Budapesti Műszaki és Gazdaságtudományi Egyetem

Java technológia

Google Web Toolkit 2.0

GWT céljai, működése nagy vonalakban

GWT BEVEZETÉS

Google Web Toolkit 2.0

- Google Java alapú AJAX toolkitje
- http://code.google.com/webtoolkit/
- Legfrissebb verzió: 2.4
  - Időhiány miatt csak a 2.0 elemeivel foglalkozunk
- Alapvetően egy Java → JavaScript compiler
  - Szerver és kliens oldal ugyanazon a nyelven íródhat
  - Szerver és kliens oldal **osztozhat** ugyanazon a kódon
  - Statikus típusellenőrzés a Java forrásnak hála
  - Kiemelkedő IDE és tooling támogatás, a Java nyelvnek köszönhetően

Mit is tud a GWT?

- Lehetővé teszi, hogy Java nyelven írjunk a **böngészőre** alkalmazásokat
  - **Nem Applet! Tiszta JavaScript-té fordítja a GWT compiler!**
- Közvetlenül, kedvenc debuggerünket, és kedvenc böngészőnket egyszerre használva kereshetünk hibákat a **developement mode-nak** köszönhetően
- Könnyedén készíthetünk böngésző, vagy nyelvfüggő alkalmazásokat, köszönhetően a **defferred binding** funkciónak
- Esztétikus felhasználói felületeket tervezhetünk köszönhetően a beépített **GWT widget**-eknek
- Fájdalommentesen kommunikálhatunk a szerverrel

Hogyan is történik mindez?

- Az általunk (Java-ban) írt kliens oldali kód kettős életet él
  - Amikor **developement mode-ban** indítjuk az alkalmazásunkat akkor az Java kódként fut, és a célböngészővel egy speciális API-n keresztül kommunikál. Ilyenkor a javascript működést a rendszer **emulálja**
  - **Ennek következménye, hogy developement mode-ban használhatjuk a fejlesztői környezetünk debuggerét és egyéb eszközeit (code coverage, JUnit tesztek, stb.), miközben egy böngésző refresh elég, hogy lássuk a módosításokat**
  - Amikor a kódot **lefordítjuk**, akkor abból tényleges JavaScript lesz, a standard Java könyvtárak helyett pedig a GWT által szolgáltatott JavaScriptben íródott osztályok fognak behelyettesítődni
  - Fordításkor a kódunk összes verziója (minden támogatott böngészőtípus, minden honosított verzió) minden kombinációban lefordításra kerül. Pl: Opera-HU, Opera-EN, IE-HU, IE-EN, FireFox-HU, FireFox-En
  - Ez **a kódunkban nem látszik**, mert a **GWT.create()** statikus metódushívások helyén a fordító a megfelelő behelyettesítést fogja alkalmazni. **Ezt hívjuk deferred bindingnak.**

Az alap Hello World alkalmazás GWT-ben

GWT A GYAKORLATBAN 1.

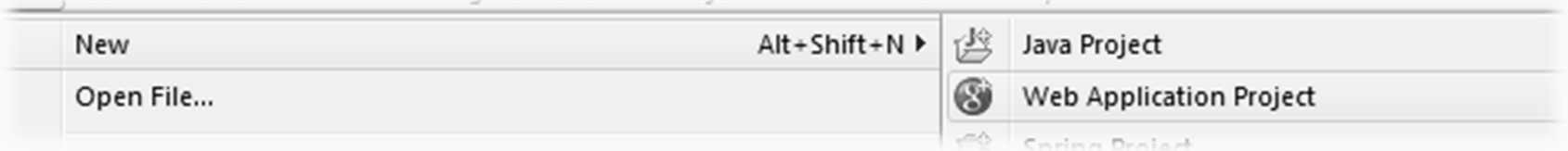
GWT Project létrehozása, kezelése

- Alap parancssori eszköz: webAppCreator
  - Ezzel itt nem foglalkozunk

- Hivatalos **Google Eclipse Plugin**
  - Ezzel fogunk dolgozni
  - Letölthető innen: http://code.google.com/eclipse/index.html

GWT Project létrehozása, kezelése 2.

