# 実験2がはじまります

- Ubuntuを起動し、ログイン
- 実験2ソフトウェアのwebページへ

  **http://www.fos.kuis.kyoto-u.ac.jp/le2soft/**

  まずはブックマーク！
- 資料を確認
  - pdf版をダウンロードしておく
  - **サンプルプログラム**をダウンロードして展開

# 計算機科学実験及演習2

# 実験2の概要

- 火曜日3〜4限 (13:00〜16:15)
- ソフトウェア、ハードウェアそれぞれの実験を行なう
- 担当教員・TA、実施場所も違います

| | | |
|---|---|---|
| 前半：ソフトウェア | 10月7日〜11月18日<br>(7回) | 総合研究7号館<br>1階計算機演習室1 |
| 後半：ハードウェア | 12月2日〜1月27日<br>(7回) | 総合研究7号館<br>地下1階実験室 |

# 注意事項

- **ソフト・ハード両方**の課題を完成させること
- **出席**を取ります
  - 一定以上の出席が必須
  - 遅刻・早退もチェック
  - ソフト・ハードそれぞれでカウント
- **実験時間外の自習**が大前提
- 時間外の計算機室利用には申請書の提出が必要
  - 詳細は実験1のWEBページを参照

# ハード実験からの伝言

- 復習しておくこと！
  - 「電気回路と微分方程式」
  - 「論理回路」
- 実験レポートの書き方を自習すること！
  - シラバスに挙げた参考書などで

# 実験2ソフトウェア

HTTPサーバとクライアントの作成

# スタッフと連絡先

- 教員
  - 中澤 篤志
  - Marco Cuturi
  - 中澤 巧爾
- TA
  - 樋口 彰
  - 宮本 洋平
  - 佐々木 健人
  - 金 応教

le2soft@fos.kuis.kyoto-u.ac.jp

# 実験WEBページ

**http://www.fos.kuis.kyoto-u.ac.jp/le2soft/**

- ニュース、スケジュールなど
  - 適宜リロードして確認すること
- 資料、サンプルソースもここから
- 「課題提出状況」は要パスワード
  - ID = **student** / PASS = **memphis**

# ソフトウェア実験の概要

- HTTPサーバとクライアントの作成
  - 簡単なwebサーバと簡単なwebブラウザ
  - それぞれをCとJavaで（合計4つのプログラム）
- **Javaの解説はあまり行ないません**
  - 自習必須
  - サンプルプログラムを参考に
- 詳細はweb上の資料を参照
  http://www.fos.kuis.kyoto-u.ac.jp/le2soft/

「すべての人のためのJavaプログラミング」
立木秀樹，有賀妙子（共立出版）

# 課題

- 課題1（C言語）
  - **URL解析器**の作成
- 課題2（C言語）
  - HTTP1.1（のサブセット）に準拠した**クライアント**の作成
- 課題3（C言語）
  - HTTP1.1（のサブセット）に準拠した**サーバ**の作成
- 課題4（C言語）
  - **複数のクライアントからの接続に同時に対応**できるようサーバを改良
- 課題5（Java）
  - ↑と同機能の**サーバ・クライアントをJavaで**作る
  - クライアントに**リンクを辿る機能**を実装

# 課題

- 課題1（C言語）
  - **URL解析器**の作成
- 課題2（C言語）
  - HTTP1.1（のサブセット）に準拠した**クライアント**の作成

**報告書1**

- 課題3（C言語）
  - HTTP1.1（のサブセット）に準拠した**サーバ**の作成
- 課題4（C言語）
  - **複数のクライアントからの接続に同時に対応**できるようサーバを改良
- 課題5（Java）
  - ↑と同機能の**サーバ・クライアントをJavaで**作る
  - クライアントに**リンクを辿る機能**を実装

# 課題

- 課題1（C言語）
  - **URL解析器**の作成
- 課題2（C言語）
  - HTTP1.1（のサブセット）に準拠した**クライアント**の作成

**報告書1**

- 課題3（C言語）
  - HTTP1.1（のサブセット）に準拠した**サーバ**の作成
- 課題4（C言語）
  - **複数のクライアントからの接続に同時に対応**できるようサーバを改良

**報告書2**

- 課題5（Java）
  - ↑と同機能の**サーバ・クライアントをJavaで**作る
  - クライアントに**リンクを辿る機能**を実装

# 課題

- 課題1（C言語）
  - **URL解析器**の作成
- 課題2（C言語）
  - HTTP1.1（のサブセット）に準拠した**クライアント**の作成

**報告書1**

- 課題3（C言語）
  - HTTP1.1（のサブセット）に準拠した**サーバ**の作成
- 課題4（C言語）
  - **複数のクライアントからの接続に同時に対応**できるようサーバを改良

