



TIBET

Bringing the Web
Back Into Balance.



What is TIBET?

- TIBET is a complete JavaScript application library
- TIBET adds significant functionality to the built-in JavaScript language & library
- TIBET is fully compliant with ECMAScript edition 2 (JavaScript 1.2/1.3)
- TIBET is cross-platform: IE4, 5, 5.5, Nav 4.6+, Nav 6.0+

What is TIBET?

The screenshot displays the TIBET Web Desktop interface. At the top, there is a navigation bar with "Home | Back | Forward" and "Technical Pursuit Inc. - TIBET Web Desktop". Below this is a breadcrumb trail: "Classes > TPReallyLong... > Methods". The main area is divided into several panels:

- Classes Panel:** A tree view showing a hierarchy of classes. The selected path is "Classes > TPReallyLong... > Methods".
- Methods Panel:** A list of methods for the selected class. The selected method is "fromArray".
- Locator Panel:** Contains the text "Not Yet Implemented. In a future version this panel will allow you to enter a search phrase and the NavPath will navigate to that object in the resource tree to the left."
- Content Panel:** Displays the source code for the selected method. The code is as follows:

```
TPDictionary.addTypeMethod('fromArray',
function(anArray)
{
    /**
     * Returns a new dictionary whose key/value pairs match for each
     * element in the array provided. This method is useful when
     * searching iteratively over an array for a collection of items since
     * it significantly improves lookup performance.
     */
    assert(isJSArray(anArray), InvalidArrayException);

    var i;
    var newInst = this.create();
    for (i=0; i<anArray.getSize(); i++)
    {
        newInst.atPut(anArray.at(i), anArray.at(i));
    }
    return newInst;
});
```
- Assistant Panel:** Contains the text "Current Task: Edit A Method" and a numbered list of instructions:
 1. Select method using the NavPath. Starting from the root, select TIBET -> Classes -> Classname. Then select the individual method.
 2. Using the Content window, edit the source code text using the text area displayed. If the method source is not displayed in a text area the method is read-only.
 3. If you've been able to edit, select "Save" to make the new version of the method active. Use the "Commit" option to send the changes you've made back to the TIBET Repository.

At the bottom of the interface, there is a status bar with "Local machine zone" and a "Local machine zone" icon.

Why TIBET?

- Enormous amount of untapped processing power on client machines running Web browsers
 - PCs incredibly cheap – incredibly powerful.
- Current Web architectures are extremely inefficient of bandwidth and server resources
- Users can be provided with a MUCH better user experience
- Bring balance back to the Web

Why TIBET?

- Whatever happened to client-server?
 - The Web
- Advantages to Centralized Model
 - Single Platform Target
 - Ease of Application Deployment
 - Ease of Software Maintenance
 - Centralized Point of Control
 - Inexpensive Client Machine

Why TIBET?

- But client-server was such a good idea!
- Advantages to Client-Server Model
 - Better performance
 - Improved scalability
 - Enhanced fault tolerance
 - Better user interface
 - Client-side application integration

TIBET – Infrastructure for Browser-based applications

- Most existing JavaScript libraries have taken a ‘UI down’ approach.
 - Many dhtml libraries and ‘widget’ kits available
 - Most libraries are developed independently of each other and don’t form an integrated whole
 - XUL/XBL is impressive, but more is needed
- TIBET takes a ‘core up’ approach
 - Much of what TIBET provides hasn’t been available
 - TIBET’s frameworks integrate tightly and have well documented API for extension

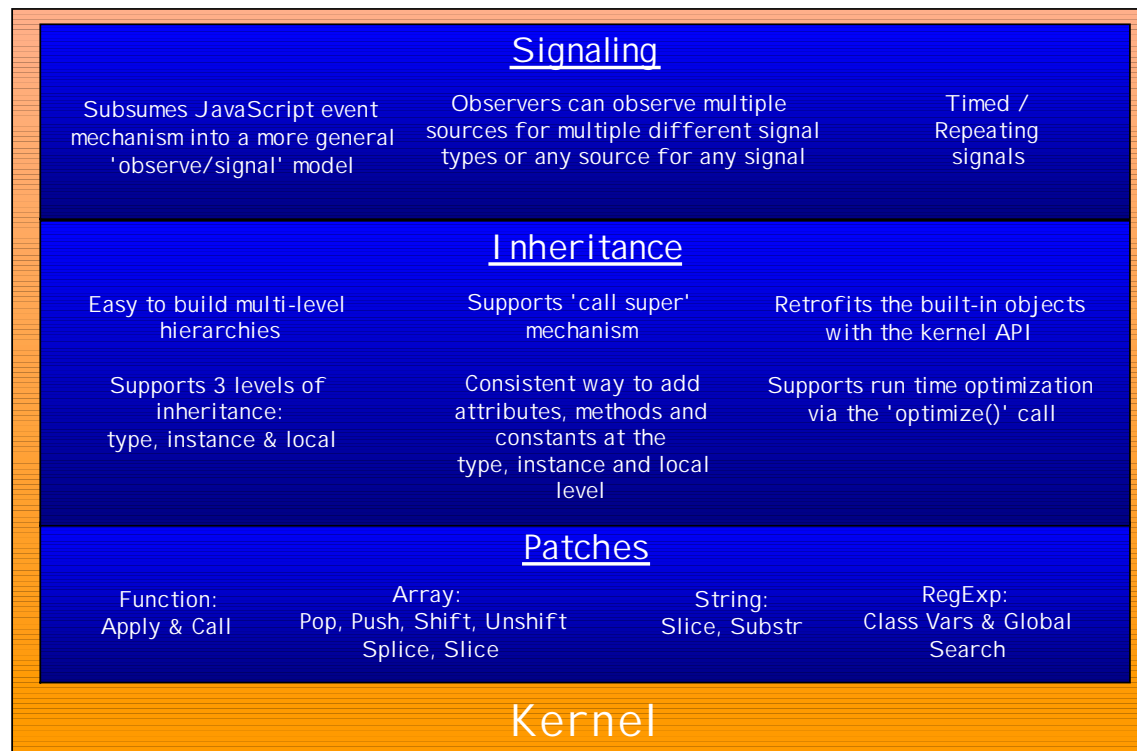
Components of TIBET

Localization	Data cache	Active Client Pages
Formatting / Validation	Server Comm.	UI Framework
Collections		
Kernel		

