Google Web Toolkit (GWT)

What is GWT?

- GWT is a development toolkit for building and optimizing complex browser-based applications
- You can develop all code, both client and server in Java (or with a different toolkit: Python)
    - The server side code could done in many languages
    - GWT is used for rich client apps

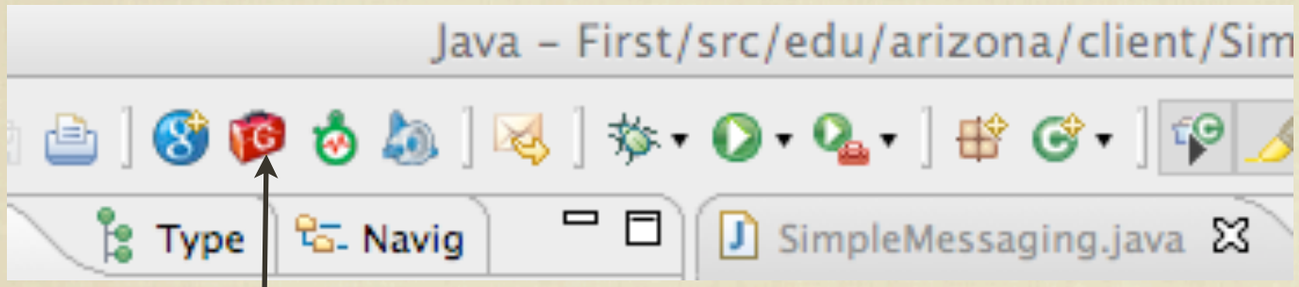
- The Java code is compiled to optimized JavaScript, which runs on
    1. any server
    2. mobile browsers for IPhone and Android
    3. any browser without worrying about the many browser quirks
        - Where developers waste tremendous amounts of time!

What is AJAX?

- The first Web: Any user change involved reload of the html page
  - Not user friendly--page disappears for a moment
- 1995: Java Applets allowed asynchronous loading of data after page displayed, others followed
  - Made possible with compiled client side
- 2005 AJAX: Asynchronous JavaScript and XML (but you don't need XML and need not be asynchronous as GWT is)
  - Client side techniques for developing rich client applications
  - Can retrieve data without interfering with the browser display
  - Puts more functionality into the browser
  - Allows interactivity usually associated with desktop applications

Eclipse Plugin

- Google has supplied an Eclipse plugin (installing take minutes)



- Create a new Web Application Project
  - `EntryPoint onModuleLoad` is where GWT begins, equivalent to
    - `main(String[])` in Java applications or `init()` in Java Applets

```
public class Ticketing implements EntryPoint {
  public void onModuleLoad() {
    // Layout the GUI
    // Register listeners (handlers)
  }
  // Handle events with classes that implement interfaces
}
```

Widgets and Layout

- Wdigets (swing uses `JButton JTextArea JLabel`)

```
private Button sendButton = new Button("Send");
private TextBox messageField = new TextBox();
private Label errorLabel = new Label();
```

- Interfaces (`ActionListener` with `actionPerformed(actionEvent)`)

```
// Create an event handler for sendButton and messageField (no private)
class MyHandler implements ClickHandler, KeyUpHandler {
  public void onClick(ClickEvent event) {
    sendMessageToServer();
  }

  public void onKeyUp(KeyUpEvent event) {
```

- RootPanel.get (like `JFrame.getContentPane()`) add (like `add`)

```
// Use RootPanel.get() to get the entire body element
RootPanel.get("messageContainer").add(messageField);
RootPanel.get("sendButtonContainer").add(sendButton);
```

Layout and listeners

- Can use html to layout `add`

```
FlowPanel p = new FlowPanel();
p.add(messageField);
p.add(sendButton);
p.add(errorLabel);
RootPanel.get().add(p);
// Focus the cursor on the message field when the app loads
messageField.setFocus(true);
```

- Register listeners `addActionListener(ActionListener)`

```
// Add a handler to send the name to the server
MyHandler handler = new MyHandler();
sendButton.addClickHandler(handler);
messageField.addKeyUpHandler(handler);
```