**報告書2**

- 課題5（Java）
  - ↑と同機能の**サーバ・クライアントをJavaで**作る
  - クライアントに**リンクを辿る機能**を実装

**報告書3**

# 実験の進め方

- web上の資料を参考に、勝手に進める
  - pdf版資料ダウンロード推奨
- 解らない事は積極的にTAに質問する
- 成績は出席・報告書・総合デモによる

# 成績について

- **出席**
  - 毎回開始時に出席を取る
  - 遅刻・早退時はTAにその旨を申し出る
- **報告書3回**（資料A章を参照）
  - 課題の結果をレポートにまとめる
  - 作成したプログラムのコードとともにメールで提出
  - 報告書3は、最終デモ完了確認後に提出
- **総合デモ**（11月18日）
  - プログラムの動作を一人ずつチェック

# 注意

- 各課題毎のディレクトリを作成することを強く勧める
  - 前課題のソースコードに**「上書き」しない**！
- 実験時間は「コーディングの時間」
  - 内容の把握は**実験時間外**に予習しておく
- Javaも**実験時間外**に自習すること
  - 実験で用意したサンプルコードも参考

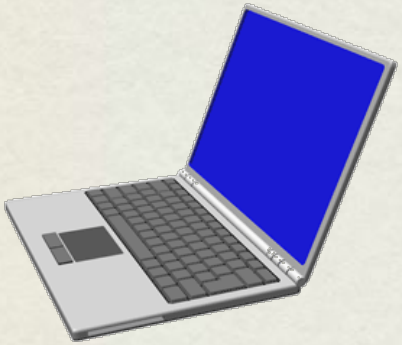
で結局、何を作ればいいのか？

# WWWの仕組み

HTTPサーバ

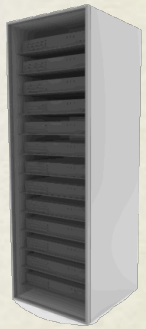
www.zzz.net

HTTPクライアント



zzz.netのホームページ
が見たい…

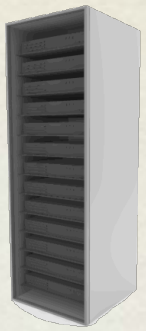
www.abc.com



www.xyz.co.jp

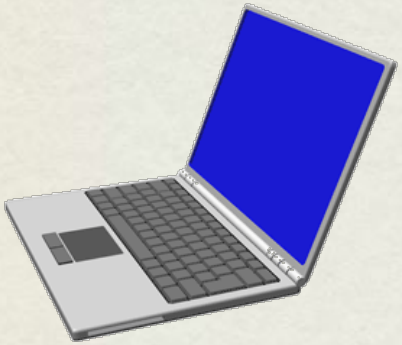
# WWWの仕組み

HTTPサーバ



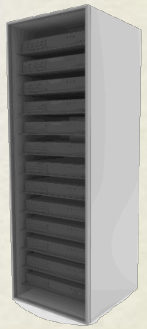
www.zzz.net

HTTPクライアント



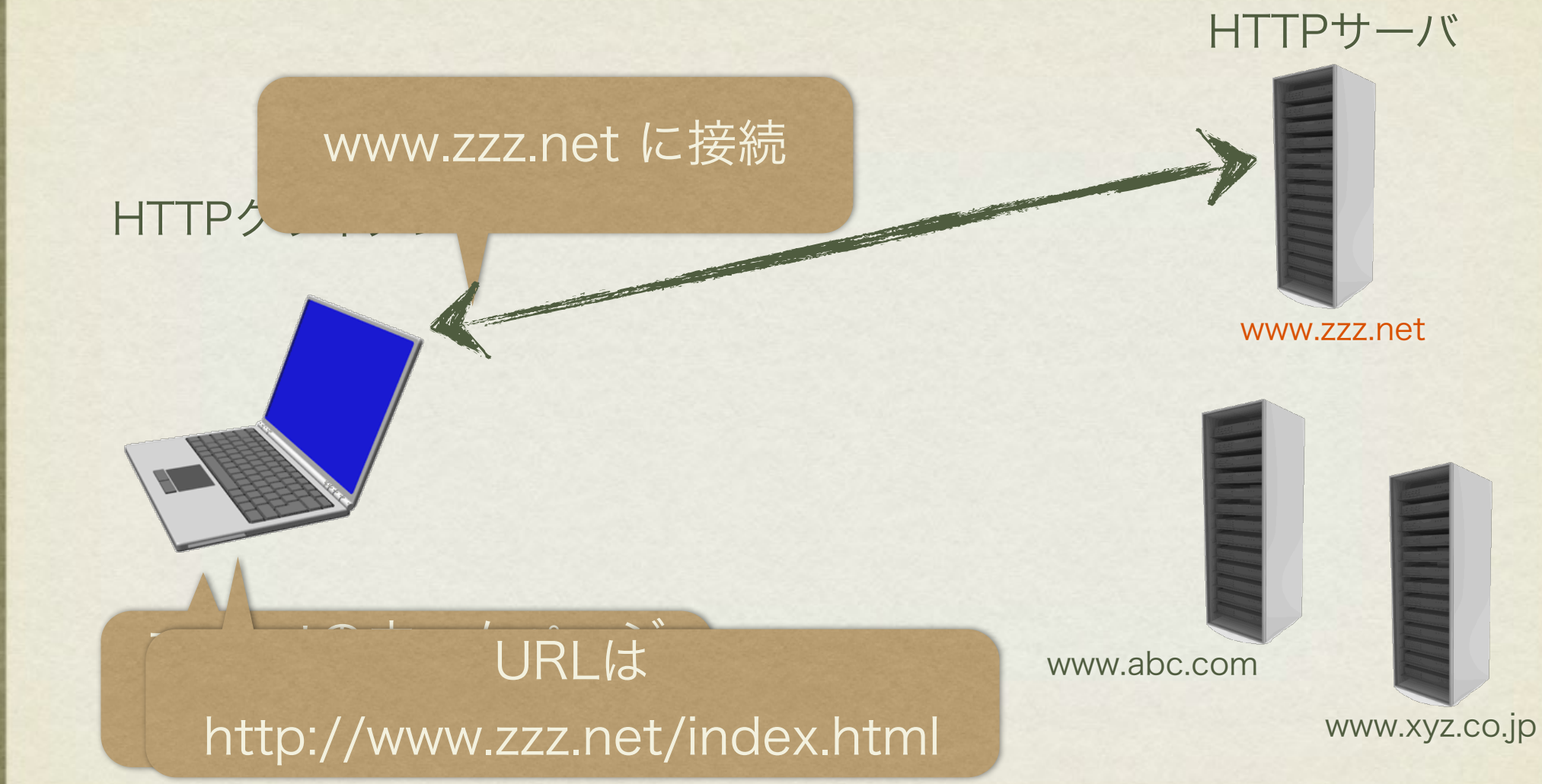
URLは
http://www.zzz.net/index.html

www.abc.com