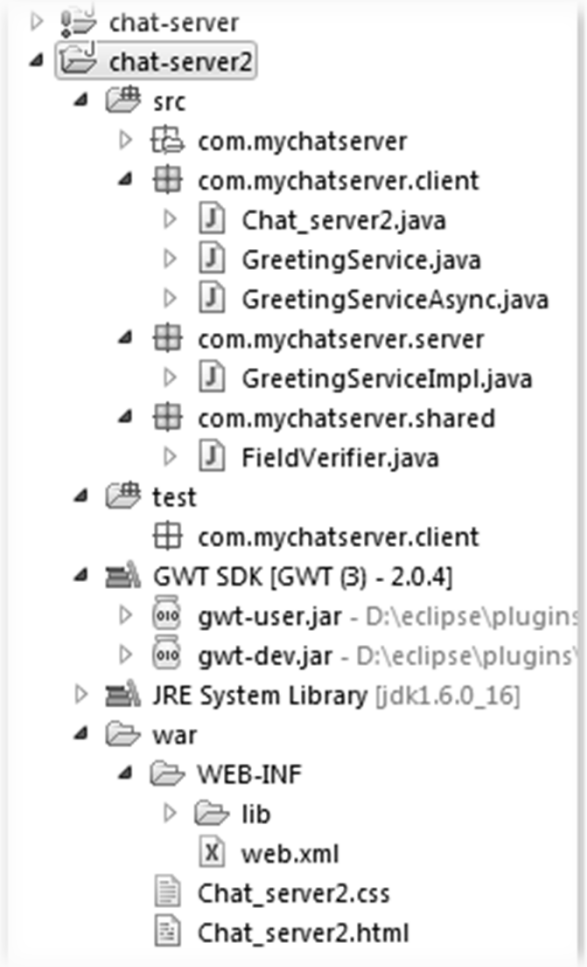
- Google Plugin segítségével Eclipse-ben egyszerű varázslóval



- **Lásd Demo**

Project szerkezet

- Az alap Hello World project
- **src** könyvtár
  - **com.mychatserver** – modul package
  - **Chat_server2.gwt.xml –** modul deszkriptor
  - **com.mychatserver.client** – kliensoldali kód
  - **com.mychatserver.server** – szerveroldali kód
  - **com.mychatserver.shared** – kód amit a kliens és szerveroldal is használ
- **test** könyvtár
  - Unit tesztelés ide jön
- **war** könyvtár
  - **Chat_server2.html** – html hoszt oldal
  - **WEB-INF** – standard war file szerkezet



GWT Project elemei – HTML Host Page

- A példánkban **Chat_server2.html**
- A hoszt HTML állomány indítja a GWT alkalmazást a megfelelő JavaScript fájl betöltésével:

```
<script type="text/javascript" language="javascript"
src="chat_server2/chat_server2.nocache.js"></script>
```

- A következő sor a Back-Button (Vissza-Gomb) támogatáshoz szükséges:

```
<iframe src="javascript:''" id="__gwt_historyFrame" tabIndex='-1'
style="position:absolute;width:0;height:0;border:0"></iframe>
```

- A HTML több elemet alapértelmezetten nem tartalmaz, **minden GUI elem dinamikusan a JavaScript-et futtatva jön létre**

GWT Project elemei – Module XML

- **A példánkban Chat_server2.gwt.xml**
- Minden GWT alkalmazás modulokból épül fel
- Modulok **örökölhetnek** más modulokat, így elérhetik azok funkcionalitását
  - Hasonlóan a programozói könyvtárakhoz
- Miért van rá szükség? Több funkciót ad az alap Java package-eken felül
  - Egy JavaScript file-nak csak egyetlen belépési pontja van, azt itt kell megadni
  - Itt kell megadni a deferred binding behelyettesítési szabályait
  - Használt stíluslapokat és külső JavaScript könyvtárakat itt lehet megadni

GWT Project elemei – Module XML 2.