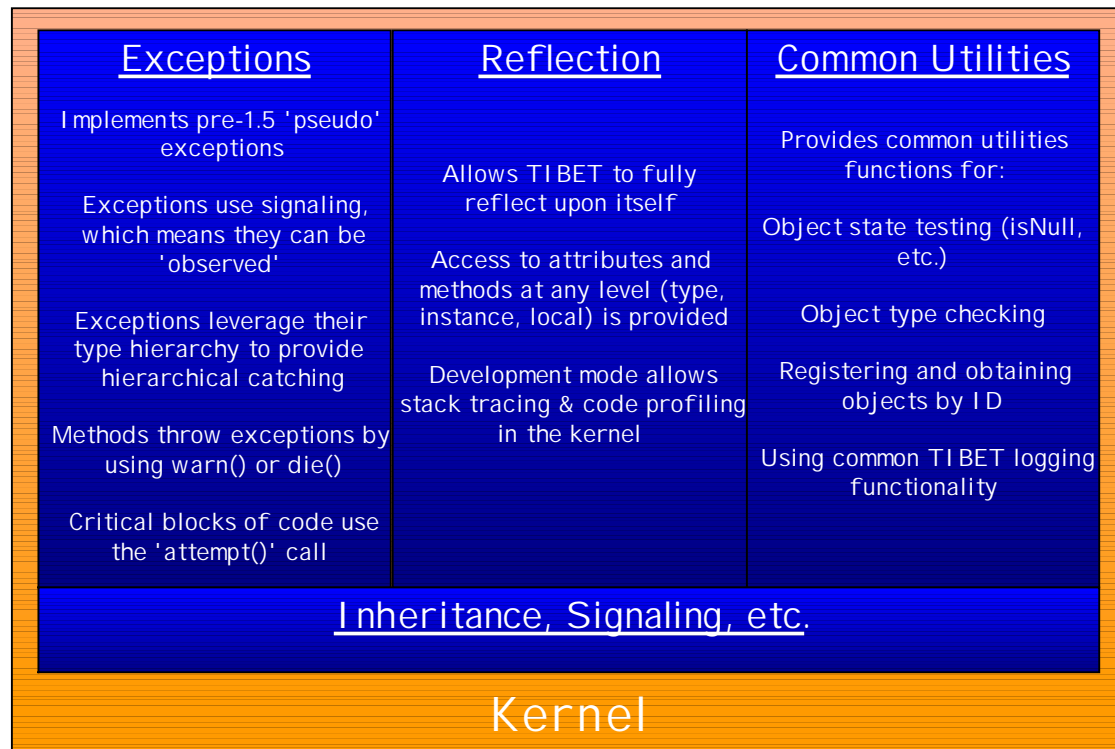
TIBET Kernel

- Provides the core facilities to do substantial application development in a Web browser
 - Patches
 - Inheritance
 - Signaling
 - Exceptions
 - Reflection
 - Common Utilities

Kernel Components – Pt. 1



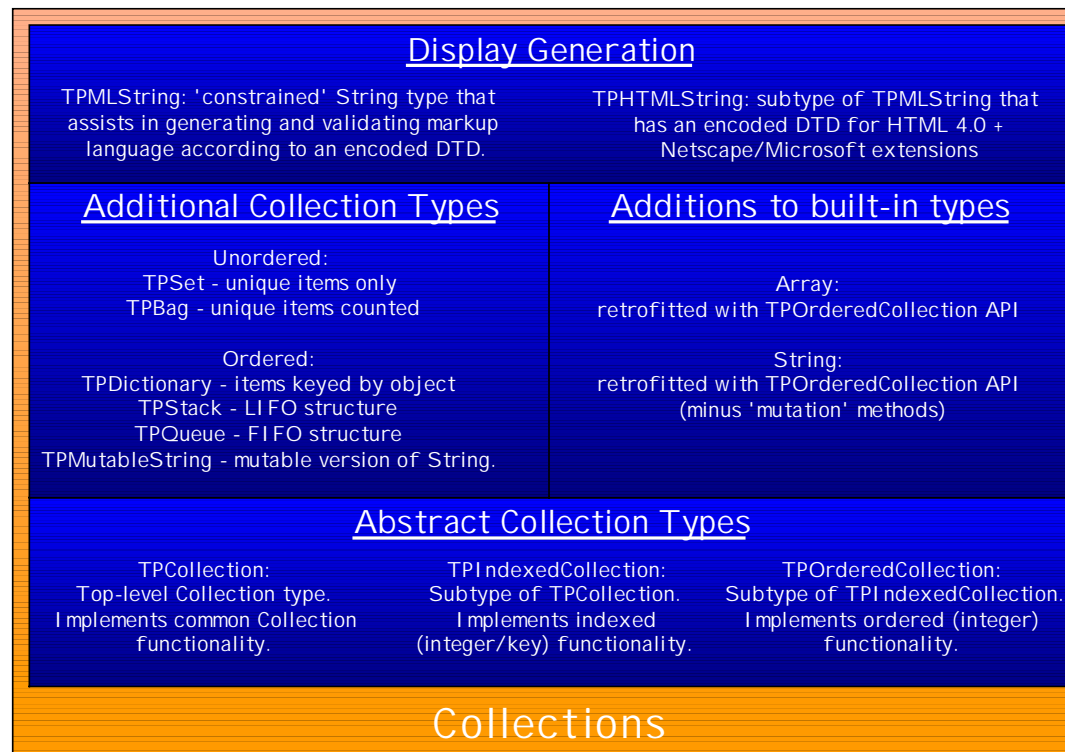
Kernel Components – Pt. 2



TIBET Collections

- Substantially enhances the built-in JavaScript data structures
- Provides common API that all collections must implement
 - Core: 64 methods
 - Indexed: 10 methods
 - Ordered: 10 methods
- Provides additional data structure types

Collection Components – Pt. 1



Collection Components – Pt. 2

Set Operations

Ability to test for equality or identity.