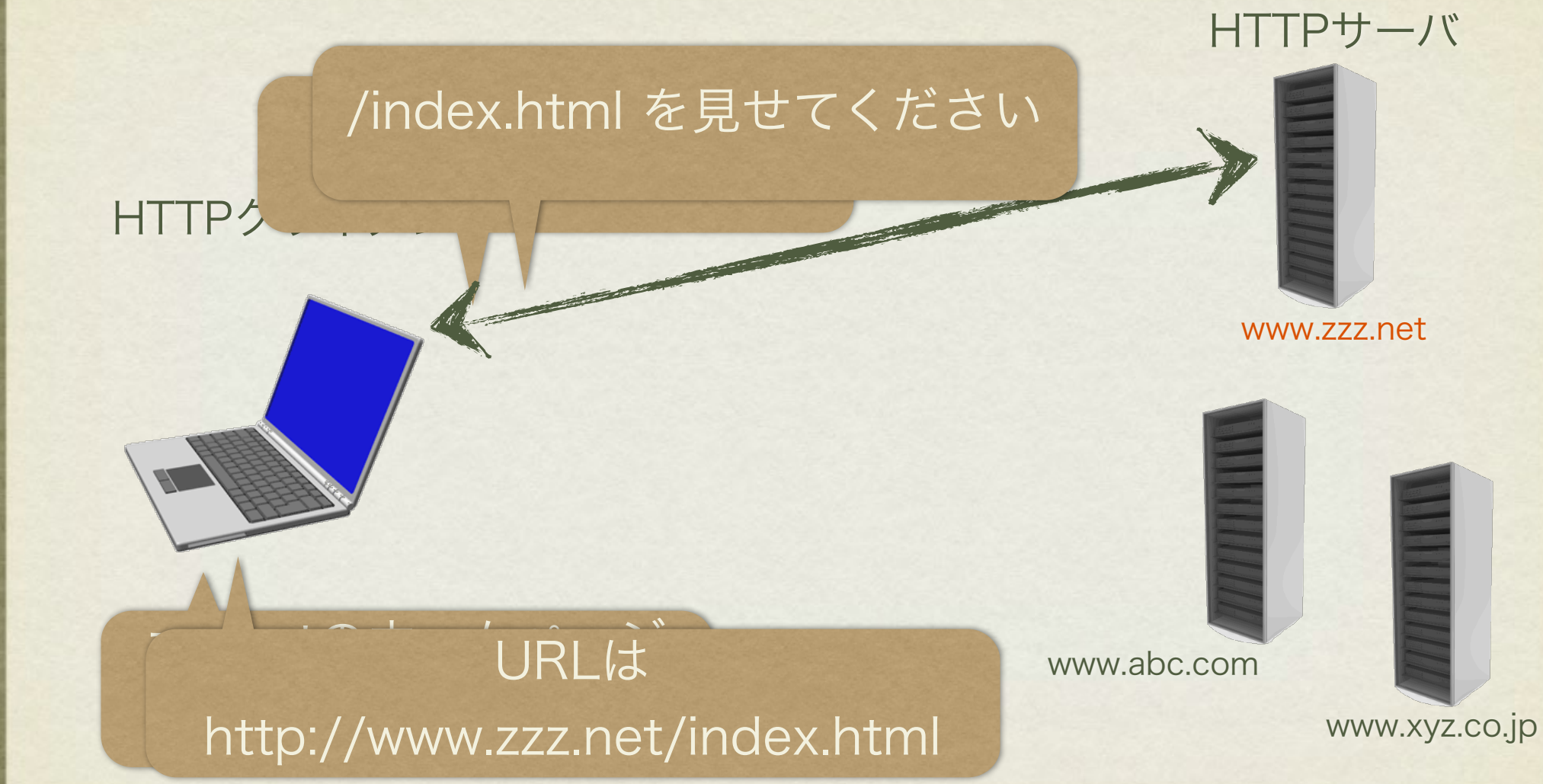


www.xyz.co.jp

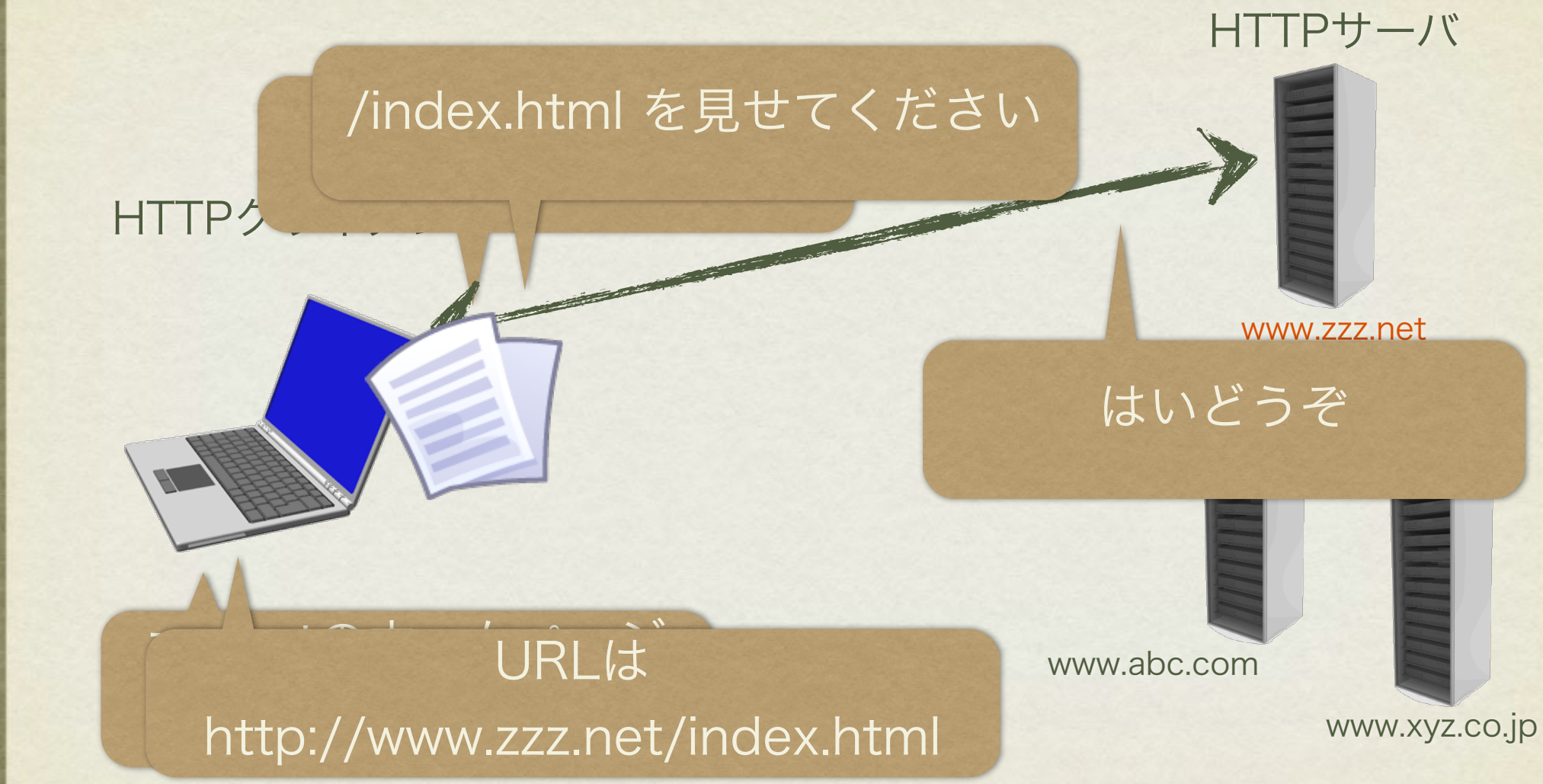
# WWWの仕組み


HTTPサーバ
www.zzz.net に接続
HTTPク
www.zzz.net
www.abc.com
www.xyz.co.jp
URLは
http://www.zzz.net/index.html

# WWWの仕組み

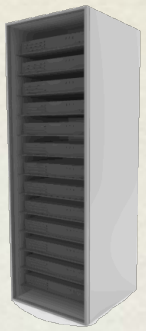

HTTPサーバ
/index.html を見せてください
HTTPク
www.zzz.net
URLは
http://www.zzz.net/index.html
www.abc.com
www.xyz.co.jp

# WWWの仕組み


HTTPサーバ
/index.html を見せてください
www.zzz.net
はいどうぞ
www.abc.com
www.xyz.co.jp
URLは
http://www.zzz.net/index.html