- A legtöbb GWT Modul így kezdődik:

```
<inherits name='com.google.gwt.user.User'/>
```

  - A fenti sor betölti a GWT alap felhasználói könyvtárát
- Egy alkalmazásnál meg kell adni a belépési pontot (**Entry point**) is:

```
<entry-point class='com.mychatserver.client.Chat_server2'/>
```

- Végül meg kell adni azoknak az al-package-eknek a listáját, melyeket JavaScript kóddá (is) akarunk fordítani:

```
<source path='client'/>
<source path='shared'/>
```

GWT Project elemei – Entry Point osztály

- A példánkban a **Chat_server2.java** file
- A belépési pont mindig implementálja az **EntryPoint** interfészt:

```
public class Chat_server2 implements EntryPoint {
  public void onModuleLoad() {
    …
  }
}
```

- Az interfész egyetlen megvalósítandó metódusa az **onModuleLoad(),** mely a programunk betöltődésekor hajtódik végre
  - Hasonló a **public static void main ()** metódushoz hagyományos Java alkalmazások esetén

GWT Project elemei – Entry Point osztály 2.

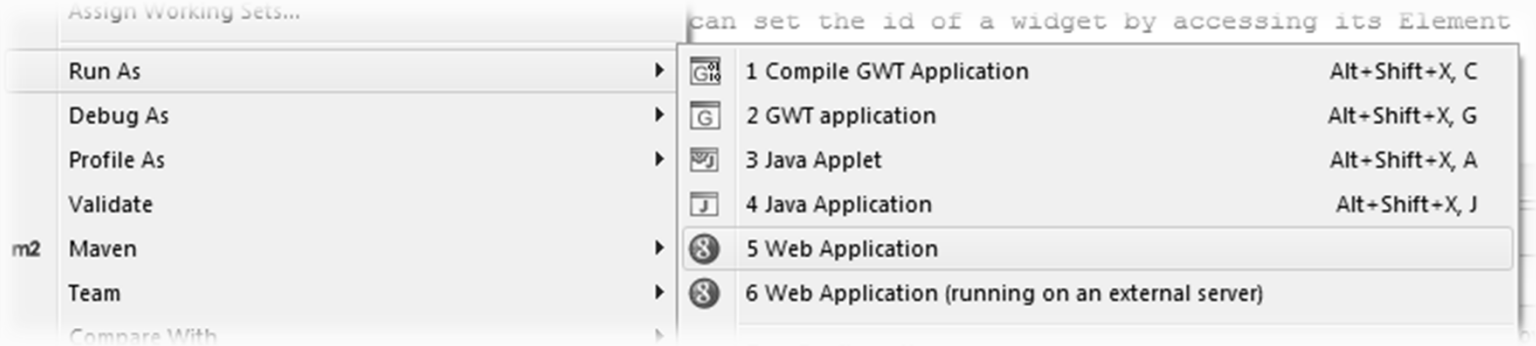
- Ahhoz, hogy GUI elemeket tudjunk az oldalra tenni, el kell kérni egy HTML elemet amibe azokat helyezni akarjuk
  - Ez a **RootPanel** osztály **get()** illetve **get(String elementId)** metódusaival lehet
  - A sima **get()** a BODY elemet adja vissza, a paraméterezett változat az adott id attribútumú elemet
- A Hello World példa Host HTML-jének részlete, illetve a kód ami az elemeket elkéri:

```
<tr>
  <td id="nameFieldContainer"></td>
  <td id="sendButtonContainer"></td>
</tr>
```

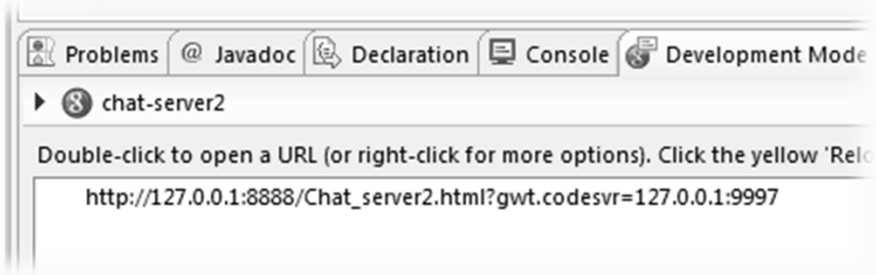
```
RootPanel.get("nameFieldContainer").add(nameField);
RootPanel.get("sendButtonContainer").add(sendButton);
```

Hello World futtatása

- **Developement mode**-ban futtatni az alkalmazást a **Run As** menüből (is) lehet:



- A Developement Mode fül mutatja az URL-t amin keresztül az alkalmazás elérhető böngészőből:



Hello World futtatása

- **Figyelem! Developement Mode-ban az alkalmazásunk NEM JavaScript-ként fut, hanem Java kódként ami a böngészővel kommunikál JS kódot imitálva!**
  - **Következmény:** Hagyományos Eclipse debuggerrel vizsgálhatjuk a kódot!
  - **Breakpoint-ok, Watch kifejezések működnek!**
- A Developement Mode fülön továbbá a log-ot is látjuk, ebben fogjuk kapni a hibákat is!

