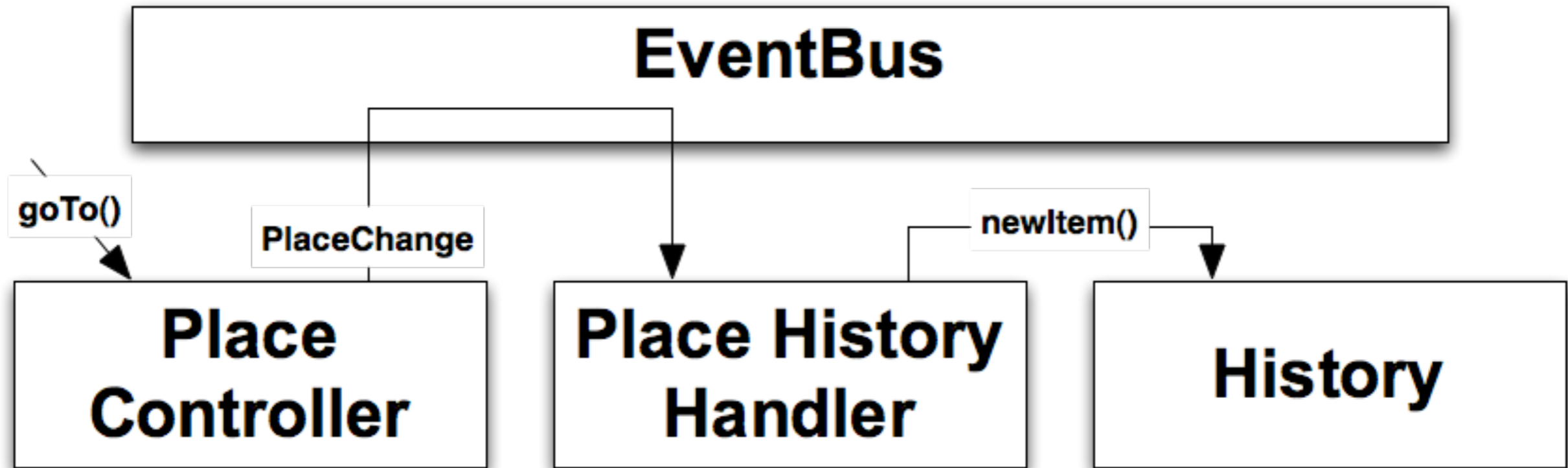
**Place History
Handler**

History

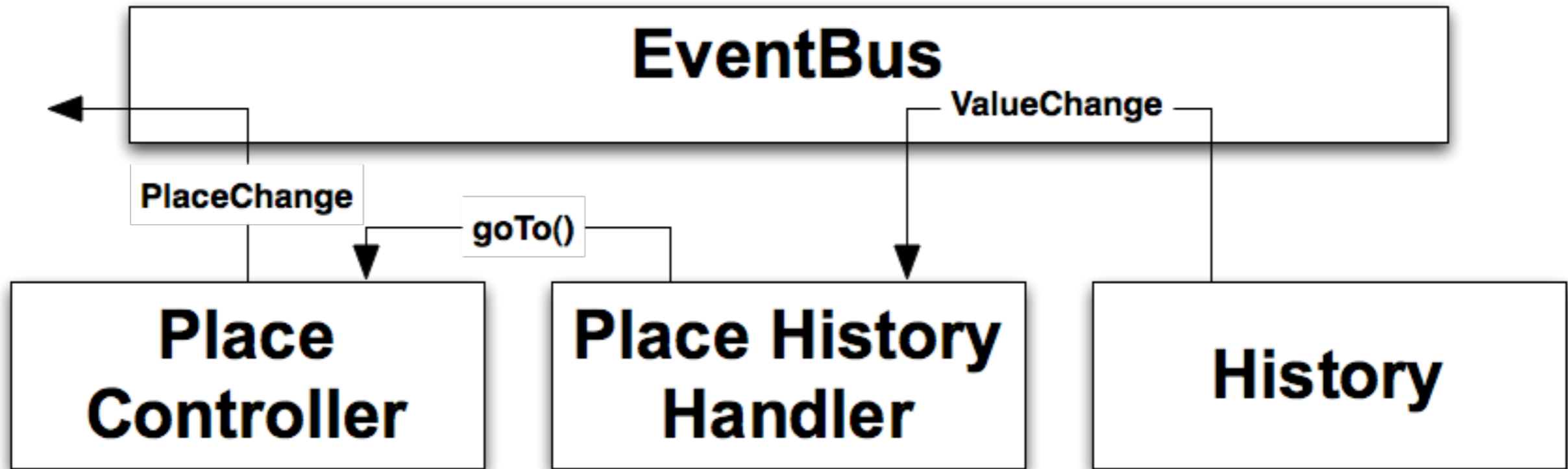
Your config here:
list of all place types