Containment: `contains()`, `containsAll()`,
`containsAny()`

Arithmetic: `difference()`, `disjunction()`,
`intersection()`, `union()`

Iteration

Separate iterators over a collection:
`next()`, `previous()`, `atStart()`, `atEnd()`,
`current()`, `currentIndex()`, `index()`

Simple iteration methods:
`perform()`, `select()`, `reject()`, `detect()`, `collect()`,
`injectInto()`, `transform()`

Additional methods

Accessing: `at()`, `atAll()`, `indexOf()`, `indexesOf()`, `lastIndexOf()`
Addition: `add()`, `addAll()`, `addAt()`, `addAllAt()`, `addWithCount()`, `append()`
Counting: `occurrencesOf()`
Manipulation: `empty()`, `reverse()`, `unique()`
Removal: `remove()`, `removeAll()`, `removeAt()`
Replacement: `atPut()`, `atAllPut()`, `replace()`, `replaceAll()`
Sizing: `getCapacity()`, `getSize()`,
Sorting: `getSortFunction()`, `isSorted()`, `sort()`, `sortByFunction()`

Collections

TIBET Data Type Conversion/Formatting/Validation

- Provides infrastructure for data type:
 - Conversion: converting from one type to another
 - Formatting: allowing types to express their representation in different ways
 - Validation: ensuring that a value can be successfully stored in a particular data type
- Provides data types for data commonly found in Web applications: dates, credit card #'s, e-mail addresses, etc.

Data Type Validation Components

Kernel validation functionality

Object validation mechanism built into the TIBET kernel

To validate, use the 'is' method:

```
myString.is('TPCreditCardType');
```

If 'is' doesn't exist (probably won't on "String"), the kernel sends 'validate<type>' to the validation type.

If a specific 'is'/'validate<type>' method is available for that data type, it will be used rather than the generic method

E.g. `is('TPCreditCardType')` becomes `isCreditCardType()`

This allows later optimization for conversion between specific types with no changes to calling code.

Constrained Types

TPConstrainedType:

Constraint registry for validation
min, max, default value, constrained value

Subtypes of TPConstrainedType:

TPAlphaType
TPAlphaNumericType
TPCreditCardType
TPIPv4AddressType,
TPEmailType (RFC822)

Data Types Validation

Data Type Conversion Components

Kernel conversion functionality

Type conversion mechanism built into the TIBET kernel

To convert, use the 'as' method:

```
myString.as('TPEmailType');
```

If 'as' doesn't exist (probably won't on "String"), the kernel sends 'from' to the conversion type.

If a specific 'as'/'from' method is available for that data type, it will be used rather than the generic method

E.g. `as('TPEmailType')` becomes `asTPEmailType()`

Data Types Conversion

Data Type Formatting Components

Prebuilt format types

TPDebugString, TPCGIString, TPSourceString

TPHTMLString (more sophisticated HTML functionality available in the Rendering component library)

'Virtual type' mechanism

Use 'as' functionality from kernel to format:

```
myPhoneNumber.as('999-999-9999');
```

Local programming allows customization of this mechanism on a per-instance basis

Kernel conversion functionality

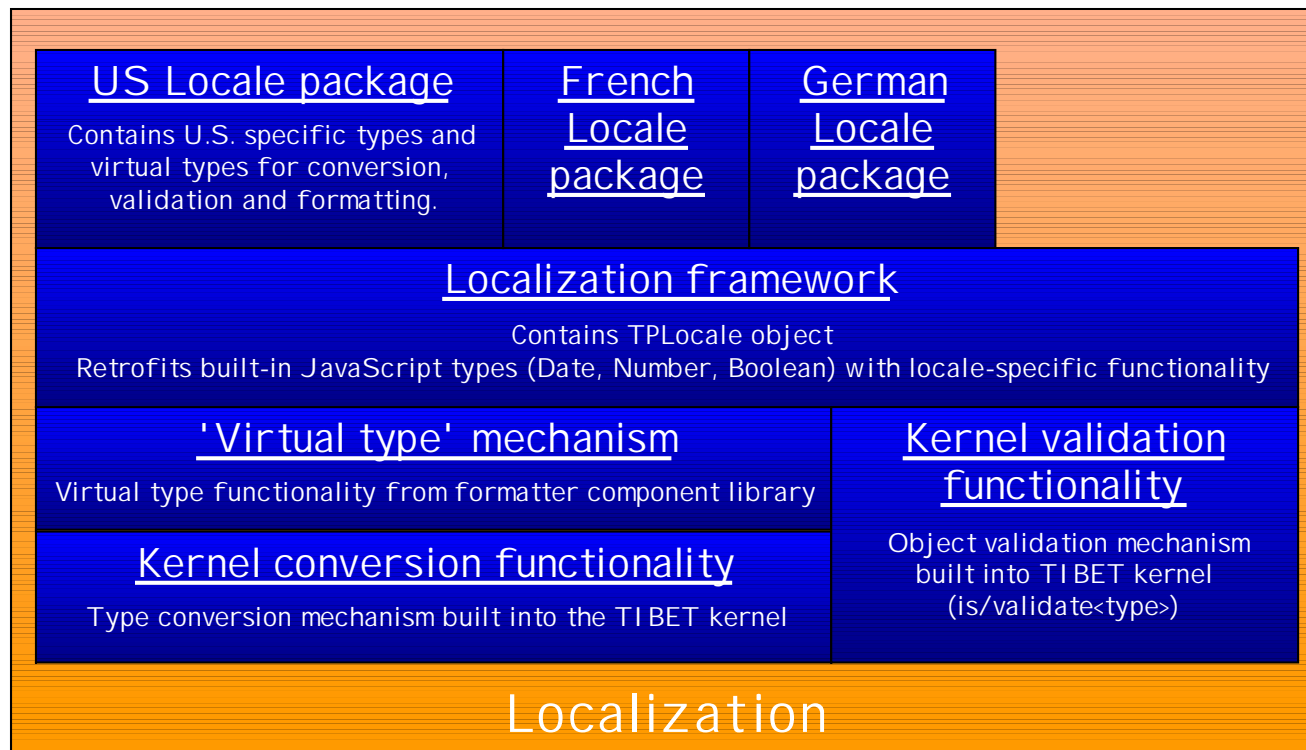
Type conversion mechanism built into the TIBET kernel