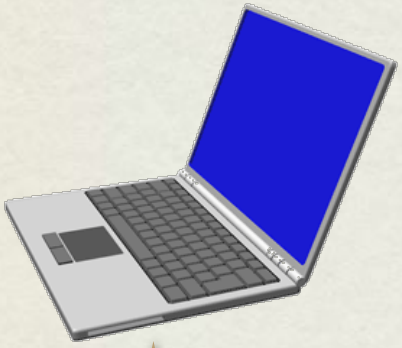
# HTTP = 通信のお約束

HTTPサーバ



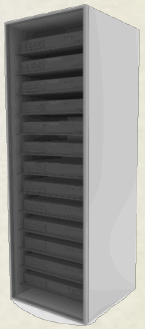
www.zzz.net

HTTPクライアント



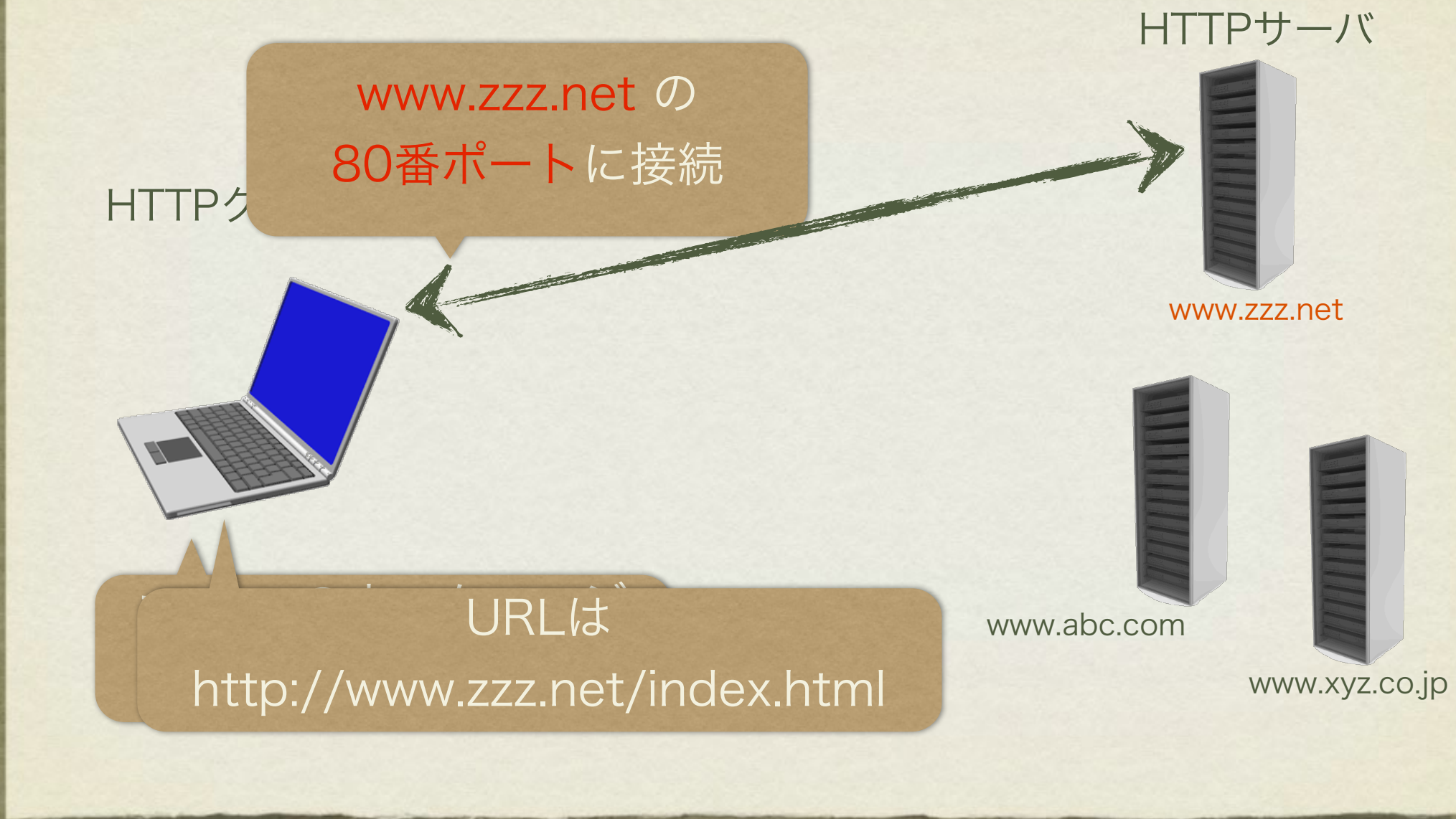
URLは
http://www.zzz.net/index.html

www.abc.com

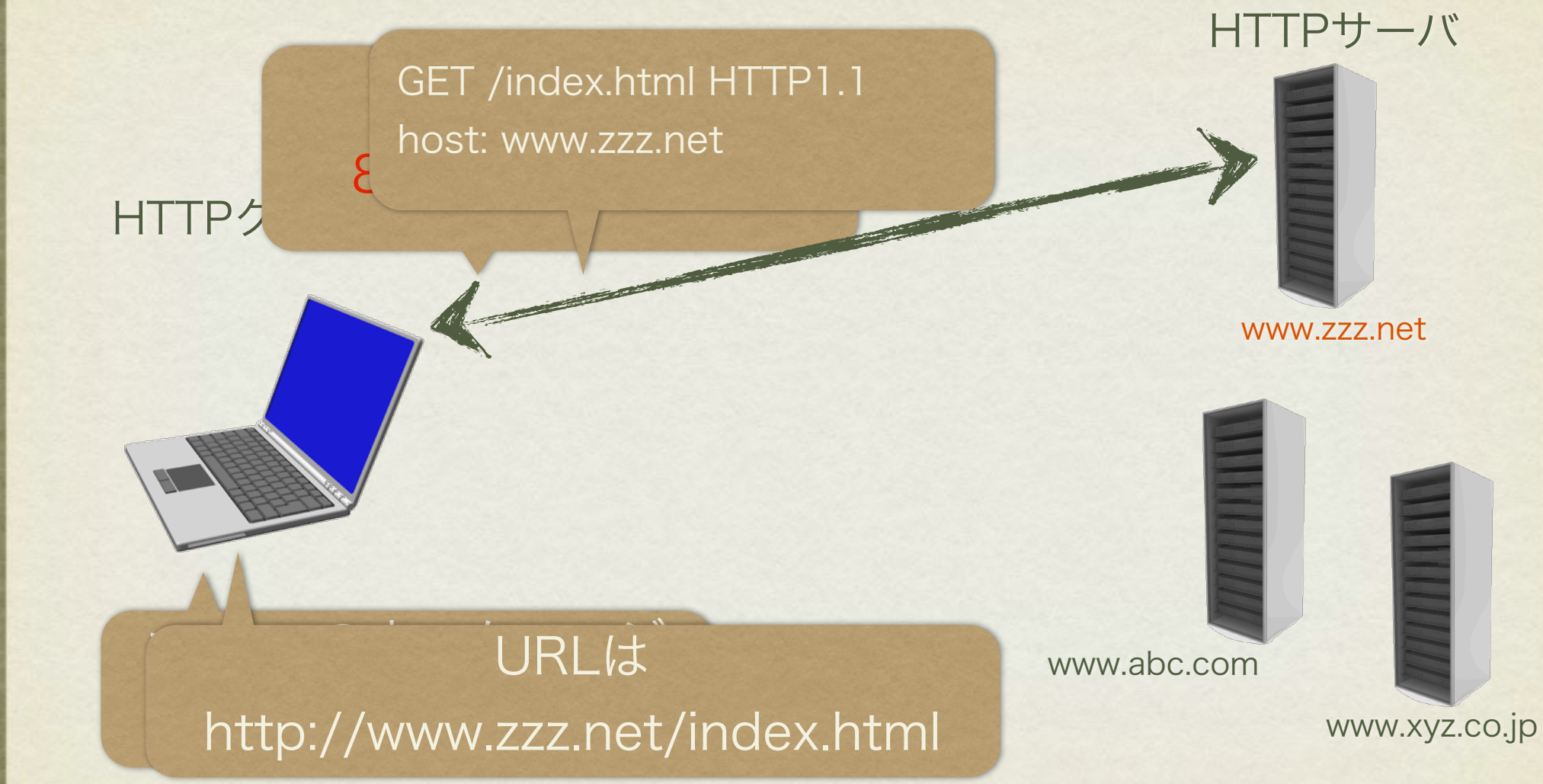


www.xyz.co.jp

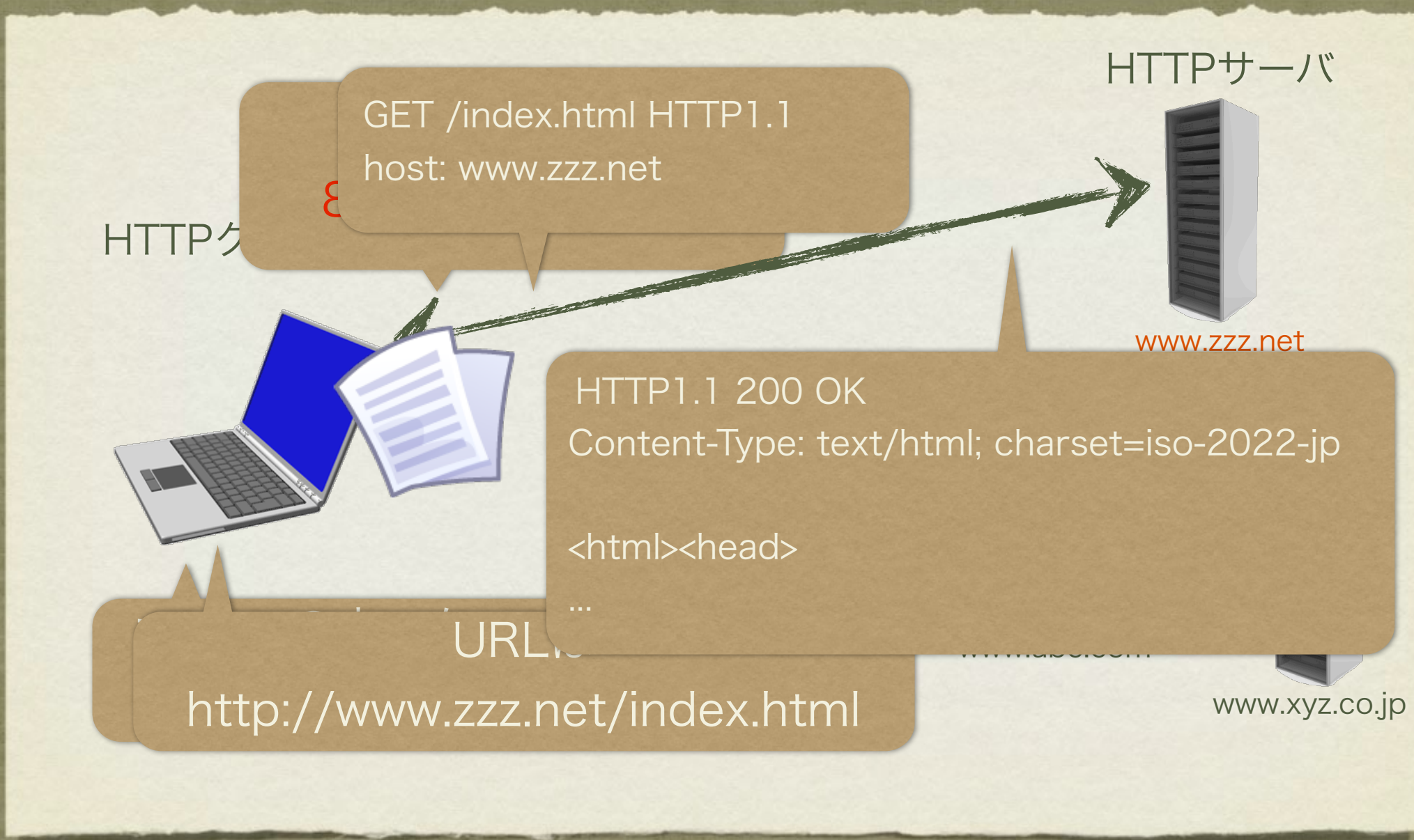
# HTTP = 通信のお約束


HTTPサーバ
www.zzz.net の
80番ポートに接続
HTTPク
www.zzz.net
www.abc.com
www.xyz.co.jp
URLは
http://www.zzz.net/index.html

# HTTP = 通信のお約束


HTTPサーバ
GET /index.html HTTP1.1
host: www.zzz.net
www.zzz.net
www.abc.com
www.xyz.co.jp
URLは