Places: Go to



Places: Back and forth



Activities

Activity, ActivityManager are not required!

ActivityManager

- Owns a region of the window
- Choose what to do with it on place change

Activity (“wake up, set up, show up”)

- Started by ActivityManager
- Provides a widget to display, asynchronously
- Can refuse to relinquish screen

Activity

```
public class EditListActivity extends AbstractActivity implements Presenter
{
    private ClientFactory clientFactory;
    private EventBus eventBus;

    public EditListActivity(ClientFactory cf, EditListPlace editListPlace)
    {
        this.clientFactory = cf;
        this.itemListToken = editListPlace.getToken();
    }

    @Override
    public void start(final AcceptsOneWidget panel, EventBus eventBus)
    {
        this.eventBus = eventBus;
        ...
        panel.setWidget(clientFactory.getEditListView());
    }
    ...
}
```

ActivityMapper

```
public class AppActivityMapper implements ActivityMapper {  
  
    private ClientFactory clientFactory;  
  
    public AppActivityMapper(ClientFactory clientFactory) {  
        super();  
        this.clientFactory = clientFactory;  
    }  
  
    @Override  
    public Activity getActivity(Place place) {  
        if (place instanceof EditListPlace) {  
            return new EditListActivity(clientFactory, (EditListPlace) place);  
        }  
        if (place instanceof ListsPlace)  
        {  
            return new ListsActivity(clientFactory);  
        }  
        return null;  
    }  
}
```