Chat alkalmazás írása a Hello World-ből kiindulva

GWT A GYAKORLATBAN 2.

Hello World fölösleges részek eltávolítása

- A Host HTML file BODY elemét átírjuk:

```
<body>
  <iframe src="javascript:''" id="__gwt_historyFrame" tabIndex='-1'
  style="position:absolute;width:0;height:0;border:0"></iframe>
</body>
```

- Az EntryPoint osztályunkból kivesszük a fölösleges részeket:

```
public class Chat_server2 implements EntryPoint {

  public void onModuleLoad() {
    RootPanel.get().add(new HTML("<b>Chat comes here!</b>"));
  }

}
```

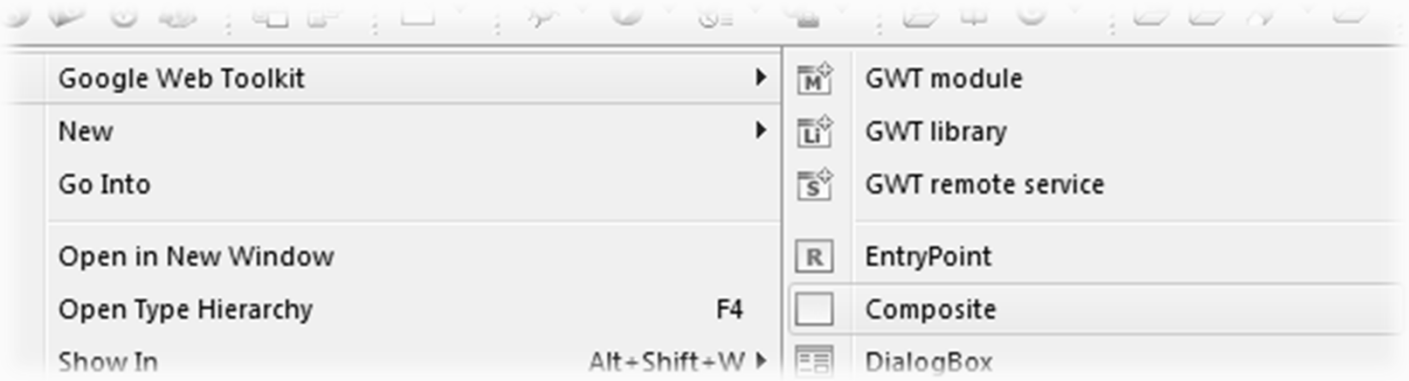
- **Browser Refresh és már működik is!**

Google WindowBuilder Pro

- Eredetileg az Instantiations cég által fejlesztett GUI tervező eszköz
  - Eclipse plugin
  - Swing, SWT és GWT felületek szerkeszthetők benne
  - 2010-ben a Google az egész céget megvásárolta és a plugineket ingyenesen elérhetővé tette
- Letölthető innen: http://code.google.com/javadevtools/download-wbpro.html