Data Types Formatting

TIBET Localization

- Provides a 'TPLocale' type for storing locale specific information
- Integrates seamlessly with the data type conversion / formatting / validation framework
 - Retrofits existing Date, Number, Boolean & String data types with locale-aware methods
- US locale with US specific data types are bundled in

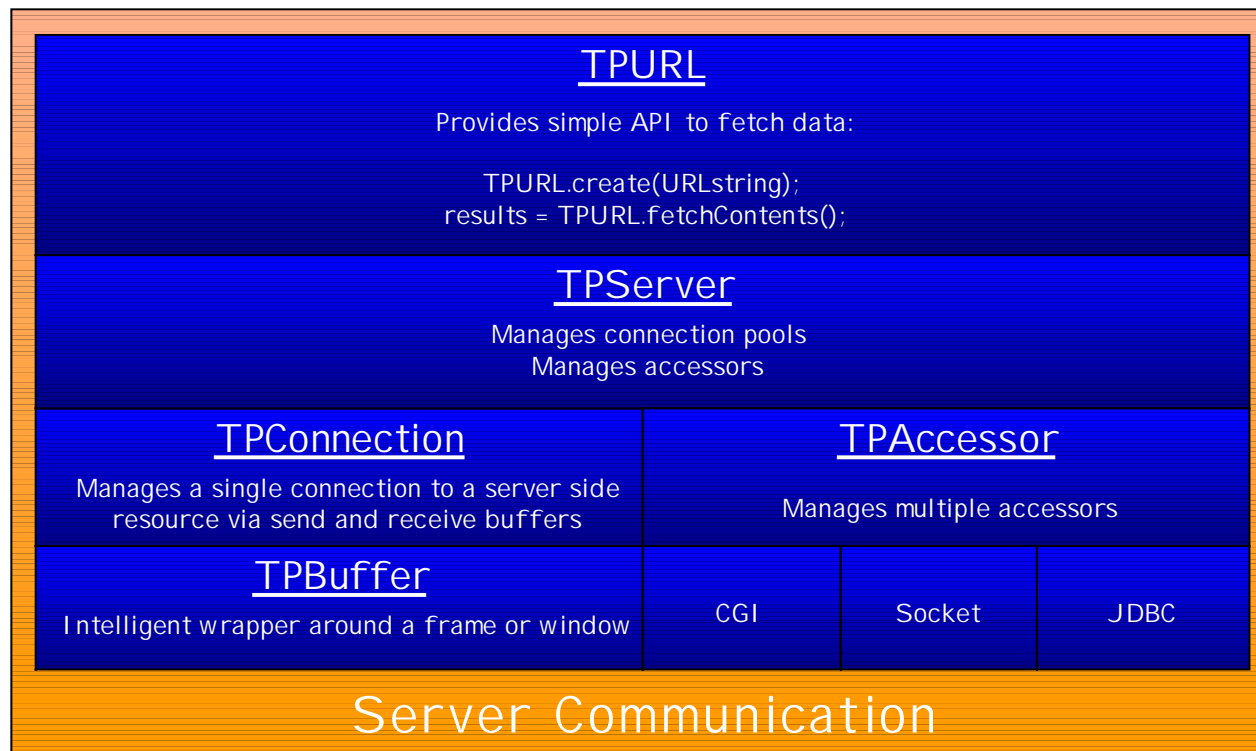
Localization Components



TIBET Server Communication

- Abstracts server communication
- Relieves programmer of the 'tyranny' of <FORM>s
 - One line of code to call any CGI and get results
- Provides ability to communicate multiple server queries in parallel
- Variety of communication strategies available (see below)