ActivityMapper Idioms

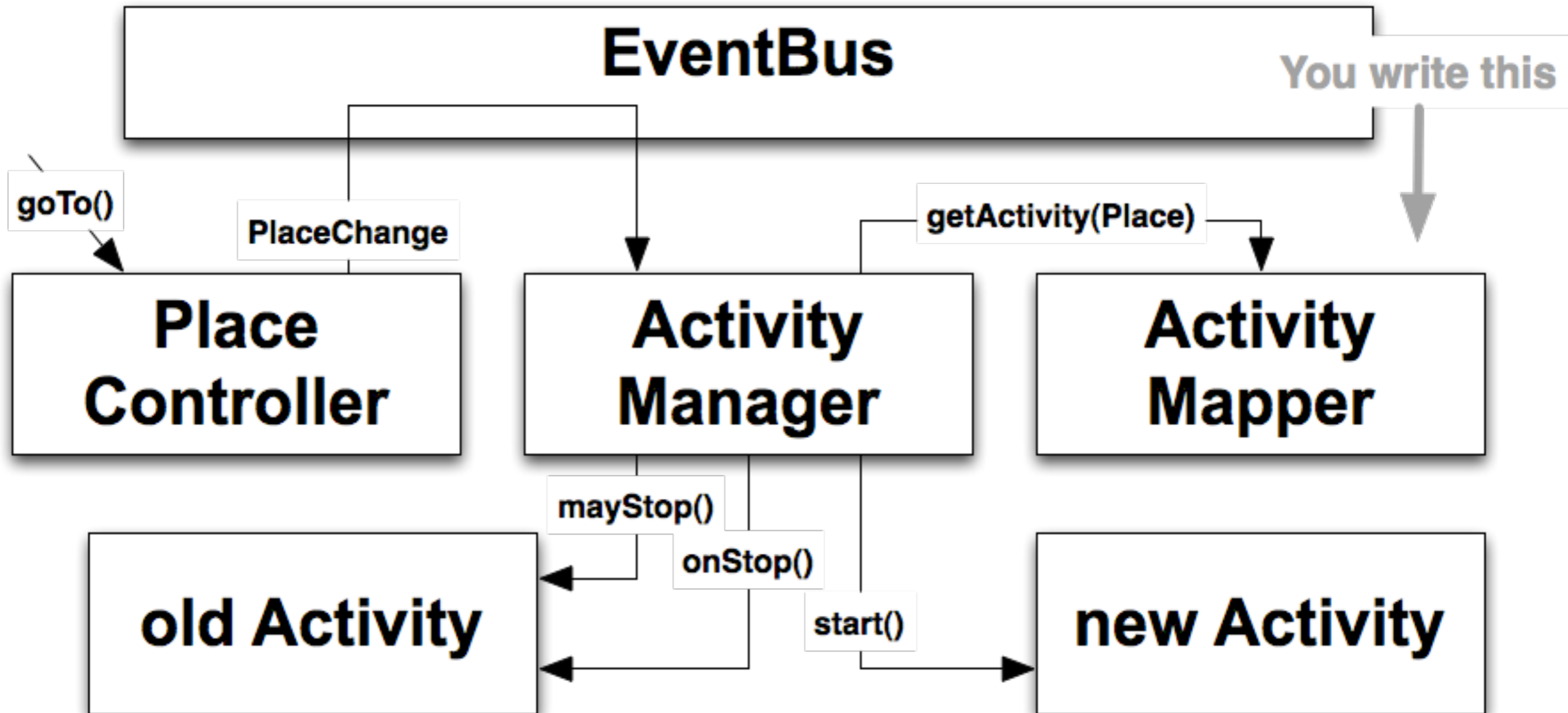
Disposable Activities, shared views:

```
if (place instanceof FooPlace) {  
    return new FooActivity(theOnlyFooView);  
}
```

Singleton:

```
if (place instanceof FooPlace) {  
    theOnlyFooActivity.update((FooPlace) place);  
    return theOnlyFooActivity;  
}
```

Get activated



Strategies

Widgets

Doo Dads

- Doo Dad Able
- Baker Doo Dad

Thingies

Gizmos

[Home](#) > [Doo Dads](#) > Doo Dad Able

Name	Doo	Dad	Ding	Dong
Doo Dad Able	34	The Goods	Peldi	<input checked="" type="checkbox"/>
Baker Doo Dad 4	18	The Guids	Pongi	<input type="checkbox"/>