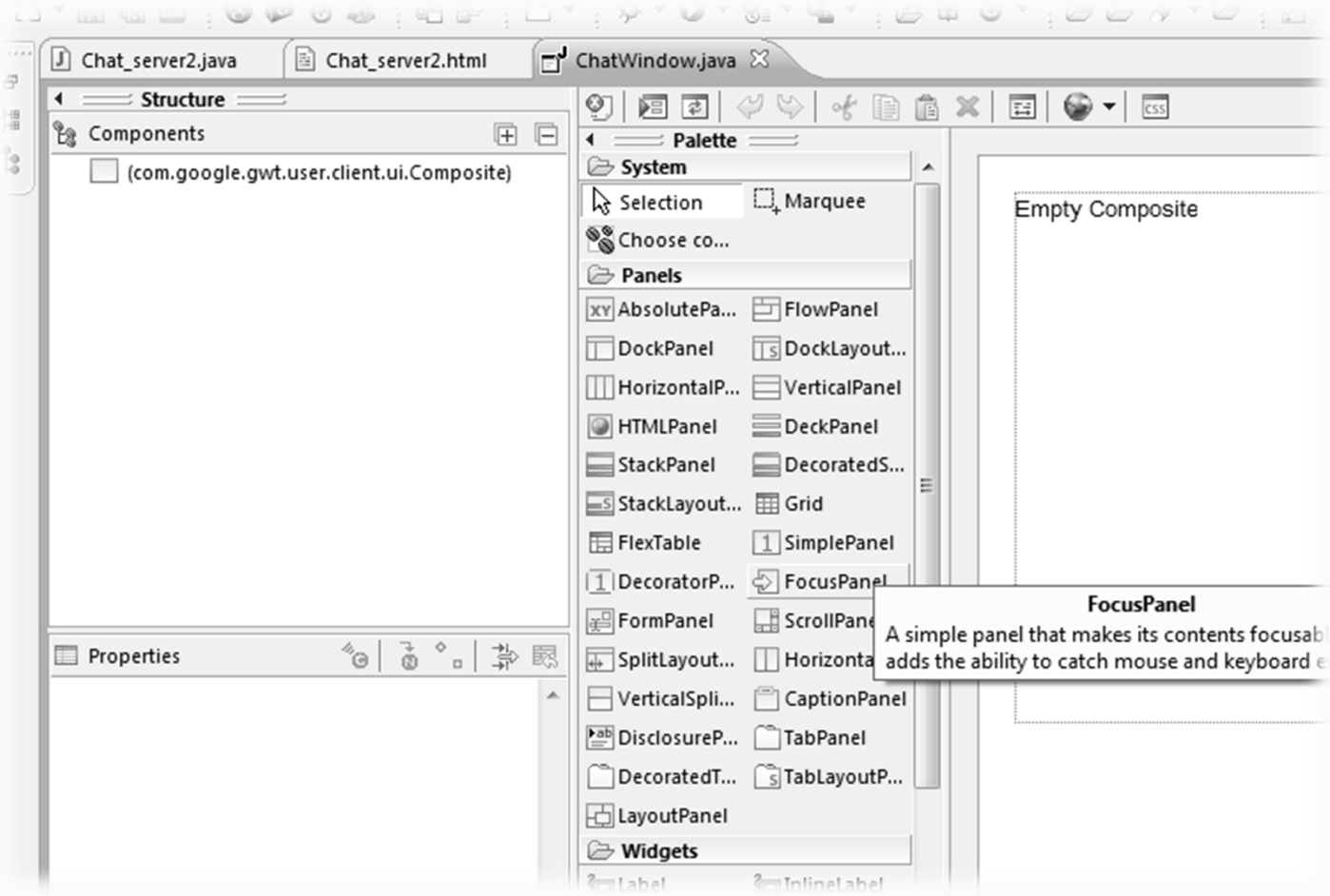
ChatWindow tervezése

- Új **Composite** objektum létrehozása:

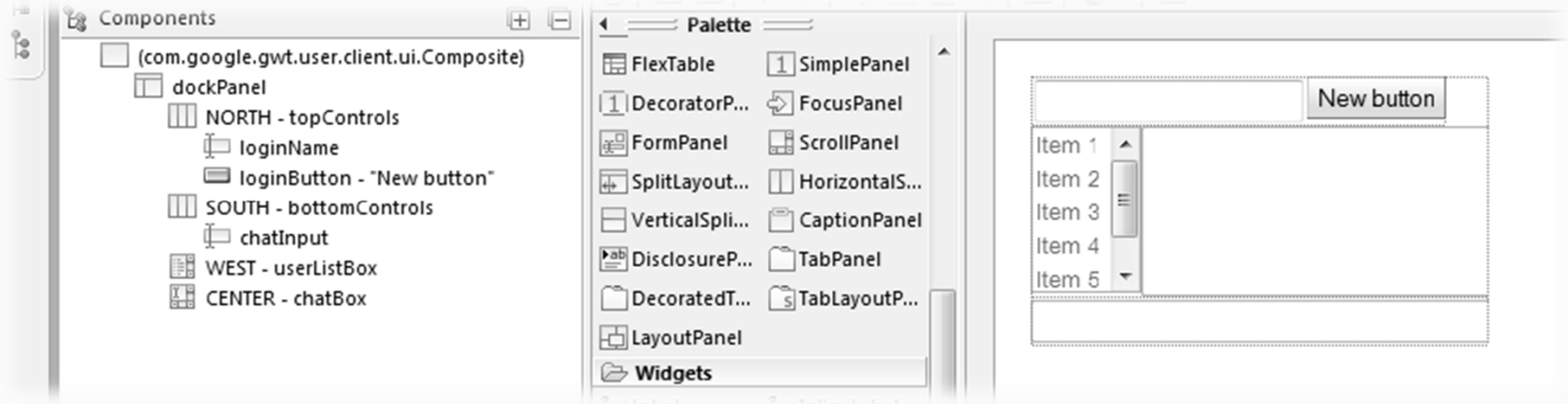


- A Composite objektumok valójában widgetek, amelyeket több widgetből rakunk össze

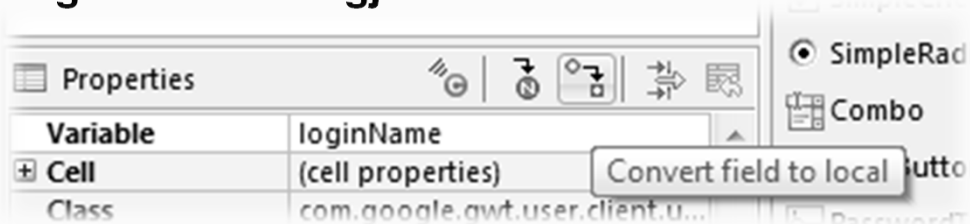
ChatWindow tervezése 2. (WindowBuilder ablak)



ChatWindow tervezése 3. (hozzávetőleges kinézet)



- **Trükk: a GUI elemeiből *mezőt* lehet csinálni egyetlen gombnyomással!**
  - Sőt, a szerkesztő elég okos, hogy **a kézzel átvariált kódot is képes grafikusan megjeleníteni**



ChatWindow eseményei

- A Chat ablakunk négy eseményt fog kezelni:
  - bejelentkezés
  - kijelentkezés
  - üzenet küldés
  - adatok frissítése

```
private void doLogin()
private void doLogout()
private void sendMessage()
private void refreshData()                                    }
```

ChatWindow eseményei 2.

- Példa grafikai elemek vezérlésére – Login/Logout gomb működése:

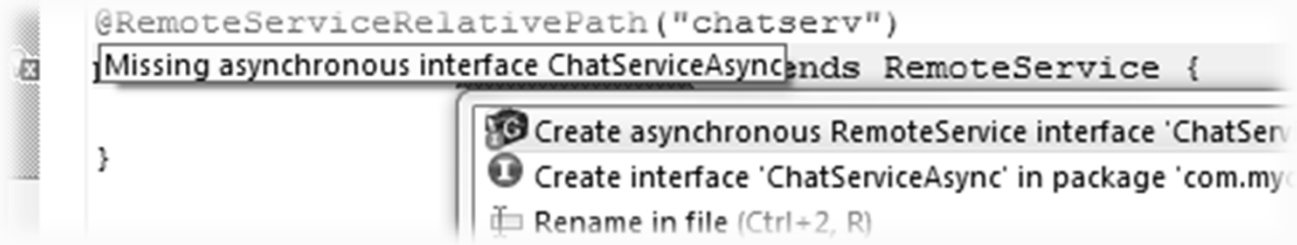
```
loginButton.addClickHandler(new ClickHandler() {
  public void onClick(ClickEvent event) {

    // on click do a logout or login depending on current status
    if (!loggedIn) { doLogin(); } else { doLogout(); }

  }
});

…

private void doLogin() {
  // TODO: COMMUNICATE WITH SERVER
  loggedIn = true;
  chatBox.setEnabled(true);
  chatInput.setEnabled(true);
  userListBox.setEnabled(true);
  loginName.setEnabled(false);
  loginButton.setText("Logout");
}
```