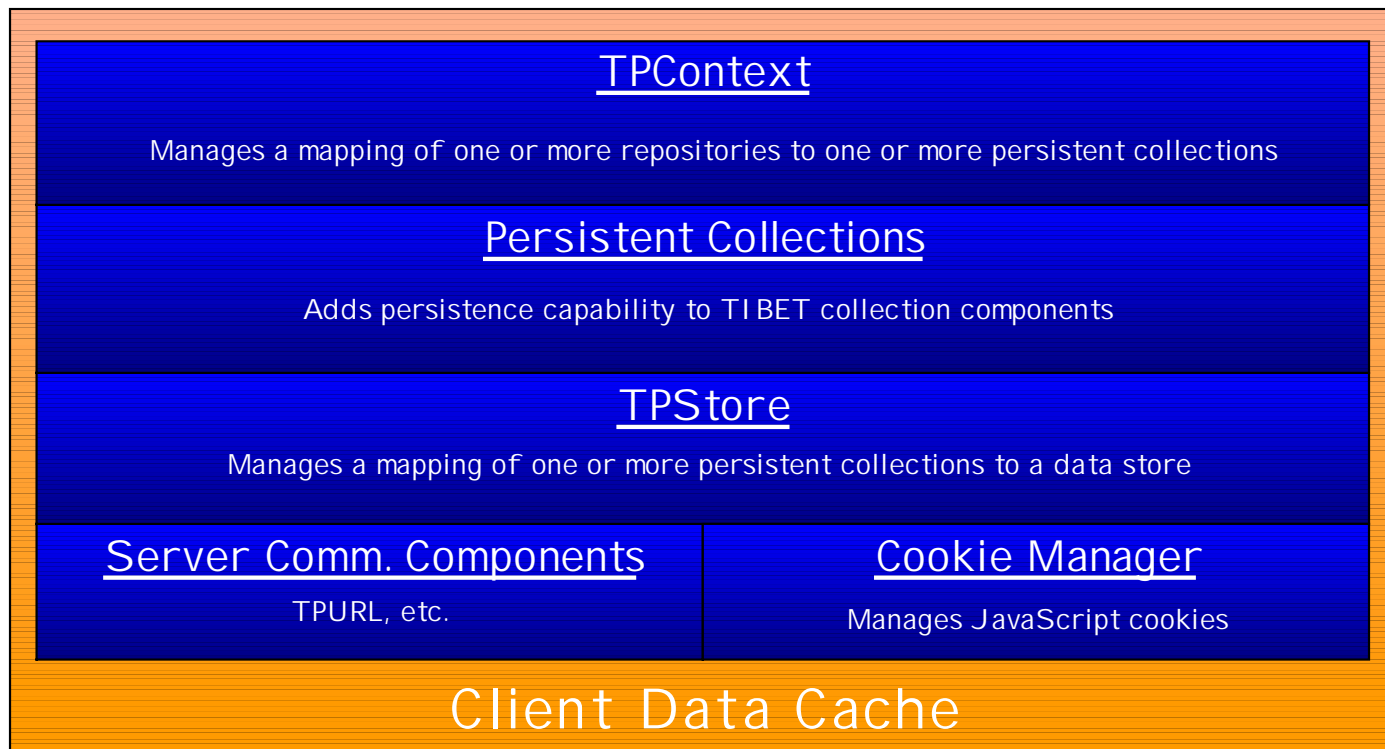
Server Communication Components



TIBET Client Data Cache

- Data manipulation in the *client*
- Eliminates session management for most applications
- Extension of TIBET Collection module
- Supports relational-like sorting and filtering
- Supports client side transaction management
 - Supports in-memory contexts
 - What-If? Scenarios
 - In-memory rollback
- Provides common abstraction layer over cookies

Client Data Cache Components



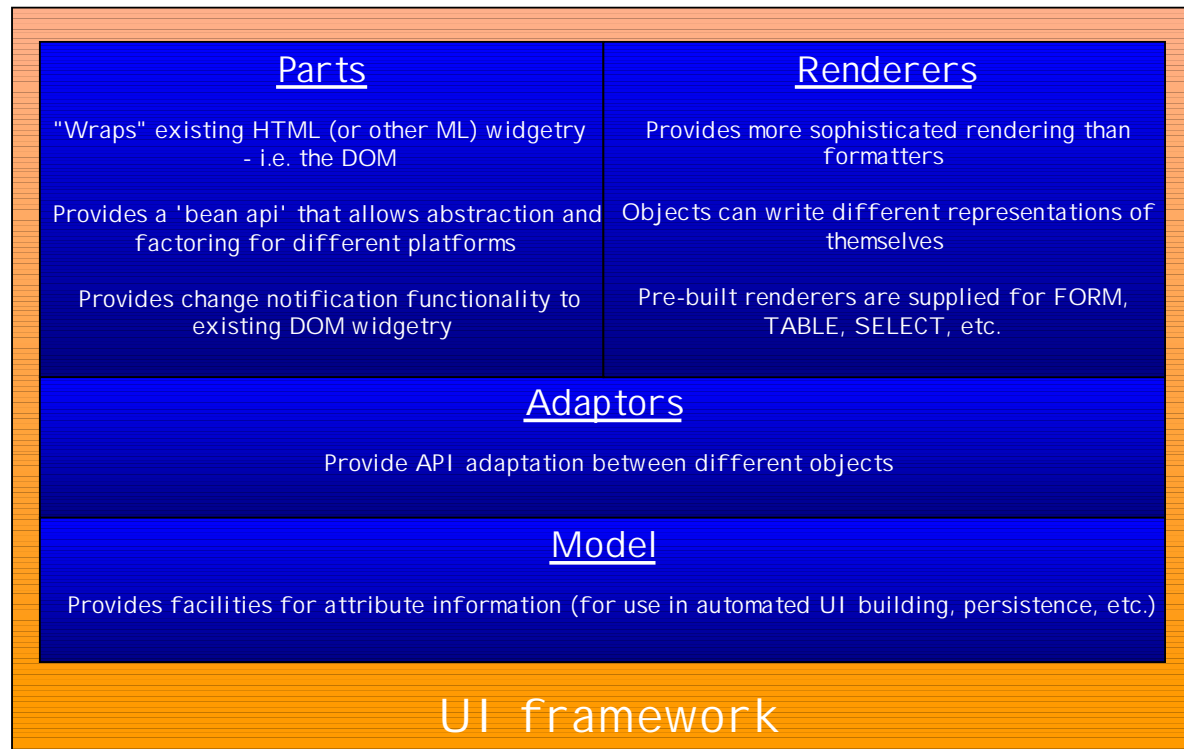
TIBET UI Framework – Pt. 1

- Renderers
 - Provides rendering capability in much more sophisticated ways than simple formatters
 - Objects can ‘show’ themselves in different ways by using different renderers
 - A more sophisticated version of formatters
- Parts:
 - Retrofits TIBET functionality onto generated HTML (or other ML) widgetry

TIBET UI Framework – Pt. 2

- Parts (continued)
 - Change notification: Parts keep themselves up-to-date with their underlying data model
 - get/set API: Allows Parts to integrate well with TIBET (which uses a get/set “bean” API)
- Adaptors:
 - Allows API adaptation between objects in the system (commonly used when binding data objects to UI objects)
 - Provides indirection necessary for change management