http://www.zzz.net/index.html

# HTTP = 通信のお約束


HTTPサーバ
GET /index.html HTTP1.1
host: www.zzz.net
www.zzz.net
HTTP1.1 200 OK
Content-Type: text/html; charset=iso-2022-jp
<html><head>
...
http://www.zzz.net/index.html
www.xyz.co.jp

# 動作の流れ

- サーバ (www.hoge.com)
  - 起動
  - ソケットを準備 (**socket**)
  - 80番ポートを割当て (**bind**)
  - 接続待ち (**listen**)
  - クライアントからの接続を受理 (**accept**)
  - 要求を受ける
  - index.htmlを送信
- クライアント
  - 起動
  - 入力 http://www.hoge.com/index.html
    - サーバ名 = www.hoge.com
    - ファイル名 = /index.html
  - ソケットを準備 (**socket**)
  - サーバの80番ポートに接続 (**connect**)
  - GETコマンドで /index.html を要求
  - 受信した index.html を表示

# 動作の流れ

- サーバ (www.hoge.com) **課題3&4**

起動

ソケットを準備 (**socket**)

80番ポートを割当て (**bind**)

接続待ち (**listen**)

クライアントからの接続を受理 (**accept**)

要求を受ける

index.htmlを送信

- クライアント **課題2**

起動 **課題1：URL解析器**

入力 http://www.hoge.com/index.html

サーバ名 = www.hoge.com

ファイル名 = /index.html

ソケットを準備 (**socket**)

サーバの80番ポートに接続 (**connect**)

GETコマンドで /index.html を要求

受信した index.html を表示

# 実際に完成品を
# 動かしてみる

# 課題1のヒント

- `char *strtok(char *str, const char *sep)`
  - 文字列**str**を、区切り文字**sep**で分割していく
  - 使い方にクセがある
  - この後のgdbの説明で一緒に解説
- `char *strchr(const char *str, int c)`
  - 文字列**str**中の文字**c**の位置を探す
  - 見つからなければNULL
  - 部分文字列の位置を探す**strstr**も

# デバッガ
# GDB

# GDB

- C言語用デバッガ
  - プログラムの段階的実行（ステップイン実行）
  - プログラム実行途中のメモリ状態などを確認
  - Segmentation faultなどメモリエラーの原因特定
- printf挿入なんて面倒なことはしない！

# GDBの使い方

1. -g オプションをつけて gccコンパイル

```
% gcc -g -o sample1 sample1.c
```

2. gdbを起動

```
% gdb sample1
…
(gdb) _
```

3. gdbコマンドによってデバッグ

# EMACS

- **~/.emacs** に次を追加

```
(global-linum-mode t)

(setq gdb-many-windows t)
```

- **M-x compile** でコンパイル
  - **-g** を忘れずに！
- **M-x gdb** でデバッガを起動
- 詳細は実験のWikiを参照

# 実際に動かしてみる

資料D.1参照