Kommunikáció a szerverrel – GWT RPC

- Szükségünk van egy interfészre ami a szerver által nyújtott szolgáltatásokat definiálja:

```
@RemoteServiceRelativePath("chatserv")
public interface ChatService extends RemoteService { … }              }
```

- Ezek után a GWT plugin QuickFix segítségével létrehozzuk annak aszinkron párját:



Kommunikáció a szerverrel 2. – GWT RPC Servlet

- Önmagában az interfészekkel még nem tudunk mit kezdeni, kell egy megvalósítás:

```
public class ChatServiceImpl extends RemoteServiceServlet implements
ChatService  { … }                                                    }
```

- A szervletet a web.xml file-ban a servleteknél tanultak alapján egy URL-hez kell rendelni
  - Mivel a `@RemoteServiceRelativePath("chatserv")` annotációval a **ChatService** interfészben a **chatserv** elérési útvonalat adtuk meg, a servlet-et ide kell mappelni
- **Most már csak a chat funkciót kell megírni!**
  - A szerveroldali kódot itt nem fogjuk megtárgyalni, mivel egyszerű Java-ról van szó

Kommunikáció a szerverrel 3. – RPC szolgáltatás elérése

- A **GWT.create()** metódusával tudunk kliens oldalon az **aszinkron interfészhez** elkérni a tényleges szolgáltatást:

```
ChatServiceAsync chatService = GWT.create(ChatService.class);
```

- **Kliens oldalon csak az aszinkron interfész érhető el!**
- Innentől kezdve a szolgáltatás anonim belső osztályok segítségével használható:

```
chatService.login(userName, new AsyncCallback<Void>() {
  public void onSuccess(Void result) {…}
  public void onFailure(Throwable caught) {…}
});
```

- A fenti példában egy **void** visszatérésű függvényt hívtunk, így a **result** paraméter a callback-ben természetesen üres

Kommunikáció a szerverrel 4. – RPC szolgáltatás elérése

- A teljes átírt doLogin() metódus:

```
chatService.login(userName, new AsyncCallback<Void>() {
  public void onSuccess(Void result) {
    loggedIn = true;
    chatBox.setEnabled(true);
    chatInput.setEnabled(true);
    userListBox.setEnabled(true);
    loginName.setEnabled(false);
    loginButton.setText("Logout");
  }
  public void onFailure(Throwable caught) {
    Window.alert("Could not login to server!");
  }
});
```

Kommunikáció a szerverrel 4. – Chat ablak frissítése

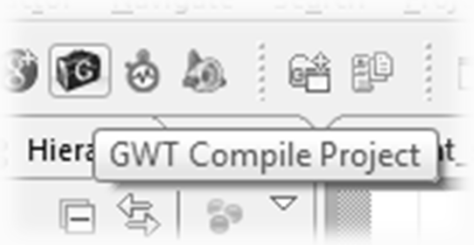
- A Chat ablakot egyszerűen periodikusan fogjuk frissíteni, időről időre lekérdezve a szervertől annak állapotát:

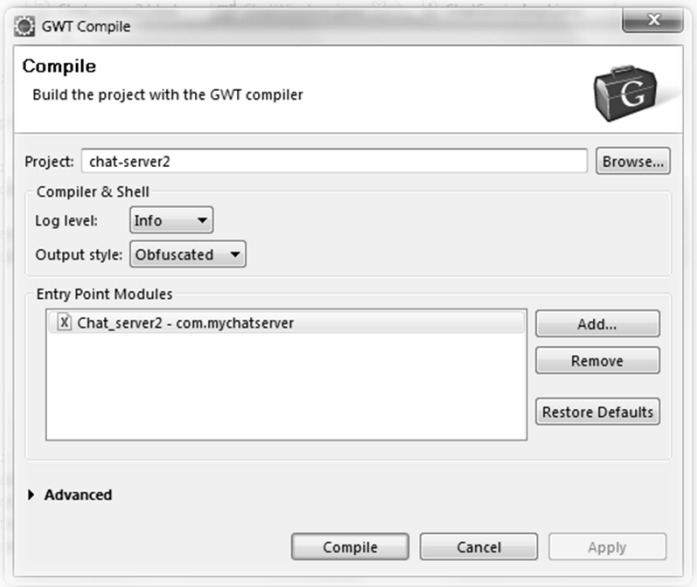
```
Timer refreshTimer = new Timer() {
  public void run() {
    refreshData();
  }
};
refreshTimer.scheduleRepeating(500);
```