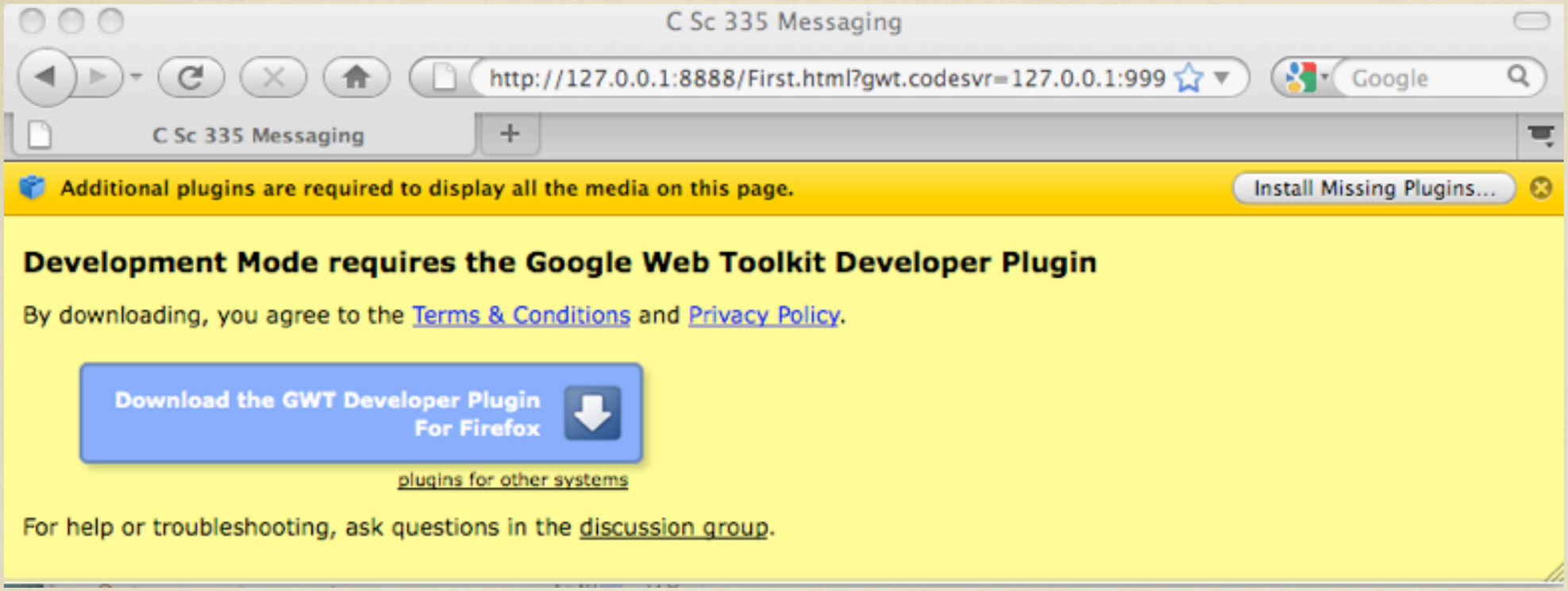
Layout Could also be HTML

```
<h1>C Sc 335 Messages</h1>

<table align="center">
  <tr>
    <td colspan="2" style="font-weight:bold;">Please enter your message:</td>
  </tr>
  <tr>
    <td id="messageContainer"></td>
    <td id="sendButtonContainer"></td>
  </tr>
  <tr>
    <td colspan="2" style="color:red;" id="errorLabelContainer"></td>
  </tr>
</table>
```

Run as a Web Application

- Use your favorite browser to develop (requires a GWT plugin)