Doo Dad Deets

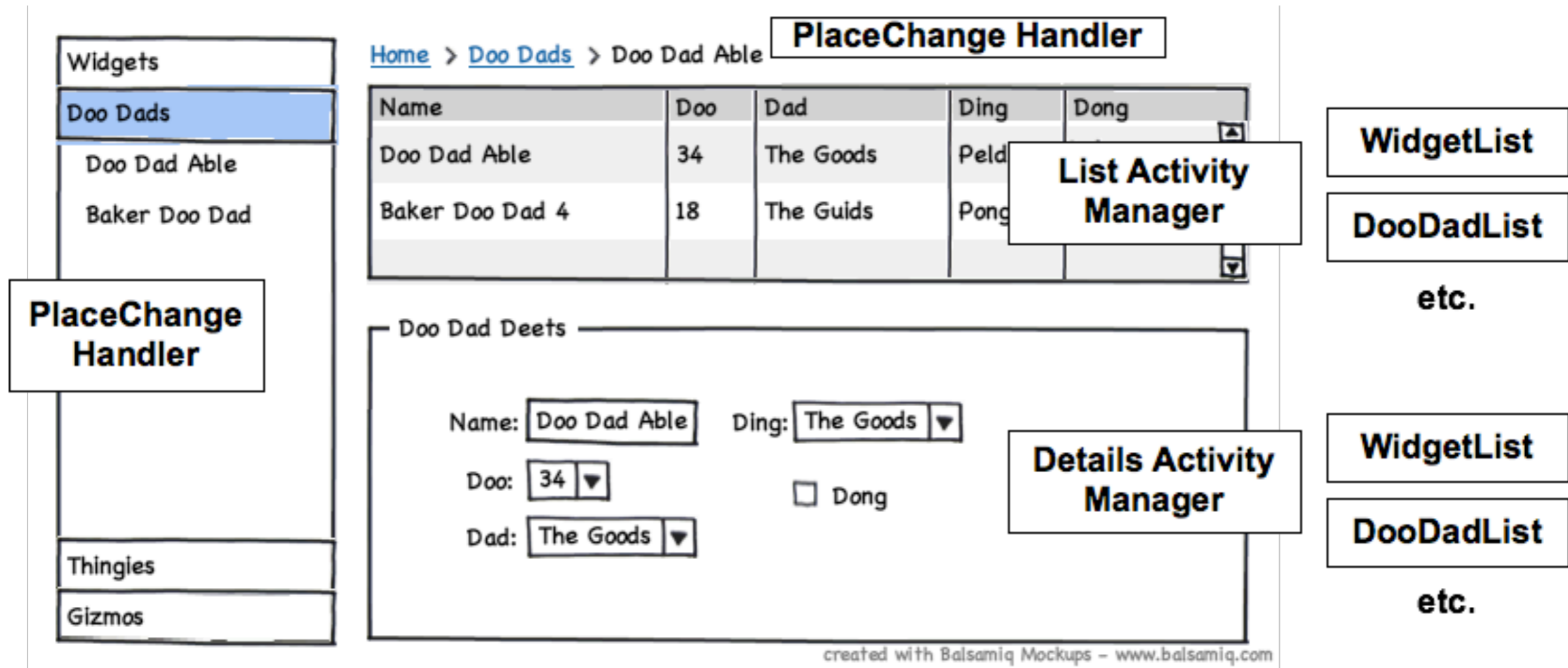
Name: Ding:

Doo: Dong

Dad:

created with Balsamiq Mockups - www.balsamiq.com

Strategies



Activities and Places wiring

```
// Start ActivityManager for the nav (west) panel with our WestActivityMapper
ActivityMapper westActivityMapper = new WestActivityMapper(clientFactory);
ActivityManager westActivityManager =
    new ActivityManager(westActivityMapper, eventBus);
westActivityManager.setDisplay(westPanel);

// Start ActivityManager for the main (center) panel with our CenterActivityMapper
ActivityMapper centerActivityMapper = new CenterActivityMapper(clientFactory);
ActivityManager centerActivityManager =
    new ActivityManager(centerActivityMapper, eventBus);
centerActivityManager.setDisplay(centerPanel);

// Start PlaceHistoryHandler with our PlaceHistoryMapper
PlaceHistoryMapper historyMapper = GWT.create(AppPlaceHistoryMapper.class);
PlaceHistoryHandler historyHandler = new PlaceHistoryHandler(historyMapper);
historyHandler.register(placeController, eventBus, defaultPlace);

RootPanel.get().add(dockLayoutPanel);
// Goes to place represented on URL or default place
historyHandler.handleCurrentHistory();
```

Cell Widgets

Fast, lightweight table rendering

Data binding with
DataProviders, ValueUpdaters

Scrolling and paging controls

Sortable columns, adjustable width

TextBox, IntegerBox, ValueBox<T>

Cell Widgets

```
// Create a CellTable.
CellTable<Contact> table = new CellTable<Contact>();

// Create name column.
@Override
TextColumn<Contact> nameColumn = new TextColumn<Contact>() {
    public String getValue(Contact contact) {
        return contact.name;
    }
};

// Create address column.
@Override
TextColumn<Contact> addressColumn = new TextColumn<Contact>() {
    public String getValue(Contact contact) {
        return contact.address;
    }
};

// Add the columns.
table.addColumn(nameColumn, "Name");
table.addColumn(addressColumn, "Address");
```