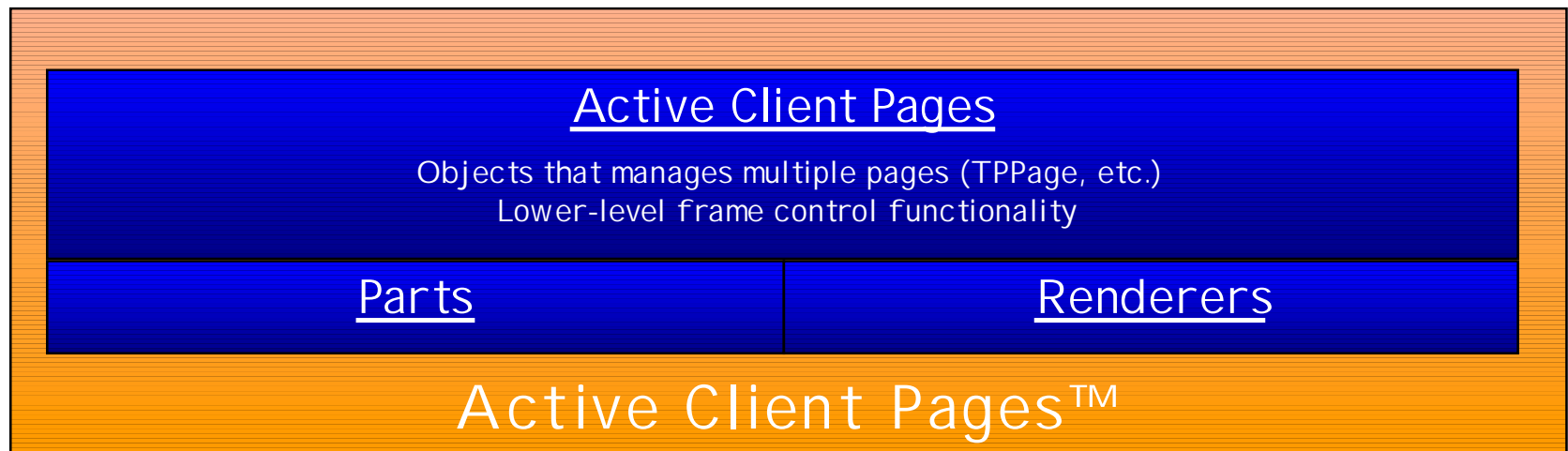
UI Framework Components



TIBET Active Client Pages™

- Allows page templates to be written by page designers
- Works like many of the “active pages” technologies (ASP, JSP, PHP), except the page<->data merge is done in the *client*
- Uses standard <SCRIPT></SCRIPT> tags in combination with the JavaScript entity (&{ };) syntax
- Leverages the browser’s built-in page cache
- One of TIBET’s key components to bandwidth reduction

Active Client Pages™ Components



TIBET Design & Coding Principles

- Designed before coded 😊
- Designed for reuse
 - Well factored
 - Tightly integrated
 - Well commented (60%+ of total bytes are comments) – ‘JavaDoc’ style with long attribute / method names
 - Comments & long names can be stripped / compressed when ‘production package’ is generated
- Designed around objects
 - Browser-specific issues handled via inheritance & factoring – NO “isNav/isIE” anywhere in ~40K lines of code.

Application Packaging – Pt. 1

- Assume compression available
 - TIBET is full featured
 - No hand optimization
- Don't transmit code that's not needed
- Application packages generated on demand
- Application packages can be compressed
 - In the highly commented TIBET code base, ratios of 16:1 are commonly seen

Application Packaging – Pt. 2

- Typical TIBET application is ~30-50K bytes – much less than the average Web home page (and it will be cached)
- The TIBET kernel is currently 10K bytes compressed and loads in
 - < 250ms (400Mhz PowerPC G3)
 - < 500ms (233Mhz Pentium II)