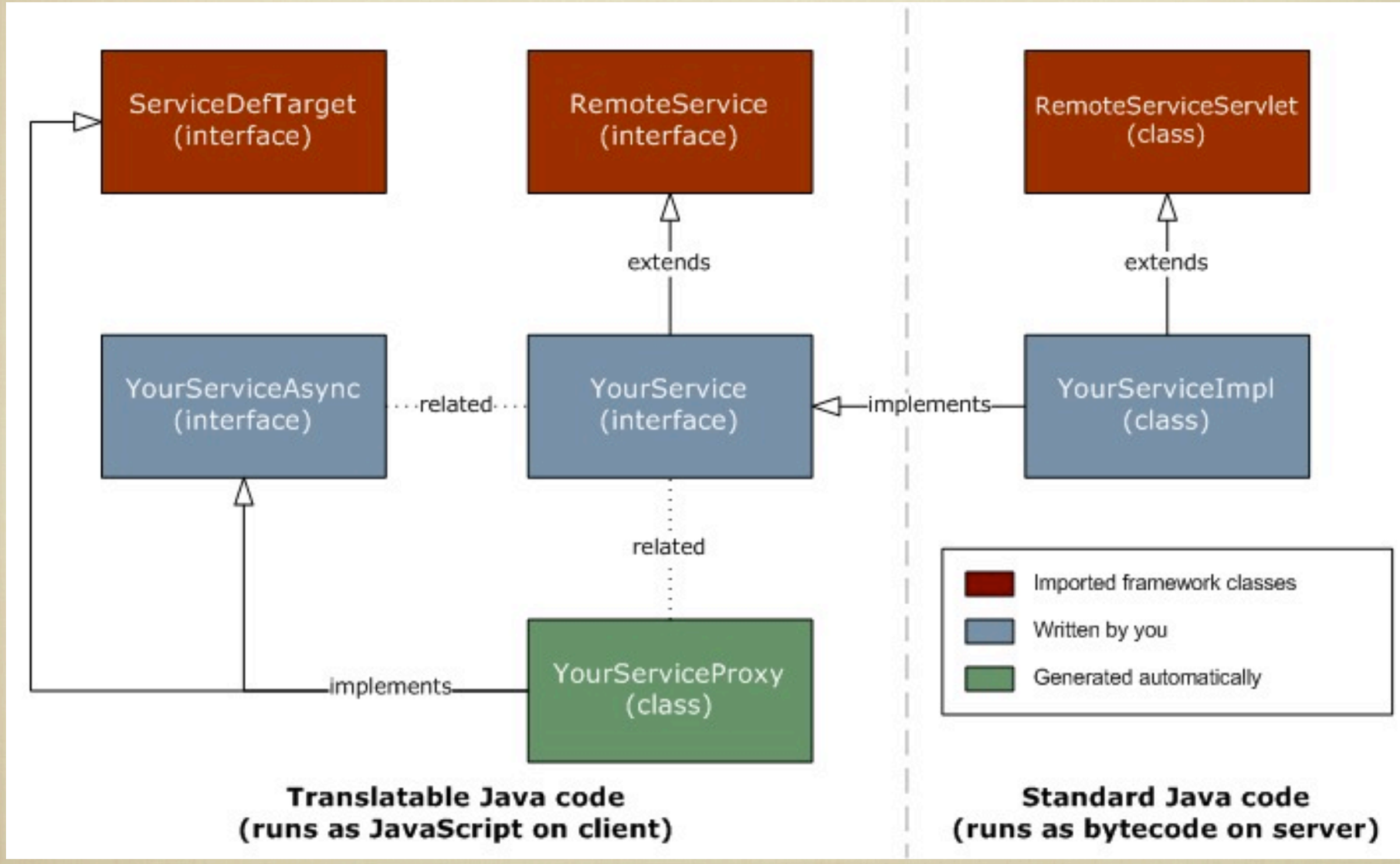


- Can change html pages and refresh and change code and reload
- Run project First from Eclipse to see

Remote Procedure Calls

- Remote procedure calls (RPC's) allow clients to get data from servers
- GWT provides its own RPC
  - The GWT client (the browser) can call server-side methods
  - Based on Java servlet technology
    - Objects can be sent between client and server
      - automatically serialized by the GWT framework
- Service: the server side servlet
- Invoking a service: The remote procedure call from the client

UML Diagram of ToDo



Creating a Service

- In order to define your RPC interface, you need to:

1) Define an interface for the service, `extend RemoteService` and list all RPC methods

```
// "message" must be defined as a service in the file web.xml
@RemoteServiceRelativePath("message")
   public interface MessageService extends RemoteService {
      String greetServer(String name) throws IllegalArgumentException;
}
```

2) Define a class to implement the server-side code that `extends RemoteServiceServlet` and implements the interface on the client side

```
public class GreetingServiceImpl extends RemoteServiceServlet
       implements MessageService {

Vector<String> messages = new Vector<String>();

public String greetServer(String input) throws IllegalArgumentException {
  messages.add(input);
  String result = "";
  int num = 1;
  for (String message : messages) {
    result = result + " " + num + ": " + message;
    num++;
  }
  return result;
```

Creating a Service continued

3) Define an asynchronous interface to your service to be called from the client-side code

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

Above goes in the client folder, Must have `AsyncCallback<String> callback` as the last argument

Then the client can call the server's greetServer method

```
// Anonymous inner class, textToServer os the String sent from client to server
greetingService.greetServer(textToServer, new AsyncCallback<String>() {
   public void onFailure(Throwable caught) {
      // Show the RPC error message to the user
      dialogBox.setText("Remote Procedure Call - Failure");
      // ...
    }
     public void onSuccess(String result) {
        dialogBox.setText("Remote Procedure Call");
        serverResponseLabel.setHTML(result);
        // ...
        closeButton.setFocus(true);
     }
  });
```

All local so far

- Have been using localhost as server
- Let's deploy
- Could have own server, buy space, or deploy into the cloud
  - We'll deploy into the cloud
  - Then see if you can load the app
  - Note: Neither system is persistent