Editable Column

```
// Editable column for list name
nameColumn = new Column<ItemListProxy,String>(new EditTextCell())
{
    @Override
    public String getValue(ItemListProxy list)
    {
        return list.getName();
    }
};
```

Custom Column type

```
// Note Flyweight pattern: only one instance of HyperlinkCell passed to the Column
Column<ItemListProxy, Hyperlink> linkColumn =
    new Column<ItemListProxy, Hyperlink>(new HyperlinkCell())
{
    @Override
    public Hyperlink getValue(ItemListProxy list)
    {
        String proxyToken =
            clientFactory.getRequestFactory().getHistoryToken(list.stableId());
        String historyToken =
            clientFactory.getHistoryMapper().getToken(new EditListPlace(proxyToken));
        Hyperlink h = new Hyperlink(list.getName(), historyToken);
        return h;
    }
};
```

Populating a CellTable

```
public static class MyDataProvider extends
    AsyncDataProvider<ItemListProxy>
{
    @Override
    protected void onRangeChanged(HasData<ItemListProxy> display)
    {
        // To retrieve relations and value types, use .with()
        Request<List<ItemListProxy>> findAllReq = rf.itemListRequest()
            .listAll().with("owner");
        // Receiver specifies return type
        findAllReq.fire(new Receiver<List<ItemListProxy>>()
        {
            @Override
            public void onSuccess(List<ItemListProxy> response)
            {
                updateRowData(0, response);
            }
        });
    }
}
```

```
this.myDataProvider = new MyDataProvider(requestFactory);
```

Updating with a CellTable

```
public class NameFieldUpdater implements FieldUpdater<ItemListProxy, String>
{
    @Override
    public void update(int index, ItemListProxy obj, final String newName)
    {
        ItemListRequestContext reqCtx = clientFactory.getRequestFactory()
            .itemListRequest();
        ItemListProxy editable = reqCtx.edit(obj);
        editable.setName(newName);
        reqCtx.save(editable).fire(new Receiver<Void>()
        {
            @Override
            public void onSuccess(Void response)
            {
                EventBus.fireEvent(new MessageEvent(newName + " updated",
                    MessageType.INFO));
            }
        });
    }
};
```

```
// Make list name field editable
display.getNameColumn().setFieldUpdater(new NameFieldUpdater());
```

GWT Canvas

<http://gwtcanvasdemo.appspot.com>

GWT Logging

GWT now has `java.util.Logging` emulation

```
# In your .gwt.xml file
<inherits name="com.google.gwt.logging.Logging" />

# In your .java file
Logger logger = Logger.getLogger("NameOfYourLogger");
logger.log(Level.SEVERE, "this message should get logged");
```

Configure in your `.gwt.xml`

```
<set-property name="gwt.logging.logLevel" value="SEVERE" />
<set-property name="gwt.logging.enabled" value="FALSE" />
<set-property name="gwt.logging.consoleHandler" value="DISABLED" />
```

Easily enable remote logging, too

```
<set-property name="gwt.logging.simpleRemoteHandler" value="ENABLED" />
```

GWT speak

“Very productive environment for Java developers, but there is a learning curve”

“UI layout and styling is a challenge”

“GWT has enabled our team to run much faster than we could otherwise”

“Would be impossible to write our app without GWT”

New GWT book

<http://code.google.com/webtoolkit/books.html>



Thank you!

Where to get it:

<http://code.google.com/webtoolkit/>

Wiki, issue tracker, source:

<http://code.google.com/p/google-web-toolkit/>

Official GWT blog:

<http://google-web-toolkit.blogspot.com/>

Twitter:

[@googledevtools](https://twitter.com/googledevtools)