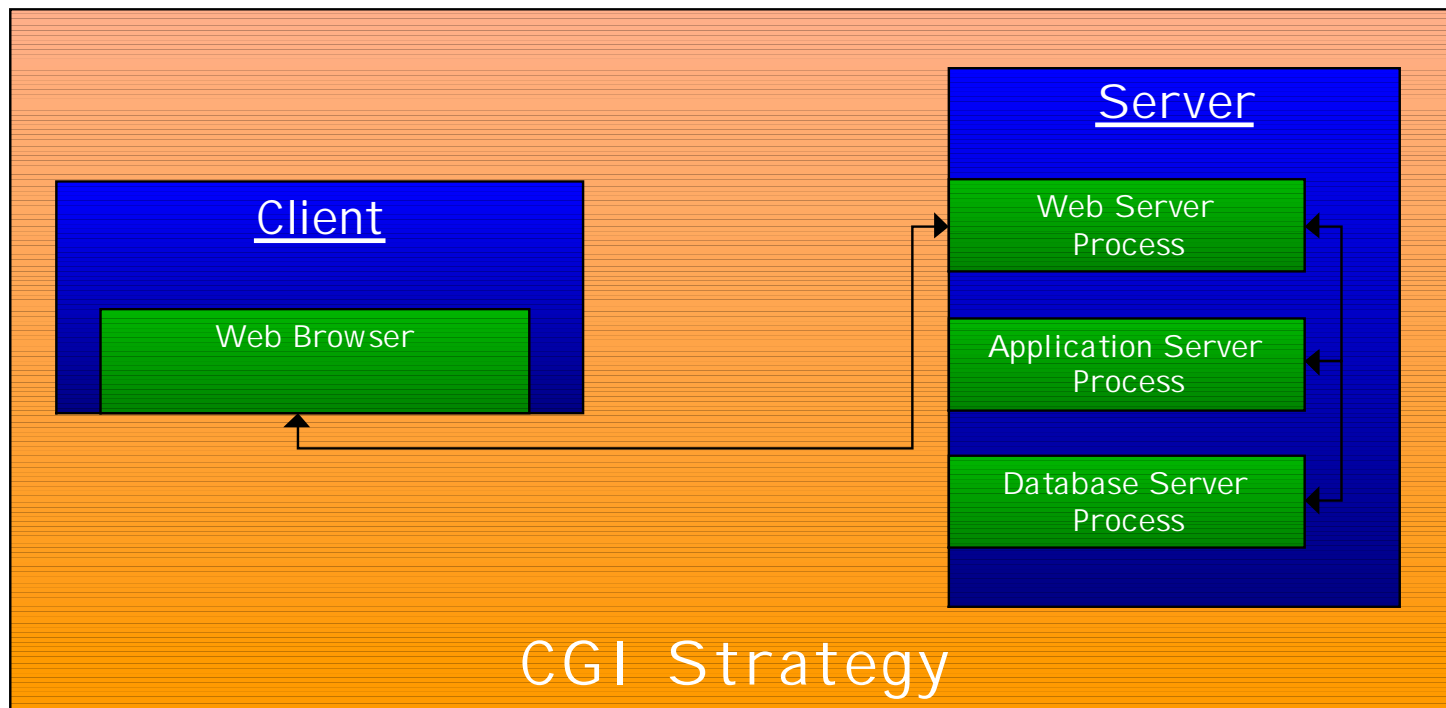


Demo



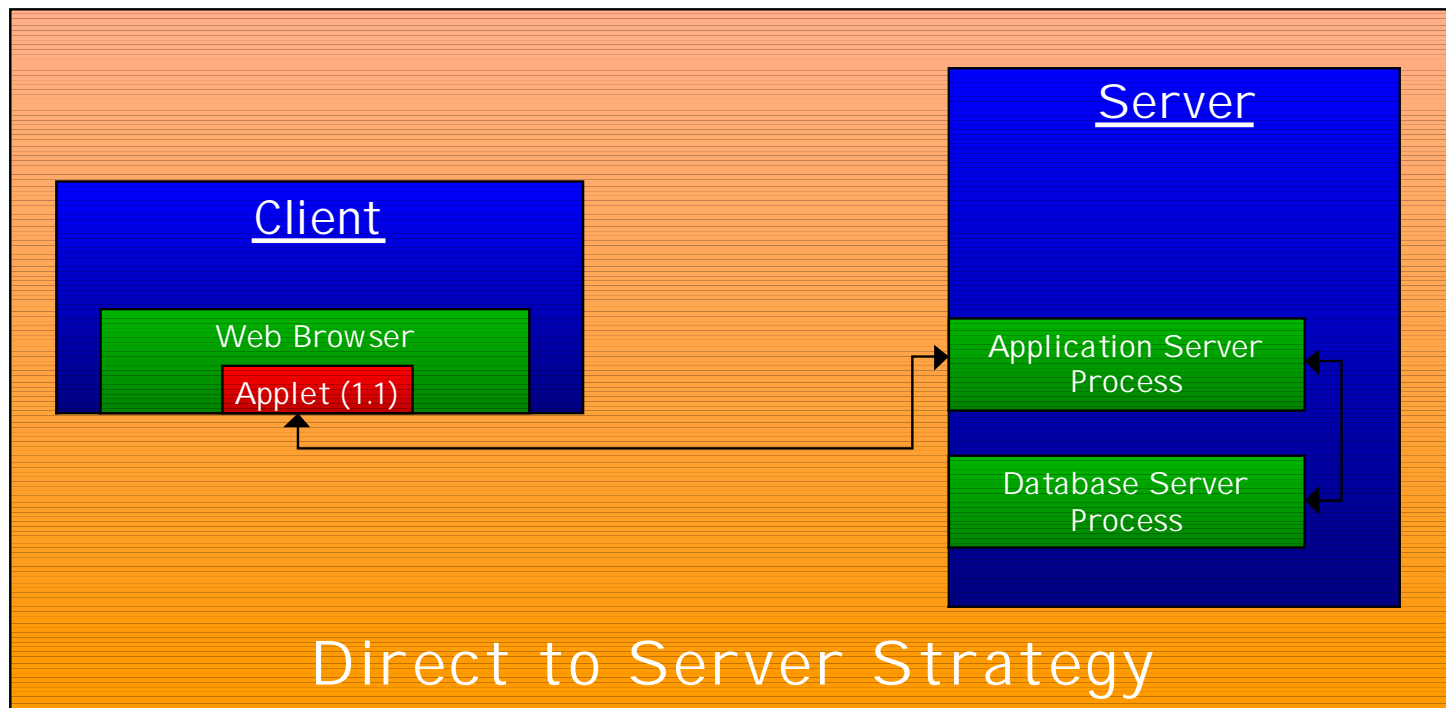
Communication Strategies

- CGI



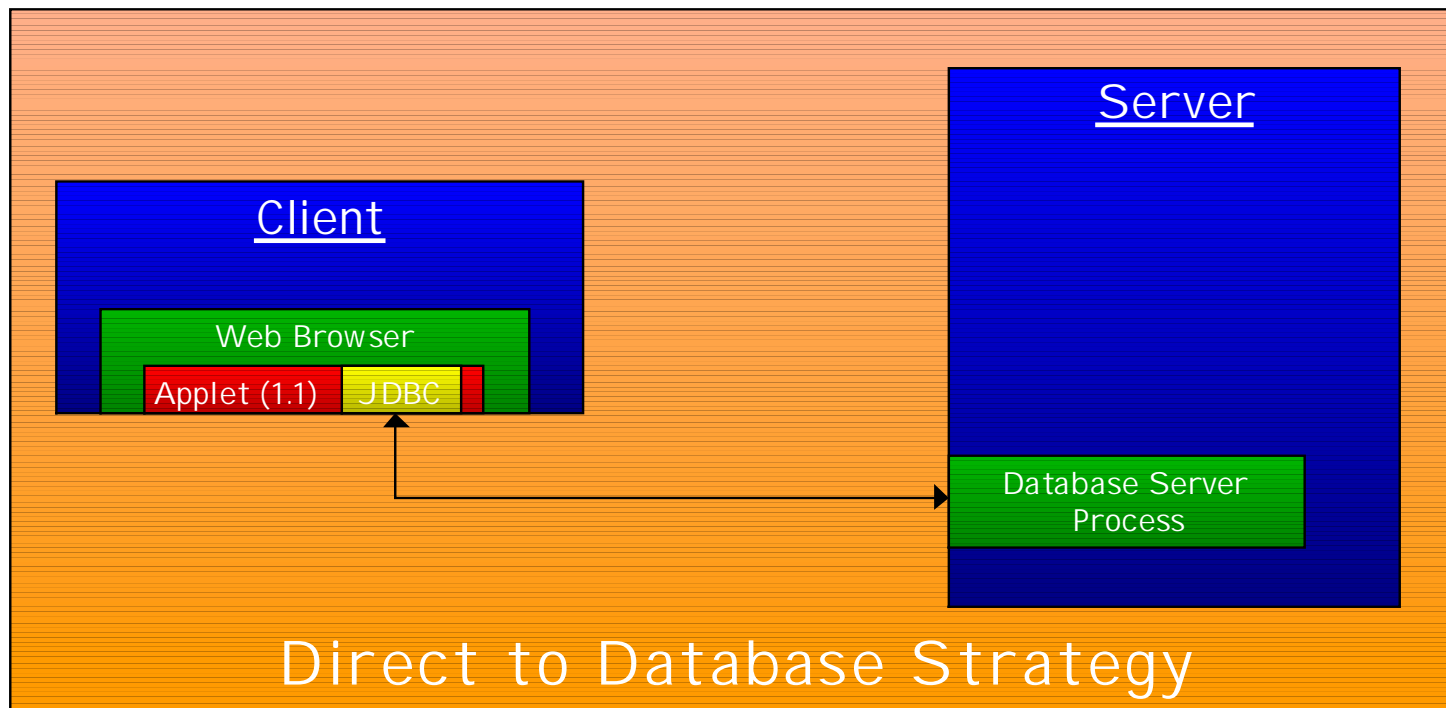
Communication Strategies

- Direct To Server



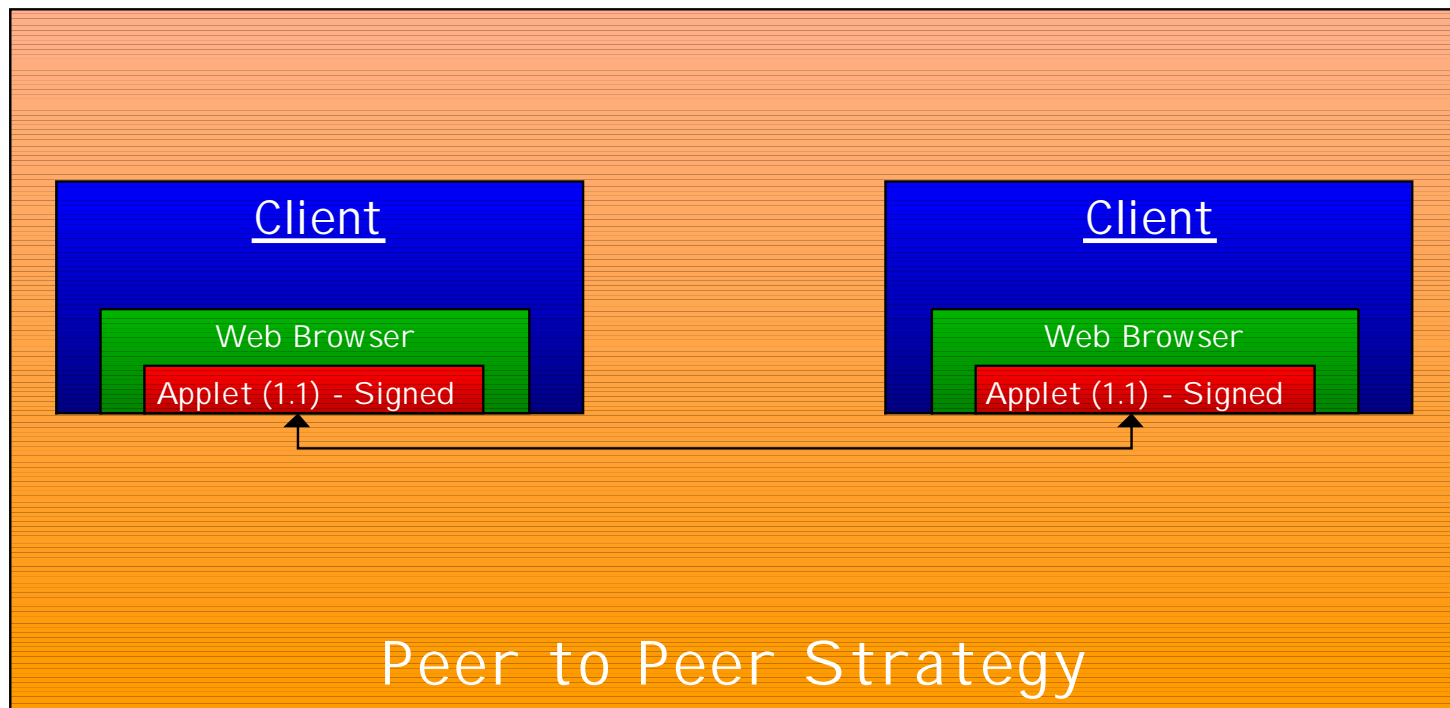
Communication Strategies

- Direct To Database



Communication Strategies

- Peer to Peer



Release Schedule

- Beta 1: mid to late March
 - Kernel
 - Collections
 - Server Communication
 - Data Type Conversion/Formatting/Validation
 - Localization
 - Active Client Pages™

Release Schedule

- Beta 2: late April
 - Client-side data cache (persistent collections)
 - UI framework: Parts & Renderers
 - Task Management
 - Basic DHTML widgets

Release Schedule

- Release 1.0: early June
 - Client data cache (TPStore & TPRepository)
 - More DHTML widgets
 - Distributed Event Server

Licensing

- Open Source
- Entire client library available under 2 licenses
 - GPL
 - Commercial
- Other components available under commercial licenses:
 - IDE
 - Event Server

Technical Pursuit Inc. Business Strategies

- Online Interactive Web Development services
- Corporate partnering
- Custom browser-based application development
- Training
- Technical support

Contact Us

- Technical Pursuit Inc.
 - By phone: (303) 626-3660
 - By email: info@technicalpursuit.com
 - By Web: www.technicalpursuit.com