- A GWT **Timer** osztályát használtuk ehhez
- a **refreshData** metódus letölti a userek listáját és a chat history-t
  - Egyáltalán nem hatékony, de ez csak egy demonstráció

Végül: fordítás JavaScript kódra

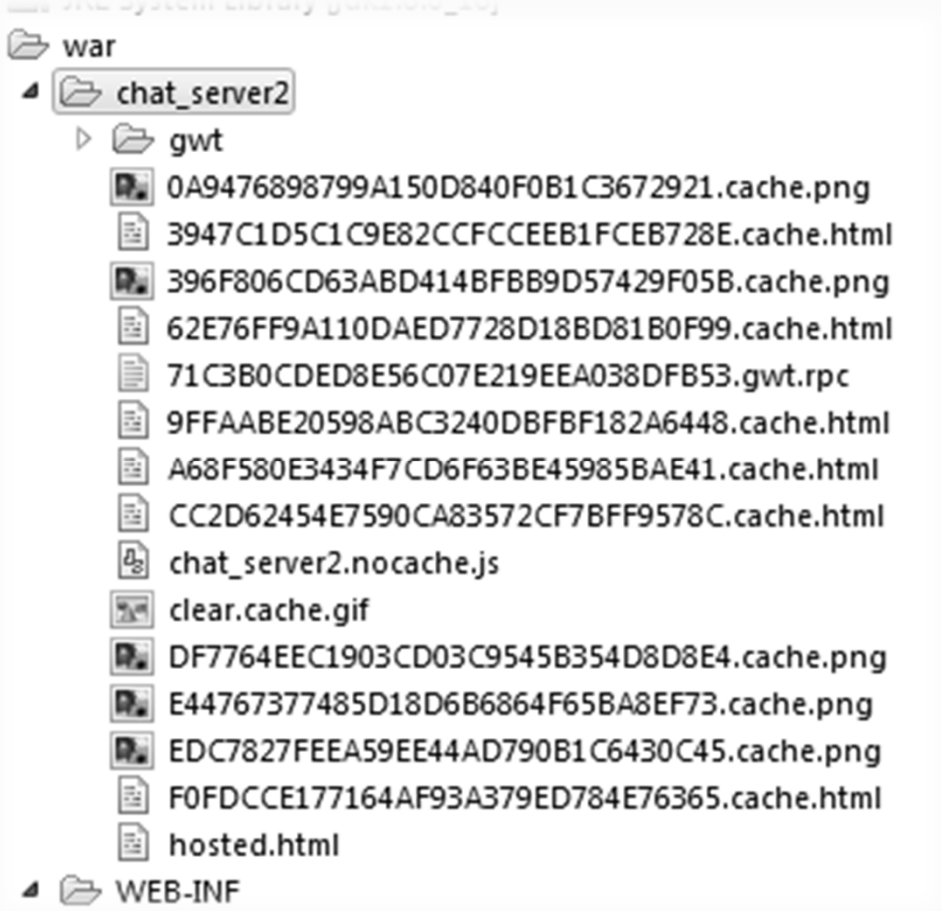
- Compile ablak behozása:
- Beállítható hogy milyen kód készüljön:
  - **Obfuscated** – „elcsúfított”
  - **Pretty** – jól olvasható
  - **Detailed** – részletezett





A lefordított Chat alkalmazás

- A lefordított file-ok a példánkban a **war/chat_server2** könyvtárba kerültek
- **Minden *.cache.* állomány az örökkévalóságig cache-elhető!**
- Amikor új verziót fordítunk, akkor az állományok neve megváltozik, így biztosan nem cache-ből tölti be a felhasználó (a név a tartalom MD5 hash-e)
- Minden böngésző/nyelv variációra külön verzió, ún. **permutáció** készül



A Chat alkalmazás hiányosságai

- Rugalmatlan API
- A periodikus frissítés nagyon rossz hatékonyságú
- Biztonsági problémák tömege
- Nem kezeljük a hibákat
- GUI túl egyszerű